SQL Anywhere サーバ
文書バージョン: 17 – 2016-05-11

SQL Anywhere - SQL リファレンス

目次

1 SQL Anywhere サーバ - SQL リファレンス

このマニュアルでは、SQL Anywhere に含まれるシステムプロシージャとカタログ (システムテーブルとビュー) について説明します。また、SQL Anywhere での SQL 言語の実装 (探索条件、構文、データ型、関数) についても説明します。

このセクションの内容:

1.1 SQL 言語の要素

使用できる SQL 言語の要件はいくつかあります。

このセクションの内容:

文字列は、サイズが最高で 2 GB の文字シーケンスです。

定数 [11 ページ]
バイナリリテラルおよび文字列リテラルを定数として使用できます。

演算子 [14 ページ]
複数の算術演算子、文字列演算子、配列演算子、ビット処理演算子があります。

SQL 文の式 [32 ページ]
式は、評価して値を返すことのできる文です。

探索条件 [53 ページ]
探索条件は、WHERE 句、HAVING 句、CHECK 句、ジョインの ON フレーズ、または IF 式に指定された基準です。探索条件は、述部とも言います。

特別値 [83 ページ]
特別値は、式や、テーブル作成時のカラムのデフォルトに使用できます。

%TYPE および %ROWTYPE 属性 [110 ページ]
オブジェクトに対してデータ型を明示的に設定する以外にも、%TYPE および %ROWTYPE 属性を指定することによってデータ型を設定することもできます。

SQL 変数 [118 ページ]
サポートされる変数は、スコープ (接続、データベース、グローバル) によってグループ化できます。

コメント [122 ページ]
コメントは、SQL 文または文ブロックに説明テキストを付加するために使用します。データベースサーバは、コメントを実行しません。

名前付きパラメータ [123 ページ]
CALL 文、EXECUTE 文 (Transact-SQL)、DML 文の FROM 句、および TRIGGER EVENT 文から参照される関数およびプロシージャは、位置パラメータと名前付きパラメータをサポートしています。使用可能なパラメータの任意のサブセットを任意の順序で指定できます。

1.1.1 キーワード

各 SQL 文には 1 つまたは複数のキーワードが含まれています。SQL 文のキーワードでは大文字と小文字を区別しませんが、このマニュアルではキーワードを大文字で表記します。

たとえば、次の文では SELECT と FROM がキーワードになります。

```
SELECT *
   FROM Employees;
```

次の文は、上の文と同等です。

```
Select *
   From Employees;
select * from Employees;
sELECT * FRoM Employees;
```

キーワードの中には、二重引用符、角括弧、または逆引用符 (`...`) で囲まないと識別子として使用できないものがあります。このようなキーワードを予約語と呼びます。引用符で囲む必要のないキーワード (DBA など) もあり、これらは予約語ではありません。

このセクションの内容:

1.1.1.1 予約語

SQL キーワードには予約語だと見なされるものがあります。

予約語は、SQL 構文で使用するときに特別な処理が必要な語です。SQL 文で使用するキーワードの多くは予約語です (たとえば、select)。SQL 文で予約語を識別子として使用するには、二重引用符、角カッコ、または逆引用符で囲む必要があります。たとえば、SELECT という名前のテーブルの内容を取得するには、次の構文を使します。

```
SELECT *
   FROM "SELECT"
```

予約語のリストを取得するには、sa_reserved_words システムプロシージャを使用します。次に例を示します。

```
SELECT * FROM sa_reserved_words() ORDER BY reserved_word;
```

予約語に関するその他の注意事項:

- SQL キーワードは大文字と小文字が区別されず、次の各語は大文字、小文字、またはその任意の組み合わせで使用されます。次の語のいずれかと、大文字/小文字の区別のみが違う文字列はすべて予約語となります。
- non_keywords オプションを使用すると、キーワード制限をオフにできます。
- reserved_keywords オプションは、デフォルトで無効にされている個別のキーワードを有効にします。
- Embedded SQL を使用している場合、データベースライブラリ関数 sql_needs_quotes を使用すると文字列に二重引用符が必要かどうかを判別できます。文字列が予約語であるか、または通常識別子に使用できない文字が含まれている場合は、文字列に二重引用符を付けます。

関連情報

sa_reserved_words システムプロシージャ [1596 ページ]

1.1.2 識別子

識別子は、ユーザ ID、テーブル、カラムなど、データベースのオブジェクト名を表します。

識別子は、最大長が 128 バイトであり、アルファベット文字や数字、アンダースコア文字 (_)、アットマーク (@)、シャープ記号 (#)、ドル記号 ($) で構成されます。先頭に数字を使用することも可能ですが、識別子を引用符で囲む必要があります。他の特殊文字の使用も可能ですが、識別子を引用符で囲む必要があります。データベースの照合順は、どの文字がアルファベットまたは数字として扱われるかを指定します。

次の文字は、識別子では使用できません。

- " (二重引用符)
- 制御文字 (32 未満の序数または文字値 127 の ASCII 文字)
- 円記号
- 角括弧
- 逆引用符

> **注記**
>
> 16.0 より前のバージョンのデータベースを再ロードしている場合は、識別子に含まれる角括弧および逆引用符をすべて削除してください。そうしないと、再ロードが失敗します。

次の文字は、ユーザ名または役割名として使用されている識別子で使用することはできません。

- 先頭または末尾のスペース
- 先頭の一重引用符
- セミコロン

quoted_identifier データベースオプションが Off に設定されている場合には、二重引用符は SQL 文字列を区切り、識別子を区切るためには使用できません。ただし角括弧または逆引用符は、quoted_identifier の設定にかかわらず、識別子の区切りとして使用できます。quoted_identifier オプションのデフォルトは、Open Client 接続と jConnect 接続の場合は Off、それ以外の場合は On に設定されています。

文では、間接識別子を識別子の代わりに使用することもできます。間接識別子を使用すると、識別子を直接指定する代わりに、識別子を格納する変数の名前を指定できます。ただし、間接識別子はオブジェクトの選択用としてのみサポートされており、これらの使用方法には制限があります。

引用符付き識別子

次のいずれかの条件と一致する場合は、識別子を二重引用符、角括弧、または逆引用符 (`...`) で囲みます。

- 識別子が、先頭、末尾、または途中にスペースを含んでいる。
- 識別子の最初の文字が、アンダースコア (_)、アットマーク (@)、シャープ記号 (#)、ドル記号 ($) などの、アルファベット以外の文字である。たとえば、最初の文字が数字の場合。
- 識別子が、アルファベット文字、数字、アンダースコア (_)、アットマーク (@)、シャープ記号 (#)、ドル記号 ($) 以外の文字を含んでいる。
- 識別子が予約語である。

他のデータベース管理システムとの互換性を確保するため、識別子名に下記をはじめとする特殊文字を使用しないことを推奨します。

- 先頭または末尾のスペース
- 先頭の一重引用符
- セミコロン

標準

ANSI/ISO SQL 標準

128 文字までの識別子を作成する機能は、オプションの ANSI/ISO SQL 言語機能 F391 です。

例

次の文字列はすべて有効な識別子です。

- Surname
- "Client Name"
- `Client Name`
- [Surname]
- SomeBigName
- '[@myVar]'

このセクションの内容:

関連情報

予約語 [6 ページ]
間接識別子 [8 ページ]
TABLE REF データ型 [180 ページ]

1.1.2.1 間接識別子

文の実行時にオブジェクト名を設定する必要がある場合や、文の中で基本となるオブジェクトの名前が公開されるのを避ける場合は、間接識別子を使用します。

構文

文に間接識別子を指定する場合は、角括弧および逆引用符で囲みます (たとえば `'[@myVariable]'`)。ここで、@myVariable は操作している実際のオブジェクトの名前を格納する既存の変数名です。

備考

文の中で識別子 A が別の識別子 B を含む変数を指定している場合、識別子 A は間接識別子と呼ばれます。EXECUTE IMMEDIATE を使用してプロシージャ内に直接文を作成する場合 (特に、DML EXECUTE IMMEDIATE 文に識別子を代入し

ている場合) は、代わりに間接識別子を使用することを検討してください。間接識別子は、アプリケーションロジック内で EXECUTE IMMEDIATE を使用するよりも安全です。

アプリケーションロジックに間接識別子を構築すると、製品の動的機能が向上します。たとえば、アプリケーションがテーブル識別子の一部にテーブル登録タイムスタンプ情報を使用して、定期的にテーブルを作成するとします (たとえば、CurrentOrders023003032015 です。ここで 023003032015 はテーブルが作成されたタイムスタンプです)。今度は、この動的に指定したテーブルへの問い合わせを必要とするプロシージャがアプリケーションにあるとします。テーブル名を格納する @currentOrders という変数を宣言し、テーブルが作成するたびに変数を更新することができます。また、テーブルの間接識別子 (`'[@currentOrders]'`) を含めるようプロシージャを修正できます。プロシージャが呼び出されると、間接識別子が変数の値に置き換えられ、実際のテーブル名を指定したかのようにプロシージャが実行されます。

間接識別子は、次のオブジェクトの明示的な識別子の代用として、SELECT 文、プロシージャの関数の呼び出し、および DML 文でサポートされています。

- テーブル
- カラム
- ミューテックス
- セマフォ
- オブジェクト所有者として指定された場合のユーザ ID (たとえば `owner.object-name`)

間接識別子は、ミューテックスとセマフォのステータスを変更する文 (たとえば、WAITFOR SEMAPHORE 文) でもサポートされています。

文の実行前に、間接識別子が参照先の変数に格納されている値に置き換えられ、権限チェックが実行されます。

間接識別子の値の最大長は 128 バイトで、CHAR 型、VARCHAR 型、または LONG VARCHAR 型を指定できます。

識別子が必要な文では、カラムまたはテーブルの名前の指定に使用する間接識別子が NULL、空の文字列、または名前の場合、結果がエラーになります。ただし、テーブルの所有者など、修飾名のオプション部分として間接識別子が使用される場合は、間接識別子が NULL になる場合があります。識別子のオプション部分が NULL の場合は、指定されていない場合と同様に扱われます。

間接識別子は識別子の一部分を置き換えます。置き換えることができません。完全な指定を置き換えることはできません。たとえば、変数 @var に `'GROUPO.Employees'` が設定されている場合、GROUPO.Employees テーブルで間接識別子 (たとえば、`SELECT * FROM '[@var]'`) を使用して SELECT 操作を実行しようとすると、エラーが返されます。代わりに、名前のユーザ部分を格納する変数を作成し、間接参照 (たとえば、`'[@owner]'.'[@var]'`) を使用して両方のオブジェクトを参照する必要があります。

間接識別子の使用とテーブル参照変数の使用は重複します。どちらも、間接的にテーブルを参照する方法です。ただし、テーブル参照変数は、現在のコンテキストでアクセスできないテーブルへのアクセスを提供することができますが、間接識別子はできません。また、テーブル参照値は作成時に解決されますが、間接識別子は実行時に解決されます。

権限およびパーミッション

文で間接的に参照されているオブジェクトに対する権限は、間接識別子が評価されるときにチェックされ、文の実行前に強制されます。

例

次に示す例の多くは、CREATE VARIABLE 文を使用して変数が作成されています。これにより、Interactive SQL での例の試行が簡単になります。ただし、より適切なシナリオは、プロシージャまたは関数のスコープ内で変数を宣言 (DECLARE 文) し、次に、後続の文で間接識別子の一部として変数を参照するか、パラメータとして変数を渡すことです。

次の例では、GROUPO.Employees テーブルのカラムの名前 (Surname) を保持する @col という変数を作成します。SELECT 文は、間接識別子 (`'[@col]'`) を指定して Employees.Surname カラムの内容を問い合わせます。

```
CREATE OR REPLACE VARIABLE @col LONG VARCHAR = 'Surname';
SELECT E.'[@col]' FROM GROUPO.Employees E;
```

次の例では、間接識別子を使用してテーブルを問い合わせる方法を示します。

```
CREATE OR REPLACE VARIABLE t_owner LONG VARCHAR = 'GROUPO';
CREATE OR REPLACE VARIABLE t_name LONG VARCHAR = 'Employees';
SELECT * FROM '[t_owner]'.'[t_name]';
```

次の例では、テーブル参照を受け取る IN パラメータ (@tableref)、カラムの名前を受け取る IN パラメータ (@columnname)、および削除するユーザの ID を反映した整数値を受け取る IN パラメータ (@value) を使用してプロシージャを作成します。プロシージャの本文では、パラメータを使用してテーブルから必要なレコードを削除する方法を定義します。

```
CREATE PROCEDURE mydelete( IN @tableref TABLE REF,
                           IN @columnname LONG VARCHAR,
                           IN @value INT )
SQL SECURITY INVOKER
NO RESULT SET
BEGIN
    DELETE FROM TABLE REF (@tableref) AS T WHERE T.'[@columnname]' = @value;
END;
CALL mydelete( TABLE REF ( FTEmployee ), 'employee_id', @employee_to_delete);
CALL mydelete( TABLE REF ( FTStudent ), 'student_id',  @student_to_delete);
```

最初の CALL 文では、データベースサーバが @employee_to_delete 変数に格納されている値に一致するローの FTEmployee.employee_id カラムを検索し、そのローを削除します。2 番目の CALL 文では、データベースサーバが @student_to_delete 変数に格納されている値に一致するローの FTStudent.student_id カラムを検索し、そのローを削除します。

関連情報

SQL 変数 [118 ページ]
TABLE REF データ型 [180 ページ]
識別子 [6 ページ]

1.1.3 文字列

文字列は、サイズが最高で 2 GB の文字シーケンスです。

文字列は SQL で次の用途に使用されます。

- 文字列リテラル。文字列リテラルとは、一重引用符 (') で囲まれ、シーケンスで並べられた文字のことです。文字列リテラルは特定の定数値を表し、文字で簡単に入力できない特殊文字のエスケープシーケンスを含めることができます。
- CHAR データ型または NCHAR データ型のカラム値または変数値。
- 式の評価結果。

文字列の長さは 2 つの方法で測定できます。

バイト長

バイト長は、文字列のバイト数です。

文字長

文字長は、文字列の文字数です。また使用している文字セットによって変わります。

cp1252 などの SBCS の場合、バイト長と文字長は同じです。マルチバイト文字セットの場合、文字列のバイト長は文字長以上です。

関連情報

文字列リテラル [13 ページ]

1.1.4 定数

バイナリリテラルおよび文字列リテラルを定数として使用できます。

このセクションの内容:

1.1.4.1 バイナリリテラル

バイナリリテラルは、0 ～ 9 の数字、大文字と小文字の A ～ F で構成される 16 進文字です。

バイナリデータをリテラルとして入力するときは、データの前に 0x (1 つのゼロと 1 つの x) を指定します。このプレフィクスの右側には、偶数の桁数を指定します。たとえば、39 の 16 進数は 0027 であるため、0x0027 のように表されます。

0x12345678 の形式の 16 進定数は、バイナリ文字列として処理されます。0x の後ろに桁数を無制限に追加できます。

バイナリリテラルは、バイナリ定数とも呼ばれます。バイナリリテラルをおすすめします。

このセクションの内容:

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、バイナリ文字列を整数に変換できます。

1.1.4.1.1 16 進値との変換

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、バイナリ文字列を整数に変換できます。

CAST 関数と CONVERT 関数は、16 進定数を TINYINT、符号付きと符号なしの 32 ビット整数、符号付きと符号なしの 64 ビット整数、NUMERIC などに変換します。HEXTOINT 関数は、16 進定数を符号付き 32 ビット整数にのみ変換します。

CAST 関数によって返される値が 8 桁を超えることはできません。8 桁を超える値では、エラーが返されます。8 桁未満の値の左側に 0 が追加されます。たとえば、次の引数は値 -2,147,483,647 を返します。

```
SELECT CAST ( 0x0080000001 AS INT );
```

次の引数はエラーを返します。これは、符号付き 32 ビット整数として 10 桁の値を表現できないためです。

```
SELECT CAST ( 0xff80000001 AS INT );
```

符号付き 32 ビット整数として値を表現できる場合、HEXTOINT 関数によって返される値は 8 桁を超えることができます。HEXTOINT 関数は、数値、大文字または小文字の A ～ F のみで構成される文字列リテラルまたは変数を受け入れます (0x プレフィクスが付いている場合、付いていない場合の両方)。16 進値の右から 8 桁目が数 8 ～ 9 か大文字または小文字の A ～ F であるか、その前の桁がすべて大文字または小文字の F の場合、その 16 進値は負の整数値になります。

次の引数は、値 -2,147,483,647 を返します。

```
SELECT HEXTOINT( '0xFF80000001' );
```

```
SELECT HEXTOINT( '0x80000001' );
```

```
SELECT HEXTOINT ( '0xFFFFFFFFFFFFFFFF80000001' );
```

次の引数はエラーを返します。これは、引数が符号付き 32 ビット整数として表現できない正の整数値を表しているためです。

```
SELECT HEXTOINT( '0x0080000001' );
```

関連情報

CAST 関数 [データ型変換] [264 ページ]
CONVERT 関数 [データ型変換] [283 ページ]
HEXTOINT 関数 [データ型変換] [383 ページ]
INTTOHEX 関数 [データ型変換] [406 ページ]

1.1.4.2 文字列リテラル

文字列リテラルとは、一重引用符 (') で囲まれ、シーケンスで並べられた文字のことです。

たとえば、`'Hello world'` は CHAR 型の文字列リテラルです。バイト長は 11 であり、文字長も 11 です。

文字列リテラルは、文字列定数、リテラル文字列、または単に文字列とも呼ばれます。文字列リテラルをおすすめします。

引用符で囲んだ値の前に N を付けることによって、NCHAR 文字列リテラルを指定できます。たとえは、`N'Hello world'` は NCHAR 型の文字列リテラルです。バイト長は 11 であり、文字長も 11 です。NCHAR 文字列リテラル内のバイトは、データベースの CHAR 文字セットを使用して解釈され、その後 NCHAR に変換されます。構文 `N'string'` は、`CAST( 'string' AS NCHAR )` の短縮形式です。

このセクションの内容:

エスケープシーケンス [13 ページ]
場合によっては、通常の方法では入力できない文字を文字列リテラルに配置する必要があります。たとえば、制御文字 (改行文字など)、一重引用符 (それ以外の場合、文字列リテラルの末尾を示す)、16 進のバイト値などです。このような場合、エスケープシーケンスを使用します。

1.1.4.2.1 エスケープシーケンス

場合によっては、通常の方法では入力できない文字を文字列リテラルに配置する必要があります。たとえば、制御文字 (改行文字など)、一重引用符 (それ以外の場合、文字列リテラルの末尾を示す)、16 進のバイト値などです。このような場合、エスケープシーケンスを使用します。

次に、文字列リテラルにエスケープシーケンスを使用する例を示します。

- 一重引用符は、文字列リテラルの開始と末尾を示すときに使用されます。そのため、文字列の一重引用符は、次のように一重引用符を追加してエスケープする必要があります。`'John''s database'`
- 円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字と解釈されます。たとえば、¥q であれば、円記号と q が挿入されます。
  16 進のエスケープシーケンスは、文字かバイナリ値かに関係なく使用できます。16 進のエスケープシーケンスは、バックスラッシュ、x に続けて 2 桁の 16 進数を指定します。16 進数値は、CHAR と NCHAR 両方の文字列リテラルで CHAR 文字セットの文字として解釈されます。単一のタブ文字として値を格納しない場合、値 ¥x09 は ¥¥x09 としてコード化してください。ただし、¥xyy は ¥xyy として格納されます。コードページ 1252 の例は、数字 1、2、3 に続けてユーロ通貨記号を表現しています。`'123¥x80'`。
- 円記号は、円記号を追加して次のようにエスケープしてください。`'c:¥¥november'`。また、パスの場合は、円記号の代わりにスラッシュ (/) を使用することもできます。`'c:/november'`。
- 文字列内に改行を表すには、円記号と n (¥n) を使用して次のように指定します。`'First line:¥nSecond line:'`

NCHAR 文字列リテラルには、CHAR 文字列リテラルと同じ文字とエスケープシーケンスを使用できます。

文字列リテラルに直接入力できない Unicode 文字を使用する場合、UNISTR 関数を使用します。

関連情報

UNISTR 関数 [文字列] [579 ページ]

1.1.5 演算子

複数の算術演算子、文字列演算子、配列演算子、ビット処理演算子があります。

一般的な演算子の優先度が適用されます。カッコ内の式が最初に評価され、続いて乗算と除算、最後に加算と減算が評価されます。その後、文字列の連結が行われます。

このセクションの内容:

比較演算子 [15 ページ]
比較演算子を使用して、値を比較できます。

論理演算子 [16 ページ]
式は、論理演算子を使用して結合、否定、またはテストできます。

算術演算子 [17 ページ]
算術演算子を使用して、式に対して算術演算を実行できます。

文字列演算子 [18 ページ]
文字列演算子を使用すると、文字列を結合できます。

配列演算子 [25 ページ]
配列演算子を使用すると、配列を結合できます。

ビット処理演算子 [31 ページ]
bit データ型、integer データ型 (bit、tinyint、SMALLINT などの変形をすべて含む)、バイナリ値、およびビット配列データ型では、複数の演算子を使用できます。

ジョイン演算子 [31 ページ]
SQL Anywhere では、Transact-SQL の外部ジョイン演算子である *= および =* の 2 つの比較演算子が追加でサポートされます。

演算子の優先度 [32 ページ]
式における演算子の優先度は重要です。

関連情報

探索条件 [53 ページ]

1.1.5.1 比較演算子

比較演算子を使用して、値を比較できます。

比較の構文は、次のとおりです。

```
expression comparison-operator expression
```

ここで comparison-operator は次のいずれかです。

演算子	説明
=	等しい
>	より大きい
<	より小さい
>=	以上
<=	以下
!=	等しくない
<>	等しくない
!>	より大きくない
!<	より小さくない

大文字と小文字の区別

デフォルトでは、データベースは大文字と小文字を区別しないで作成されます。比較処理は、該当のデータベースの文字の区別に合わせて実行されます。データベースでの大文字と小文字の区別は、データベースの作成時に -c オプションを使用して制御できます。

大文字と小文字の区別は、データベース作成時に設定されます。

> **注記**
>
> 文字列の比較では、大文字と小文字を区別するようにデータベースを作成しないかぎり、大文字と小文字を区別しません

後続ブランク

文字列を比較するときの SQL Anywhere の動作は、データベース作成時に設定されます。

1.1.5.2 論理演算子

式は、論理演算子を使用して結合、否定、またはテストできます。

たとえば、探索条件は、AND 演算子または OR 演算子で結合できます。また、NOT 演算子を使用して探索条件を否定したり、IS 演算子を使用して式が true、false、または unknown と評価されるかをテストしたりすることもできます。

AND 演算子

AND 演算子は、次のように探索条件の間に配置されます。

```
...WHERE condition1 AND condition2
```

AND を使用すると、両方の条件が TRUE の場合、結合した条件は TRUE になります。条件のいずれかが FALSE の場合は FALSE、それ以外の場合は UNKNOWN になります。

OR 演算子

OR 演算子は、次のように探索条件の間に配置されます。

```
...WHERE condition1 OR condition2
```

OR を使用すると、条件のいずれかが TRUE の場合、結合した条件は TRUE になります。両方の条件が FALSE の場合は FALSE、それ以外の場合は UNKNOWN になります。

NOT 演算子

NOT 演算子は、次のように条件の前に配置され、条件を否定します。

```
...WHERE NOT condition
```

`condition` が FALSE の場合、NOT 条件は TRUE です。`condition` が TRUE の場合は FALSE、`condition` が UNKNOWN の場合は UNKNOWN になります。

IS 演算子

IS 演算子は、式とテストする真理値の間に配置されます。IS 演算子の構文は、次のとおりです。

```
expression IS [ NOT ] truth-value
```

`expression` が指定の `truth-value` (TRUE、FALSE、UNKNOWN、NULL のいずれか) と評価されれば IS 条件は TRUE になります。それ以外の場合、値は FALSE です。

たとえば、`5*3=15 IS TRUE` は、式 `5*3=15` が TRUE と評価されるかどうかをテストします。

関連情報

3 値的論理 [80 ページ]

1.1.5.3 算術演算子

算術演算子を使用して、式に対して算術演算を実行できます。

ソフトウェアでは、次の算術演算子がサポートされています。

expression + expression

加算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

expression - expression

減算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

-expression

反転。式が NULL 値の場合、結果は NULL 値になります。

expression * expression

乗算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

expression / expression

除算。いずれかの式が NULL の場合、または 2 番目の式が 0 の場合、結果は NULL になります。

expression % expression

モジュロによる、2 つの整数での除算の余り (整数) の算出。たとえば、21 を 11 で割ると商は 1、余りは 10 なので、21 % 11 = 10 になります。

日付および時刻データ型に対する算術演算子のサポート

日付および時刻データ型に対する算術演算子のサポートは、+ および - に制限されています。これらの演算子の使用方法にはいくつかの制約があります。

`expression1 + expression2` の演算の場合、一方の式が日付であり、もう一方の式が時刻である必要があります。この結果、日付と時刻を組み合わせた形式の TIMESTAMP が生成されます。

`expression1 - expression2` の演算の場合、制限および動作は次のとおりです。

expression1 は **TIMESTAMP** のサブタイプ、**expression2** は **SIGNED LONG** のサブタイプ

この結果、`expression2` 日間が引かれた TIMESTAMP が生成されます。

expression1 は **TIMESTAMP** のサブタイプ、**expression2** も同様

この結果、`expression1` から `expression2` までの日数が生成されます。これは、`( DATEDIFF ( day, expression1,expression2 ) )` として表すこともできます。

expression1 は **TIMESTAMP** のサブタイプ、**expression2** は **NUMERIC** または **VARCHAR** のサブタイプ

`expression2` は最初に NUMERIC に変換されてから、日数として解釈されます。この結果、`expression1` から `expression2` までの日数が生成されます。これは、`( DATEDIFF ( day, expression1 , expression2 ) )` として表すこともできます。

標準

ANSI/ISO SQL 標準

モジュロ演算子としての % の使用は標準にありません。

1.1.5.4 文字列演算子

文字列演算子を使用すると、文字列を結合できます。

ソフトウェアでは、次の文字列演算子がサポートされています。

expression || expression

文字列連結 (2 つのパイプ記号)。いずれかの文字列が NULL 値の場合、連結には空文字列として扱われます。

expression + expression

代替の文字列連結。+ 連結演算子を使用する場合は、暗黙的データ変換を行わないで、必ずオペランドを文字データ型に明示設定してください。

たとえば、次のクエリは整数値 579 を返します。

```
SELECT 123 + 456;
```

これに対し、次のクエリは文字列 123456 を返します。

```
SELECT '123' + '456';
```

標準

ANSI/ISO SQL 標準

|| 演算子は、ANSI/ISO SQL 標準の文字列連結演算子です。ただし、SQL 標準では、|| のいずれかのオペランドが NULL 値の場合、連結の結果も NULL となります。ソフトウェアでは、|| オペランドは NULL を空の文字列として扱います。

このセクションの内容:

OPENXML 演算子 [19 ページ]
XML ドキュメントから結果セットを生成します。

1.1.5.4.1 OPENXML 演算子

XML ドキュメントから結果セットを生成します。

構文 - XML の指定

```
OPENXML(
xml-data
, xpath
[, flags
[, namespaces ] ]
)
WITH ( column-name column-type
[ xpath ] [ , ... ]
)
```

構文 - XML が含まれるファイルの指定

```
OPENXML( { USING FILE | USING VALUE }
xml-data
, xpath
[, flags
[, namespaces ] ]
)
WITH ( column-name column-type
[ xpath ] [ , ... ]
)
[ OPTION ( scan-option ) ]
[ AS ] correlation-name
```

```
scan-option :
ENCODING encoding
| BYTE ORDER MARK { ON | OFF }
```

パラメータ

WITH 句

結果セットのスキーマと、結果セットの各カラムに対して値を見つける方法を指定します。WITH 句の `xpath` 引数は、2 番目の引数の `xpath` に対する一致と相対的に一致します。WITH 句の式が複数のノードと一致した場合、ドキュメントの順序における最初のノードだけが使用されます。ノードがテキストノードではない場合、テキストノードの下位ノードをすべて追加することにより結果を検索します。WITH 句の式がどのノードにも一致しない場合は、そのローのカラムは NULL になります。

WITH 句内の `xpath` 引数には、リテラル文字列または変数を指定できます。http://www.w3.org/TR/xpath を参照してください。

OPENXML の WITH 句構文は、ストアドプロシージャから選択する際に使用する構文と似ています。

USING FILE | USING VALUE

USING FILE 句は、ファイルからデータをロードするときに使用します。

USING VALUE 句は、CHAR 型、NCHAR 型、BINARY 型、または LONG BINARY 型の式や、BLOB 文字列からデータをロードするときに使用します。

xml-data

結果セットのベースとなる XML。定数、変数、カラムなど、任意の文字列式が使用できます。

出力に NCHAR カラムが存在する場合、`xml-data` は NCHAR エンコードで直接解析されます。また、xpath 引数と namespaces 引数も NCHAR エンコードで変換され、解析されます。

xpath

XPath クエリを含む文字列。XPath を使用すると、問い合わせる XML ドキュメントの構造を記述するパターンを指定できます。この引数にある XPath パターンによって、XML ドキュメントからノードが選択されます。2 番目の `xpath` 引数の XPath クエリに一致する各ノードが、テーブルにローを 1 つずつ生成します。

WITH 句の `xpath` 引数に指定できるのは、メタプロパティのみです。メタプロパティは、属性のように XPath クエリ内でアクセスされます。`namespaces` が指定されていない場合、デフォルトでプレフィクス mp が Uniform Resource Identifier (URI) urn:sap-com:sa-xpath-metaprop にバインドされます。`namespaces` が指定された場合、この URI は、クエリのメタプロパティにアクセスするために mp または他のプレフィクスにバインドされます。メタプロパティ名の大文字と小文字は区別されます。OPENXML 文は、次のメタプロパティをサポートします。

@mp:id

XML ドキュメント内のユニークなノードの ID を返します。データベースサーバが再起動されると、ドキュメント内の指定されたノードの ID が変更される場合もあります。このメタプロパティの値は、ドキュメントの順序で増えていきます。

@mp:localname

ノードの名前のローカル部分を返します。ノードに名前がない場合は NULL を返します。

@mp:prefix

ノードの名前のプレフィクス部分を返します。ノードに名前がない場合または名前にプレフィクスが付いていない場合は NULL を返します。

@mp:namespaceuri

ノードが含まれているネームスペースの URI を返します。ノードがネームスペースに含まれていない場合は NULL を返します。

@mp:xmltext

XML ドキュメントのサブツリーを XML 形式で返します。たとえば、内部ノードを照合する場合、このメタプロパティを使用して、下位のテキストノードの連結値ではなく、XML 文字列を返します。

flags

WITH 句に XPath クエリが指定されていない場合、XML データと結果セットの間で使用されるマッピングを示します。`flags` パラメータが指定されない場合、デフォルトで結果セットのカラムに属性がマッピングされます。`flags` パラメータには、次のいずれかの値を指定します。

値	説明
1	XML 属性が結果セットのカラムにマッピングされます (デフォルト)。
2	XML 要素が結果セットのカラムにマッピングされます。

namespace-declaration

XML ドキュメント。クエリのスコープ内のネームスペースは、ドキュメントのルート要素から取得されます。ネームスペースが指定されている場合は、すべての `xpath` 引数が指定されていても、必ず `flags` 引数を含めます。

column-name

結果セットのカラムの名前。

column-type

結果セットのカラムのデータ型。データ型は、XML ドキュメントから選択した値と互換性がなければなりません。

OPTION 句

OPTION 句は、エスケープ文字、デリミタ、エンコードなど、入力ファイルに使用する解析オプションを指定するときに使用します。

ENCODING 句

ENCODING 句では、ファイルの読み込みに使用されるエンコードを指定できます。

ENCODING 句を指定しない場合、値が CHAR 型または BINARY 型のときはデータベース文字セット (db_charset) が、値が NCHAR 型のときは NCHAR データベース文字セット (nchar_charset) が値のエンコードと見なされます。

BYTE ORDER MARK 句

BYTE ORDER MARK 句は、エンコード内にバイト順マーク (BOM) があるかどうかを制御するときに指定します。デフォルトでは、このオプションは ON です。この場合、サーバはデータの先頭でバイトオーダマーク (BOM) を検索して解釈できます。BYTE ORDER MARK が OFF の場合、サーバは BOM を検索しません。

入力データがエンコードされている場合は、BYTE ORDER MARK 句を指定してください。

ENCODING 句が指定されている場合

- BYTE ORDER MARK オプションが ON のときに、UTF-16BE または UTF-16LE などのエンディアンを持つ UTF-16 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの検証に使用されます。間違ったエンディアンを指定すると、エラーが返されます。
- BYTE ORDER MARK オプションが ON のときに、明示的なエンディアンのない UTF-16 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの確認に使用されます。BOM がない場合は、オペレーティングシステムのエンディアンであると想定されます。
- BYTE ORDER MARK オプションが ON のときに UTF-8 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は無視されます。

ENCODING 句が指定されていない場合

- ENCODING 句を指定しないで BYTE ORDER MARK オプションを ON にすると、サーバは入力データの先頭で BOM を検索します。BOM が見つかると、BOM のエンコード (UTF-16BE、UTF-16LE、または UTF-8) に基づいてソースエンコードが自動的に選択され、BOM はロードするデータの一部であるとは見なされなくなります。

- ENCODING 句を指定せず、BYTE ORDER MARK オプションが OFF であるか、または BOM が入力データの先頭で見つからない場合は、データベースの CHAR エンコードが使用されます。

備考

OPENXML 演算子は、`xml-data` を解析し、結果をツリーとして雛型化します。ツリーには、要素、属性、テキストノード、その他の XML 構成要素ごとに個別のノードがあります。OPENXML 演算子に指定される XPath クエリは、ツリーからノードを選択する際に使用され、選択されたノードはその後、結果セットにマッピングされます。

OPENXML 演算子で使用される XML パーサは妥当性が検証されず、外部 DTD サブセットまたは外部パラメータのエンティティを読み取りません。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

カラム式に複数の一致がある場合、ドキュメントの順序 (解析される前の元の XML ドキュメントの順序) における最初の一致が使用されます。一致するノードがない場合は、NULL が返されます。内部ノードが選択される場合、結果は連結された内部ノードの下位のテキストノードすべてになります。

データ型が BINARY、LONG BINARY、IMAGE、VARBINARY のカラムは、base64 エンコード形式とみなされ、自動的に復号化されます。FOR XML 句を使用して XML を生成した場合、これらのタイプは base64 エンコードであり、OPENXML 演算子を使用して復号化できます。

OPENXML 演算子は、XPath 構文のサブセットを次のようにサポートしています。

- 子、自分、属性、子孫、子孫と自分、親の各軸は、完全にサポートされています。
- 省略形または省略形ではない構文のどちらも、サポートされるすべての機能に使用できます。たとえば、`'a'` は `'child::a'` に等しく、`'..'` は `'parent::node()'` と同じです。
- 名前のテストにワイルドカードが使用できます。たとえば、`'a/*/b'` のように記述します。
- サポートされるテストの種類は、node()、text()、processing-instruction()、comment() です。
- フォーム `expr1[expr2]` および `expr1[expr2="string"]` の修飾子を使用できます。`expr2` は、サポートされている XPath 式です。`expr2` が 1 つ以上のノードと一致する場合、修飾子は TRUE と評価されます。たとえば、`'a[b]'` は、少なくとも 1 つの子 `b` を持つ `a` ノードを検索します。`a[b="I"]` は、テキスト値 `I` を持つ子 `b` を少なくとも 1 つ持つ `a` ノードを検索します。

権限

USING FILE 句を指定する場合は、READ FILE システム権限が必要です。それ以外の場合、権限は必要ありません。

例

次のクエリは、OPENXML 演算子への最初の引数として指定された XML ドキュメントから結果セットを生成します。

```
SELECT * FROM OPENXML( '<products>
                <ProductType ID="301">Tee Shirt</ProductType>
                <ProductType ID="401">Baseball Cap</ProductType>
                </products>',
                '/products/ProductType' )
```

```
WITH ( ProductName LONG VARCHAR 'text()', ProductID CHAR(3) '@ID');
```

このクエリは、次の結果を生成します。

ProductName	ProductID
Tee Shirt	301
Baseball Cap	401

次の例では、最初の <ProductType> 要素にエンティティがあります。クエリを実行すると、このノードは、次の 4 つの子がある要素として解析されます: Tee、&、Sweater、Set。結果セット内で子を連結するには、ピリオド (.) を使用します。

```
SELECT * FROM OPENXML( '<products>
                 <ProductType ID="301">Tee &amp; Sweater Set</ProductType>
                 <ProductType ID="401">Baseball Cap</ProductType>
                 </products>',
                 '/products/ProductType' )
WITH ( ProductName LONG VARCHAR '.', ProductID CHAR(3) '@ID');
```

このクエリは、次の結果を生成します。

ProductName	ProductID
Tee & Sweater Set	301
Baseball Cap	401

次のクエリは、等号述部を使用して、指定された XML ドキュメントから結果セットを生成します。

```
SELECT * FROM OPENXML('<EmployeeDirectory>
   <Employee>
      <column name="EmployeeID">105</column>
      <column name="GivenName">Matthew</column>
      <column name="Surname">Cobb</column>
      <column name="Street">7 Pleasant Street</column>
      <column name="City">Grimsby</column>
      <column name="State">UT</column>
      <column name="PostalCode">02154</column>
      <column name="Phone">6175553840</column>
   </Employee>
  <Employee>
      <column name="EmployeeID">148</column>
      <column name="GivenName">Julie</column>
      <column name="Surname">Jordan</column>
      <column name="Street">1244 Great Plain Avenue</column>
      <column name="City">Woodbridge</column>
      <column name="State">AZ</column>
      <column name="PostalCode">01890</column>
      <column name="Phone">6175557835</column>
   </Employee>
  <Employee>
      <column name="EmployeeID">160</column>
      <column name="GivenName">Robert</column>
      <column name="Surname">Breault</column>
      <column name="Street">358 Cherry Street</column>
      <column name="City">Milton</column>
      <column name="State">PA</column>
      <column name="PostalCode">02186</column>
      <column name="Phone">6175553099</column>
   </Employee>
  <Employee>
      <column name="EmployeeID">243</column>
```

```
      <column name="GivenName">Natasha</column>
      <column name="Surname">Shishov</column>
      <column name="Street">151 Milk Street</column>
      <column name="City">Grimsby</column>
      <column name="State">UT</column>
      <column name="PostalCode">02154</column>
      <column name="Phone">6175552755</column>
   </Employee>
</EmployeeDirectory>', '/EmployeeDirectory/Employee')
WITH ( EmployeeID INT 'column[@name="EmployeeID"]',
       GivenName    CHAR(20) 'column[@name="GivenName"]',
       Surname      CHAR(20) 'column[@name="Surname"]',
       PhoneNumber  CHAR(10) 'column[@name="Phone"]');
```

このクエリは、次の結果セットを生成します。

EmployeeID	GivenName	Surname	PhoneNumber
105	Matthew	Cobb	6175553840
148	Julie	Jordan	6175557835
160	Robert	Breault	6175553099
243	Natasha	Shishov	6175552755

次のクエリは、XPath @attribute 式を使用して結果セットを生成します。

```
SELECT * FROM OPENXML( '<Employee
      EmployeeID="105"
      GivenName="Matthew"
      Surname="Cobb"
      Street="7 Pleasant Street"
      City="Grimsby"
      State="UT"
      PostalCode="02154"
      Phone="6175553840"
      />', '/Employee' )
WITH ( EmployeeID INT '@EmployeeID',
       GivenName    CHAR(20) '@GivenName',
       Surname      CHAR(20) '@Surname',
       PhoneNumber  CHAR(10) '@Phone');
```

次のクエリは、上記のクエリで使用されているような XML ドキュメント (ただし、XML 名前空間は使用されています) を操作します。XPath クエリに名前のテストにワイルドカードを使用し、上記のクリックと同じ結果セットを生成する例です。

```
SELECT * FROM OPENXML( '<Employee  xmlns="http://www.sap.com/EmployeeDemo"
      EmployeeID="105"
      GivenName="Matthew"
      Surname="Cobb"
      Street="7 Pleasant Street"
      City="Grimsby"
      State="UT"
      PostalCode="02154"
      Phone="6175553840"
/>', '/*:Employee' )
WITH ( EmployeeID INT '@EmployeeID',
       GivenName    CHAR(20) '@GivenName',
       Surname      CHAR(20) '@Surname',
       PhoneNumber  CHAR(10) '@Phone');
```

または、名前空間の宣言を指定することもできます。

```
SELECT * FROM OPENXML( '<Employee  xmlns="http://www.sap.com/EmployeeDemo"
```

```
      EmployeeID="105"
      GivenName="Matthew"
      Surname="Cobb"
      Street="7 Pleasant Street"
      City="Grimsby"
      State="UT"
      PostalCode="02154"
      Phone="6175553840"
/>', '/prefix:Employee', 1, '<r xmlns:prefix="http://www.sap.com/EmployeeDemo"/
>' )
WITH ( EmployeeID INT '@EmployeeID',
       GivenName    CHAR(20) '@GivenName',
       Surname      CHAR(20) '@Surname',
       PhoneNumber  CHAR(10) '@Phone');
```

関連情報

SQL データ型 [124 ページ]
FROM 句 [1112 ページ]

1.1.5.5 配列演算子

配列演算子を使用すると、配列を結合できます。

ソフトウェアでは、次の配列演算子がサポートされています。

expression || expression

配列連結 (2 つのパイプ記号)。いずれかの配列が NULL 値の場合、連結には長さ 0 の配列として扱われます。

標準

ANSI/ISO SQL 標準

|| 演算子は、ANSI/ISO SQL 標準の連結演算子です。ただし、SQL 標準では、|| のいずれかのオペランドが NULL 値の場合、連結の結果も NULL となります。ソフトウェアでは、|| 演算子は NULL を長さ 0 の配列として扱います。

このセクションの内容:

UNNEST 配列演算子 [26 ページ]
指定された配列式から、配列要素ごとに 1 つのローで構成される派生テーブルを作成します。

1.1.5.5.1 UNNEST 配列演算子

指定された配列式から、配列要素ごとに 1 つのローで構成される派生テーブルを作成します。

構文

```
UNNEST ( array-expression [, ...] )
[ WITH ORDINALITY ]
```

パラメータ

array-expression

テーブルカラムの派生元の配列。

WITH ORDINALITY

WITH ORDINALITY 句を使用して、アプリケーションに各値の取得元の配列要素を再呼び出しさせることができます。有効な UNNEST 派生テーブルでは、結果となる式ごとに (AS 句を使用して) 指定された名前が付いている必要があります。UNNEST によって生成されるローの順序は保証されません。ユーザは ORDER BY 句を使用して目的の順序にすることができます。

備考

配列式ごとにカーディナリティが異なる場合、短い配列から失われた出力式は NULL に設定されます。WITH ORDINALITY 句が指定されると、結果セットには、ローが表す配列要素のカーディナル番号を識別する整数カラムが含まれます。新しいカラムは UNNEST 派生テーブルに最後のカラムとして追加されます。

権限

なし

例

次の例は、カーディナリティが異なる 2 つの配列を使用した UNNEST 演算子の使い方を示しています。

```
SELECT * FROM UNNEST( ARRAY(2,3,4), ARRAY(4,5,6) ) WITH ORDINALITY AS DT(X,Y,Z);
```

この SQL 文は次の結果を返します。

X	Y	Z
2	4	1

X	Y	Z
3	5	2
4	6	3

例

次の文は、簡単な配列を作成し、この配列にデータを移植します。

```
CREATE OR REPLACE VARIABLE x1 ARRAY (10) OF INT;
SELECT ARRAY_AGG(id) INTO x1 FROM GROUPO.Products;
```

次の文は、ネスト解除された配列内のデータを返します。

```
SELECT * FROM UNNEST(x1) AS DT(X);
```

X
300
301
302
400
401
500
501
600
601
700

次の文は、同じデータを返しますが、ordinality (Y カラム) を追加します。

```
SELECT * FROM UNNEST(x1) WITH ORDINALITY AS DT(X,Y);
```

X	Y
300	1
301	2

X	Y
302	3
400	4
401	5
500	6
501	7
600	8
601	9
700	10

次の文は、配列のセル 1 からデータを返します。

```
SELECT x1[[1]];
```

x1[[1]]
300

次の文は、2 次元配列を作成し、この配列にデータを移植します。

```
CREATE OR REPLACE VARIABLE x1 ARRAY(2) OF ARRAY (10) OF INT;
SELECT ARRAY_AGG( "id" ) INTO x1[[1]] FROM GROUPO.Products;
SELECT ARRAY_AGG( GROUPO.Products.Quantity ) INTO x1[[2]] FROM GROUPO.Products;
```

次の文は、ネスト解除された配列内のデータを返します。

```
SELECT * FROM UNNEST( x1[[1]], x1[[2]] ) WITH ORDINALITY AS DT( X, Y, Z);
```

X	Y	Z
300	28	1
301	54	2
302	75	3
400	112	4
401	12	5
500	36	6
501	28	7

X	Y	Z
600	39	8
601	32	9
700	80	10

次の文は、配列の最初のローの 2 番目のカラムで見つかったデータを返します。

```
SELECT (x1[[2]])[[1]];
```

(x1[[2]])[[1]]
28

次の文は、配列とローを作成し、これらにデータを移植します。

```
CREATE OR REPLACE VARIABLE x1 ARRAY (10) OF INT;
CREATE OR REPLACE VARIABLE x2 ROW( a1 INT, b1 ARRAY(10) OF INT );
SELECT ARRAY_AGG( "id" ) INTO x1 FROM GROUPO.Products;
SET x2.a1 = 10;
SET x2.b1 = x1;
```

次の文は、配列とロー内のデータをまとめて返します。

```
SELECT x2.a1 AS a1, X, Z FROM UNNEST( x2.b1 ) WITH ORDINALITY AS DT(X,Z);
```

a1	X	Z
10	300	1
10	301	2
10	302	3
10	400	4
10	401	5
10	500	6
10	501	7
10	600	8
10	601	9
10	700	10

次の文は、配列とロー内の最初の値を返します。

```
SELECT x2.a1, x2.b1[[1]];
```

a1	x2.b1[[1]]
10	300

次の文は、ローの配列を作成し、これらにデータを移植します。

```
CREATE OR REPLACE VARIABLE x4 ARRAY(10) OF ROW( a1 INT, b1 INT, c1 VARCHAR(120) );
SELECT ARRAY_AGG( ROW( ID, Quantity, Name ) ) INTO x4 FROM GROUPO.Products;
```

次の文は、ネスト解除されたローを配列から返します。

```
SELECT (x).a1 FROM UNNEST( x4 ) AS dt(x);
```

式
300
301
302
400
401
500
501
600
601
700

関連情報

複合データ型 ROW および ARRAY [178 ページ]
複合型の比較 [190 ページ]
ARRAY コンストラクタ [複合] [234 ページ]
FROM 句 [1112 ページ]

1.1.5.6 ビット処理演算子

bit データ型、integer データ型 (bit、tinyint、SMALLINT などの変形をすべて含む)、バイナリ値、およびビット配列データ型では、複数の演算子を使用できます。

演算子	説明
&	ビット処理 AND
|	ビット処理 OR
^	ビット処理の排他 OR
~	ビット処理 NOT

ビット処理演算子 &、|、~ は、論理演算子 AND、OR、NOT で代用することはできません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文では、正しいビットが設定されているローが選択されます。たとえば、Options の値が `0x1001` である場合は、そのローが含まれます。

```
SELECT *
FROM tableA
WHERE ( Options & 0x0101 ) <> 0;
```

1.1.5.7 ジョイン演算子

SQL Anywhere では、Transact-SQL の外部ジョイン演算子である *= および =* の 2 つの比較演算子が追加でサポートされます。

これらの演算子のいずれかが比較述部に使用されると、暗黙的な LEFT OUTER JOIN または RIGHT OUTER JOIN が指定されます。

注記

Transact-SQL 外部ジョイン演算子 *= と =* はサポートされなくなりました。Transact-SQL の外部ジョインを使用するには、tsql_outer_joins データベースオプションを On に設定します。

1.1.5.8 演算子の優先度

式における演算子の優先度は重要です。

次のリストの最上部にある演算子から順に評価されます。

1. 単項演算子 (1 つのオペランドを必要とする演算子)
2. &、|、^、~
3. *、/、%
4. +、-
5. ||
6. **not**
7. **and**
8. **or**

1 つの式に複数の演算子を使用する場合、かっこを使用して演算子の順序を明示的に指定してください。

1.1.6 SQL 文の式

式は、評価して値を返すことのできる文です。

構文

```
expression:
 case-expression
| constant
| [correlation-name.]column-name
| - expression
| expression operator expression
| ( expression )
| function-name ( expression, ... )
| if-expression
| special value
| ( subquery )
| variable-name
| sequence-expression
```

```
case-expression :
CASE expression
WHEN expression
THEN expression,...
[ ELSE expression ]
END
```

```
alternative form of case-expression :
CASE
WHEN search-condition
THEN expression, ...
[ ELSE expression ]
END
```

```
constant :
 integer | number | string | host-variable
```

```
special-value :
 CURRENT { DATE | TIME | TIMESTAMP }
| NULL
| SQLCODE
| SQLSTATE
| USER
```

```
if-expression :
IF condition
THEN expression
[ ELSE expression ]
ENDIF
```

```
sequence-expression :
sequence-name.[ CURRVAL | NEXTVAL ]
FROM table-name
```

```
java-ref:
 .field-name [ java-ref ]
| >> field-name [ java-ref ]
| .method-name ( [ expression,... ] ) [ java-ref ]
| >> method-name ( [ expression,...] ) [ java-ref ]
```

```
operator:
{  + |  - |  * |  / |  || |  % }
```

備考

式は、さまざまな場所で使用されます。

式は、数種類の要素で構成されます。これらについては、関数や変数に関する項で説明します。

式を評価するには、データベースに接続する必要があります。

関連する動作

なし。

このセクションの内容:

IF 式 [35 ページ]
IF 式は、条件が TRUE、FALSE、または UNKNOWN のいずれであるかをテストします。

CASE 式 [36 ページ]
CASE 式は条件付きの SQL 式を提供します。CASE 式は、式が使用できればどこでも使用できます。

正規表現の概要 [38 ページ]
正規表現とは、文字列内で検索するパターンを定義する一連の文字、ワイルドカード、または演算子のことです。

正規表現の構文 [39 ページ]
正規表現は、SIMILAR TO 検索条件、REGEXP 検索条件、および REGEXP_SUBSTR 関数でサポートされています。

正規表現の例 [49 ページ]
正規表現には、参照できる有用な例が多数用意されています。

式の互換性 [52 ページ]
SQL Anywhere では、アポストロフィで囲まれた文字列を定数式、二重引用符で囲まれた文字列を区切り識別子 (データベースオブジェクト用の名前) とする ANSI/ISO SQL 標準の表記規則を採用しています。

関連情報

特別値 [83 ページ]
SQL 関数 [198 ページ]
SQL 変数 [118 ページ]
探索条件 [53 ページ]
SQL データ型 [124 ページ]

1.1.6.1 式内の定数

定数とは、数値または文字列リテラルです。文字列定数は、アポストロフィ ('一重引用符') で囲まれています。文字列内にアポストロフィを表すには、アポストロフィを 2 つ続けて使用します。

1.1.6.2 式内のカラム名

カラム名は、オプションの相関名の後に続く識別子です。相関名は、通常はテーブル名です。

カラム名に英字、数字、アンダースコア以外の文字が使用されている場合は、二重引用符 ("") で囲んでください。次の例は、有効なカラム名です。

- Employees.Name
- address
- "date hired"

- "salary"."date paid"

関連情報

識別子 [6 ページ]
FROM 句 [1112 ページ]

1.1.6.3 式のサブクエリ

サブクエリとは、別の SELECT 文、INSERT 文、UPDATE 文、DELETE 文、または別のサブクエリにネストされた SELECT 文のことです。

サブクエリは、一致するローがない場合は NULL と評価されます。

SELECT 文は、カッコで囲み、1 つの SELECT リスト項目のみを含めます。式として使用すると、通常、サブクエリは 1 つの値だけを返します。

サブクエリは、カラム名を使用できれば、任意の位置で使用できます。たとえば、別の SELECT 文の SELECT リストでも、サブクエリを使用できます。

関連情報

探索条件内のサブクエリ [56 ページ]

1.1.6.4 IF 式

IF 式は、条件が TRUE、FALSE、または UNKNOWN のいずれであるかをテストします。

IF 式の構文、次のとおりです。

```
IF condition
THEN expression1
[ ELSE expression2 ]
{ ENDIF | END IF }
```

この式は次のように返します。

- condition が TRUE の場合、IF 式は expression1 を返します。
- condition が FALSE の場合、IF 式は expression2 を返します。
- condition が FALSE で expression2 がない場合、IF 式は NULL を返します。
- condition が UNKNOWN の場合、IF 式は NULL を返します。

expression1 は、condition が TRUE の場合にのみ評価されます。同様に、expression2 は、condition が FALSE の場合にのみ評価されます。expression1 と expression2 は任意の式であり、condition は有効な検索条件です。

> **注記**
>
> IF 式は、IF 文と同じではありません。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

IF 文 [1156 ページ]
探索条件 [53 ページ]
NULL 特別値 [99 ページ]

1.1.6.5 CASE 式

CASE 式は条件付きの SQL 式を提供します。CASE 式は、式が使用できればどこでも使用できます。

CASE 式の構文は、次のとおりです。

```
CASE expression-1
WHEN expression-2
THEN expression-3, ...
[ ELSE expression-4 ]
{ END | END CASE }
```

CASE 句に続く式が WHEN 句に続く式と等しい場合、THEN 文の後の式に復帰します。それ以外の場合、ELSE 文があればそれに続く式に復帰します。

ELSE 句が存在しなく、expression-1 が expression-2...expression-n のいずれの値とも一致しない場合、CASE 式は NULL を返します。

たとえば、下記のコードでは CASE 式が SELECT 文の 2 番目の句として使用されています。

```
SELECT ID,
   ( CASE Name
      WHEN 'Tee Shirt' THEN 'Shirt'
      WHEN 'Sweatshirt' THEN 'Shirt'
      WHEN 'Baseball Cap' THEN 'Hat'
      ELSE 'Unknown'
```

```
   END ) AS Type
FROM GROUPO.Products;
```

代替構文は、次のとおりです。

```
CASE
WHEN search-condition
THEN expression-1, ...
[ ELSE expression-2 ]
END [ CASE ]
```

WHEN 句の後の探索条件が満たされた場合、THEN 文に続く式に復帰します。それ以外の場合、ELSE 文があればそれに続く式に復帰します。

たとえば、次の文では、CASE 式が SELECT 文の 3 番目の句として使用され、探索条件と文字列を関連付けています。

```
SELECT ID, Name,
   ( CASE
      WHEN Name='Tee Shirt' THEN 'Sale'
      WHEN Quantity >= 50  THEN 'Big Sale'
      ELSE 'Regular price'
   END ) AS Type
FROM GROUPO.Products;
```

省略形 CASE 式の NULLIF 関数

NULLIF 関数は、CASE 句を省略形で記述する方法の 1 つです。NULLIF の構文は、次のとおりです。

```
NULLIF ( expression-1, expression-2 )
```

NULLIF は 2 つの式の値を比較します。1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。1 番目の式と 2 番目の式が異なる場合、NULLIF は 1 番目の式を返します。

> **注記**
>
> CASE 句の構文と CASE 式の構文を混同しないでください。

標準

ANSI/ISO SQL 標準

コア機能。この標準では、文によって参照される式が、実行中の任意の時点で評価されることを許可しています。ソフトウェアでは、式の評価は、コンパイル時に決定できる定数値を除き、各 WHEN 句が評価されるときに、構文の順序に従って実行されます。

CASE 式で END 以外に END CASE を使用することは標準にありません。この標準の定義では、END は CASE 式で使用し、END CASE は CASE 句で使用します。

関連情報

CASE 文 [760 ページ]

1.1.6.6 正規表現の概要

正規表現とは、文字列内で検索するパターンを定義する一連の文字、ワイルドカード、または演算子のことです。

正規表現は、SELECT 文の WHERE 句での REGEXP 探索条件、または SIMILAR TO 探索条件の一部として、および REGEXP_SUBSTR 関数の引数としてサポートされています。LIKE 探索条件では正規表現がサポートされていませんが、LIKE で指定できる一部のワイルドカードと演算子は、正規表現のワイルドカードと演算子に似ています。

次の SELECT 文は、正規表現 (`(K|C[^h])%`) を使用して、Contacts テーブルを検索し、K または C で始まる (ただし Ch は除く) 姓を持つ連絡先を返します。

```
SELECT Surname, GivenName
   FROM GROUPO.Contacts
   WHERE Surname SIMILAR TO '(K|C[^h])%';
```

正規表現では、追加構文を組み込んで、グループ化、量指定、アサーション、代替を指定できます。

グループ化

グループ化を使用すると、正規表現の一部をグループ化していくつかの追加一致基準に適用できます。たとえば、`'(abc){2}'` は abcabc に一致します。

グループ化を使用すると、式の各部が評価される順序の制御もできます。たとえば、`'ab(cdcd)'` では最初に cdcd の出現を検索し、次に、cdcd のインスタンスの前に ab があるかどうかを評価します。

量指定

量指定を使用すると、式の先行部分が出現する回数を制御できます。たとえば、疑問符 (?) は、その前の文字の 0 または 1 つのインスタンスに一致する量指定子です。このため、`'honou?r'` は honor と honour の両方に一致します。

アサーション

通常、パターンの検索ではそのパターンが返されます。アサーションを使用すると、返される表現にパターンに含めることなく、パターンの存在をテストできます。たとえば、`'SQL(?= Anywhere)'` は、SQL の後ろに 1 つのスペースと Anywhere が続く SQL にのみ一致します。

代替

代替を使用すると、前のパターンが検出されない場合に検索する代替パターンを指定できます。代替パターンは左から右の順に評価され、最初の一致で検索が停止します。たとえば、`'col(o|ou)r'` は、color のインスタンスを検索します。インスタンスが検出されない場合、代わりに colour が検索されます。

関連情報

正規表現の構文 [39 ページ]
LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]

探索条件 [53 ページ]
REGEXP 探索条件 [67 ページ]
SIMILAR TO 探索条件 [69 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]

1.1.6.7 正規表現の構文

正規表現は、SIMILAR TO 検索条件、REGEXP 検索条件、および REGEXP_SUBSTR 関数でサポートされています。

SIMILAR TO での正規表現の構文は ANSI/ISO SQL 標準に準拠しています。REGEXP と REGEXP_SUBSTR での正規表現の構文とサポートは Perl 5 に準拠しています。

正規表現は、REGEXP と SIMILAR TO では文字列の一致に使用されますが、REGEXP_SUBSTR では部分文字列の一致に使用されます。REGEXP と SIMILAR TO で部分文字列の一致を実現するには、一致させようとしているパターンの前後にワイルドカードを指定する方法があります。たとえば、`REGEXP '.*car.*'` は car、carwash、vicar に一致します。また、クエリを書き直すことにより REGEXP_SUBSTR 関数を使用することも可能です。

SIMILAR TO での正規表現を使用した一致では、大文字と小文字、およびアクセント記号が区別されます。REGEXP と REGEXP_SUBSTR は、データベースの大文字と小文字、およびアクセント記号の影響を受けません。

このセクションの内容:

正規表現: メタ文字 [39 ページ]
メタ文字とは、正規表現内で特別な意味を持つ記号または文字のことです。

正規表現:特殊部分文字クラス [42 ページ]
部分文字クラスは、より大きな文字クラス内に埋め込まれた特殊文字クラスです。

正規表現:サポートされているその他の構文表記規則 [44 ページ]
次の構文表記規則が REGEXP 探索条件と REGEXP_SUBSTR 関数でサポートされており、円記号がエスケープ文字と認識されます。これらの表記規則は、*SIMILAR TO* 検索式ではサポートされません。

正規表現: アサーション [47 ページ]
アサーションは、条件が true かどうかをテストし、文字列内の一致が開始される位置に影響します。アサーションは文字を返しません。最終的な一致にはアサーションのパターンは含まれません。

1.1.6.7.1 正規表現: メタ文字

メタ文字とは、正規表現内で特別な意味を持つ記号または文字のことです。

メタ文字の扱いは次の条件で決まります。

- 正規表現が使用されているのが、SIMILAR TO 探索条件または REGEXP 探索条件であるか、REGEXP_SUBSTR 関数であるか
- メタ文字が使用されているのが、正規表現の含まれている文字クラス内部かどうか

この先を読み進む前に、文字クラスの定義を理解しておく必要があります。文字クラスは角カッコで囲まれた文字のセットで、文字列中の文字との一致対象になります。たとえば、構文 `SIMILAR TO 'ab[1-9]'` の [1-9] は文字クラスで、1 ～ 9 の

いずれかの数字に一致します。正規表現でのメタ文字の扱いは、メタ文字が文字クラス内にあるかどうかで異なります。具体的には、文字クラス内に配置されたメタ文字は、そのほとんどがメタ文字としてではなく通常の文字として扱われます。

SIMILAR TO の場合 (のみ)、メタ文字 *、?、+、_、|、(、)、{ は、文字クラス内でエスケープする必要があります。

文字クラスにリテラルのマイナス記号 (-)、脱字記号 (^)、または右側の角カッコ (]) を含めるには、これらの文字をエスケープする必要があります。

サポートされている正規表現メタ文字を次に示します。ほとんどすべてのメタ文字は、SIMILAR TO、REGEXP、および REGEXP_SUBSTR で同じように扱われます。

文字	その他の情報
[]	左および右角カッコは、文字クラスの指定に使用されます。文字クラスとは、一致させる文字のセットです。 ハイフン (-) と脱字記号 (^) を除き、文字クラス内に指定されたメタ文字 (* など) や量指定子 ({m} など) は特別な意味を持たず、実際の文字として評価されます。 POSIX 文字クラスなどの部分文字クラスもサポートされています。
*	アスタリスクは、1 文字との 0 回以上の一致に使用できます。たとえば、`REGEXP '.*abc'` は、任意の文字列で始まって abc で終わる文字列と一致します。したがって、aabc、xyzabc、abc に一致しますが、bc や abcc には一致しません。
?	疑問符は、1 文字との 0 回または 1 回の一致に使用できます。たとえば、`'colou?r'` は color と colour に一致します。
+	プラス記号は、1 文字との 1 回以上の一致に使用できます。たとえば、`'bre+'` は bre と bree に一致しますが、br には一致しません。
-	文字クラス内でハイフンを使用すると、範囲を指定できます。たとえば、`REGEXP '[a-e]'` は a、b、c、d、e に一致します。
[%]	パーセント記号は、SIMILAR TO で、任意の長さの文字列との一致に使用できます。 パーセント記号は、REGEXP および REGEXP_SUBSTR ではメタ文字として認識されません。指定されると、パーセント記号 (%) に一致します。
_	アンダースコア文字は、SIMILAR TO で、1 文字との一致に使用できます。 アンダースコアは、REGEXP および REGEXP_SUBSTR ではメタ文字として認識されません。指定されると、アンダースコア (_) に一致します。
|	パイプ記号は、文字列の一致における代替パターンの指定に使用できます。パイプ記号で区切られたパターンを持つ文字列において、パイプ記号は OR と解釈され、一致は一番左のパターンから始まり、最初の一致で停止します。したがって、パターンは重要な順にリストしてください。代替パターンはいくつでも指定できます。

文字	その他の情報
( )	左および右丸カッコは、正規表現の一部のグループ化に使用されるメタ文字です。たとえば、(ab) * は、0 回以上の ab の繰り返しに一致します。数値式と同様、グループ化を使用すると、正規表現の各部が評価される順序を制御できます。
{ }	左および右中カッコは、量指定子の指定に使用されるメタ文字です。量指定子では、一致を構成するためにパターンが繰り返される回数を指定します。次に例を示します。 **{m}** ある 1 つの文字に正確に m 回一致します。たとえば、'519-[0-9]{3}-[0-9]{4}' は、市外局番 519 の電話番号に一致します (データが構文に定義されるフォーマットの場合)。 **{m,}** ある 1 つの文字に少なくとも m 回一致します。たとえば、'[0-9]{5,}' は、5 桁以上の数字の文字列に一致します。 **{m,n}** ある 1 つの文字に少なくとも m 回、ただし n 回以下、一致します。たとえば、SIMILAR TO '_{5,10}' は、5 ～ 10 文字の文字列に一致します。
¥	円記号は、メタ文字のエスケープ文字として使用されます。また、メタ文字ではない文字のエスケープにも使用できます。
^	REGEXP と REGEXP_SUBSTR では、文字クラス外にある脱字記号は文字列の先頭に一致します。たとえば、'^[hc]at' は、文字列の先頭にある hat と cat にのみ一致します。 文字クラス内で使用された場合は、次のような動作になります。 **REGEXP および REGEXP_SUBSTR** 脱字記号が文字クラスの先頭文字である場合は、その文字セットに含まれていない文字に一致します。たとえば、REGEXP '[^abc]' は、a、b、c を除く任意の 1 文字に一致します。 脱字記号が角カッコ内の先頭文字ではない場合は、脱字記号に一致します。たとえば、REGEXP_SUBSTR '[a-e^c]' は a、b、c、d、e、^ に一致します。 **SIMILAR TO** SIMILAR TO では、脱字記号は減算演算子として扱われます。たとえば、SIMILAR TO '[a-e^c]' は a、b、d、e に一致します。
$	REGEXP と REGEXP_SUBSTR で使用されると、文字列の末尾に一致します。たとえば、REGEXP 'cat$' は cat に一致し、catfish には一致しません。

文字	その他の情報
.	REGEXP と REGEXP_SUBSTR で使用されると、任意の 1 文字に一致します。たとえば、`REGEXP 'a.cd'` は、a で始まり cd で終わる 4 文字の文字列に一致します。 SIMILAR TO で使用されると、ピリオド (.) に一致します。
:	コロンは、サブ文字クラスを指定するために文字セット内で使用されます。たとえば、`'[[:alnum:]]'` のように記述します。

関連情報

LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]
正規表現の例 [49 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]
正規表現:特殊部分文字クラス [42 ページ]

1.1.6.7.2 正規表現:特殊部分文字クラス

部分文字クラスは、より大きな文字クラス内に埋め込まれた特殊文字クラスです。

一致する文字のセットを定義するカスタム文字クラスのほかに (たとえば、`[abxq4]` は一致文字のセットを a、b、x、q、4 に制限します)、SQL Anywhere では、大部分の POSIX 文字クラスなどの部分文字クラスをサポートしています。たとえば、`[[:alpha:]]` はすべての大文字と小文字のセットを表します。

REGEXP 探索条件と REGEXP_SUBSTR 関数では、次の表に示す構文の表記規則がすべてサポートされていますが、SIMILAR TO 検索式ではサポートされていません。SIMILAR TO でサポートされている表記規則は、"SIMILAR TO" の欄に Y と示されています。

REGEXP 内と REGEXP_SUBSTR 関数を使用する場合は、脱字記号を使用して部分文字クラスを否定できます。たとえば、`[[:^alpha:]]` は、アルファベット文字を除くすべての文字のセットに一致します。

部分文字クラス	その他の情報	SIMILAR TO
[:alpha:]	現在の照合ではアルファベットの大文字と小文字に一致します。たとえば、`'[0-9]{3}[[:alpha:]]{2}'` は、3 つの数字とそれに続く 2 つの文字に一致します。	Y
[:alnum:]	現在の照合では数字およびアルファベットの大文字と小文字に一致します。たとえば、`'[[:alnum:]]+'` は、1 つ以上の英数字の文字列に一致します。	Y

部分文字クラス	その他の情報	SIMILAR TO
[:digit:]	現在の照合では数字に一致します。たとえば、'[[:digit:]-]+' は、1 つ以上の数字またはダッシュの文字列に一致します。同様に、'[^[:digit:]-]+' は、数字またはダッシュ以外の 1 文字以上の文字列に一致します。	Y
[:lower:]	現在の照合ではアルファベットの小文字に一致します。たとえば、A は大文字なので、'[[:lower:]]' は A に一致しません。	Y
[:space:]	1 つのブランク (' ') に一致します。たとえば、次の文は、Contacts.City から 2 語で構成される名前の任意の都市を検索します。 `SELECT City FROM GROUPO.Contacts WHERE City REGEXP '.*[[:space:]].*';`	Y
[:upper:]	現在の照合ではアルファベットの大文字に一致します。たとえば、'[[:upper:]ab]' は a または b の大文字のどれかに一致します。	Y
[:whitespace:]	スペース、タブ、フォームフィード、キャリッジリターンなどのホワイトスペース文字に一致します。	Y
[:ascii:]	7 ビット ASCII 文字 (0 ～ 127 の序数) に一致します。	
[:blank:]	ブランクスペースまたは水平タブに一致します。 [[:blank:]] は [¥t] と同義です。	
[:cntrl:]	32 未満の序数または文字値 127 の ASCII 文字 (制御文字) に一致します。制御文字には、改行文字、フォームフィード、バックスペースなどが含まれます。	
[:graph:]	出力された文字に一致します。 [[:graph:]] は [[:alnum:] [:punct:]] と同義です。	

部分文字クラス	その他の情報	SIMILAR TO
[:print:]	出力された文字とスペースに一致します。 `[[:print:]]` は `[[:graph:][:whitespace:]]` と同義です。	
[:punct:]	次のいずれかに一致します:!"#$%&'()*+,-./:;<=>?@[¥]^_`{|}~。 `[:punct:]` の部分文字列クラスには、現在の照合で使用できる ASCII 以外の句読表記文字が含まれない場合があります。	
[:word:]	現在の照合ではアルファベット文字、数字、またはアンダースコア文字に一致します。 `[[:word:]]` は `[[:alnum:]_]` と同義です。	
[:xdigit:]	文字クラス [0-9A-Fa-f] に含まれる文字に一致します。	

関連情報

LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]
正規表現の例 [49 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]
正規表現: メタ文字 [39 ページ]

1.1.6.7.3 正規表現:サポートされているその他の構文表記規則

次の構文表記規則が REGEXP 探索条件と REGEXP_SUBSTR 関数でサポートされており、円記号がエスケープ文字と認識されます。これらの表記規則は、*SIMILAR TO* 検索式ではサポートされません。

正規表現の構文	名前と意味
¥0 xxx	値が ¥0xxx の文字に一致します。ここで、xxx は任意の 8 進数の並びであり、0 はゼロです。たとえば、¥0134 は円記号に一致します。
¥a	ベル文字に一致します。

正規表現の構文	名前と意味
¥A	文字セットの外側で使用し、文字列の先頭に一致します。 文字セットの外側で使用する ^ と同義です。
¥b	バックスペース文字に一致します。
¥B	円記号 (¥) に一致します。
¥c x	指定された制御文字に一致します。たとえば、ctrl-Z の場合は ¥cZ です。
¥d	現在の照合では数字に一致します。たとえば、次の文は、Contacts.Phone から 00 で終わるすべての電話番号を検索します。 `SELECT Surname, Surname, City, Phone FROM GROUPO.Contacts WHERE Phone REGEXP '¥¥d{8}00';` ¥d は、文字クラスの内側と外側の両方で使用可能で、`[[:digit:]]` と同義です。
¥D	数字以外のものに一致します。これは、¥d の反対です。 ¥D は、文字クラスの内側と外側の両方で使用可能で、`[^[:digit:]]` と同義です。 否定の省略形を角カッコの内側で使用する場合は、注意が必要です。`[¥D¥S]` と `[^¥d¥s]` は同じではありません。後者は、数字またはホワイトスペース以外のすべての文字に一致します。このため、後者は x に一致しますが 8 には一致しません。一方、前者は、数字以外の文字またはホワイトスペース以外の文字のいずれかに一致します。数字はホワイトスペースではなく、ホワイトスペースは数字ではないため、`[¥D¥S]` は、任意の文字、数字、ホワイトスペース、その他に一致します。
¥e	エスケープ文字に一致します。
¥E	¥Q によって開始されたメタ文字を非メタ文字として扱う処理を終了します。
¥f	フォームフィードに一致します。
¥n	改行に一致します。
¥Q	¥E の箇所までは、すべてのメタ文字を非メタ文字として扱います。たとえば、`¥Q[$¥E` は `¥[¥$` と同義です。
¥r	キャリッジリターンに一致します。

正規表現の構文	名前と意味

¥s	スペースまたはホワイトスペースとして扱われる文字に一致します。たとえば、次の文は、Products.ProductName から名前に 1 つ以上のスペースが含まれるすべての製品名を返します。
	`SELECT Name`
	`FROM GROUPO.Products`
	`WHERE Name REGEXP '.*¥¥s.*'`
	¥s は、文字クラスの内側と外側の両方で使用可能で、`[[:whitespace:]]` と同義です。

¥S	ホワイトスペース以外の文字に一致します。これは ¥s の反対であり、`[^[:whitespace:]]` と同義です。 ¥S は、文字クラスの内側と外側の両方で使用できます。 否定の省略形を角カッコの内側で使用する場合は、注意が必要です。`[¥D¥S]` と `[^¥d¥s]` は同じではありません。後者は、数字またはホワイトスペース以外のすべての文字に一致します。このため、後者は x に一致しますが 8 には一致しません。一方、前者は、数字以外の文字またはホワイトスペース以外の文字のいずれかに一致します。数字はホワイトスペースではなく、ホワイトスペースは数字ではないため、`[¥D¥S]` は、任意の文字、数字、ホワイトスペース、その他に一致します。
¥t	水平タブに一致します。
¥v	垂直タブに一致します。

¥w	現在の照合ではアルファベット文字、数字、またはアンダースコアに一致します。たとえば、次の文は、Contacts.Surname から 7 文字長の英数字で構成されるすべての姓を返します。
	`SELECT Surname`
	`FROM GROUPO.Contacts`
	`WHERE Surname REGEXP '¥¥w{7}';`
	¥w は、文字クラスの内側と外側の両方で使用できます。
	`[[:alnum:]_]` と同義です。

¥W	現在の照合ではアルファベット文字、数字、またはアンダースコア以外のものに一致します。これは ¥w の反対であり、`[^[:alnum:]_]` と同義です。 この正規表現は、文字クラスの内側と外側の両方で使用できます。
¥x `hh`	値が 0x`hh` の文字に一致します。ここで、`hh` は 2 桁以下の 16 進数です。たとえば、`¥x2D` はハイフンと同義です。 ¥x{`hh`} と同義です。

正規表現の構文	名前と意味
¥x{ hhh }	値が 0xhhh の文字に一致します。ここで、hhh は 2 桁以下の 16 進数です。
¥z	文字列の末尾の位置 (文字ではない) に一致します。 $ と同義です。
¥Z	文字列の末尾の位置 (文字ではない) に一致します。 $ と同義です。

関連情報

LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]
正規表現の例 [49 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]
正規表現: メタ文字 [39 ページ]
正規表現:特殊部分文字クラス [42 ページ]

1.1.6.7.4 正規表現: アサーション

アサーションは、条件が true かどうかをテストし、文字列内の一致が開始される位置に影響します。アサーションは文字を返しません。最終的な一致にはアサーションのパターンは含まれません。

これらのアサーションは、REGEXP 探索条件と REGEXP_SUBSTR 関数でサポートされています。これらの表記規則は、SIMILAR TO 検索式ではサポートされません。

REGEXP_SUBSTR で文字列を分割する場合は、先読みアサーションと後読みアサーションが便利です。たとえば、次の文を実行すると、Customers テーブルの Address カラムに含まれる街路名のリストを (街路番号を付けずに) 返すことができます。

```
SELECT REGEXP_SUBSTR( Street, '(?<=^¥¥S+¥¥s+).*$' )
FROM GROUPO.Customers;
```

次の例は、正規表現を使用してパスワードが特定のルールに準拠しているかどうかを検証します。次の文に類似したゼロ幅アサーションを使用できます。

```
IF password REGEXP '(?=.*[[:digit:]])(?=.*[[:alpha:]].*[[:alpha:]])[[:word:]]
{4,12}' THEN
   MESSAGE 'Password conforms' TO CLIENT;
ELSE
   MESSAGE 'Password does not conform' TO CLIENT;
END IF
```

次の条件を満たす場合、パスワードは有効です。

- password に 1 つ以上の数字が含まれている ([[:digit:]] を使用したゼロ幅の正のアサーション)
- password に 2 つ以上のアルファベット文字が含まれている ([[:alpha:]].*[[:alpha:]] を使用したゼロ幅の正のアサーション)
- password に英数字またはアンダースコア文字のみが含まれている ([[:word:]])
- password が 4 文字以上で 12 文字以下である ({4,12})

以下の表にはサポートされているアサーションがあります。

構文	意味
(?=pattern**)**	正の先読みゼロ幅アサーション 一致文字列に pattern を含めないで、文字列内の現在位置のすぐ後ろに pattern があるかどうかを調べます。'A(?=B)' は、後ろに B が続く A に一致します。B は一致に含まれません。 たとえば、SELECT REGEXP_SUBSTR('in new york city', 'new(?=¥¥syork)'); は、部分文字列 new の後ろに ' york' (スペースの後ろに york があります) が続く場合に、new を返します。
(?!pattern**)**	負の先読みゼロ幅アサーション 一致文字列に pattern を含めないで、文字列内の現在位置のすぐ後ろに pattern がないかどうかを調べます。したがって、'A(?!B)' は、後ろに B が続かない A に一致します。 たとえば、SELECT REGEXP_SUBSTR('new jersey', 'new(?!¥¥syork)'); は、部分文字列 new を返します。
(?<=pattern**)**	正の後読みゼロ幅アサーション 一致文字列に pattern を含めないで、文字列内の現在位置のすぐ前に pattern があるかどうかを調べます。したがって、'(?<=A)B' は、すぐ前に A がある B に一致します。A は一致に含まれません。 たとえば、SELECT REGEXP_SUBSTR('new york', '(?<=new¥¥s)york'); は、部分文字列 york を返します。
(?<!pattern**)**	負の後読みゼロ幅アサーション 一致文字列に pattern を含めないで、文字列内の現在位置のすぐ前に pattern がないかどうかを調べます。 たとえば、SELECT REGEXP_SUBSTR('about york', '(?<!new¥¥s)york'); は、部分文字列 york を返します。

構文	意味
(?>pattern**)**	所有ローカル部分式 pattern に一致する残りの文字列の最大プレフィクスにのみ一致します。 たとえば、'aa' REGEXP '(?>a*)a' では、(?>a*) は先行の a にのみ一致するのではなく、aa に一致 (つまり、条件の評価対象全体に一致) します。このため、'aa' REGEXP '(?>a*)a' は、false と評価されます。
(?:pattern**)**	非取得ブロック これは、機能的には単なる pattern と同じであり、互換性を保つために提供されています。 たとえば、'bb' REGEXP '(?:b*)b' では、(?:b*) は bb に一致 (つまり、条件の評価対象全体に一致) します。ただし、所有ローカル部分式とは異なり、bb の最後の b は解放され、全体の一致が続行されます (つまり、非取得ブロックの外側に指定されている b との一致が可能)。 同様に、'a(?:bc|b)c' は abcc と abc に一致します。abc との一致の場合は、bc の最後の c でバックトラックが実行され、グループの外側にある c が使用されて一致が成功します。
(?#text**)**	コメントに使用します。text の内容は無視されます。

関連情報

LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]
正規表現の例 [49 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]
正規表現: メタ文字 [39 ページ]
正規表現:特殊部分文字クラス [42 ページ]

1.1.6.8 正規表現の例

正規表現には、参照できる有用な例が多数用意されています。

これらの例はすべて REGEXP で機能します。一部は SIMILAR TO でも機能し、その場合は "例" の欄にその旨が示されています。結果は、使用する探索条件によって異なります。SIMILAR TO で機能する正規表現の場合は、大文字と小文字やアクセント記号を区別するかどうかによって、結果はさらに異なります。

これらの例をリテラル文字列 ('.+@.+¥¥..+' など) で使用する場合は、円記号を 2 つ重ねる必要があります。

例	一致のサンプル

クレジットカード番号 (REGEXP のみ):
ビザ:
`4[0-9]{3}¥s[0-9]{4}¥s[0-9]{4}¥s[0-9]{4}`
マスターカード:
`5[0-9]{3}¥s[0-9]{4}¥s[0-9]{4}¥s[0-9]{4}`
アメリカンエキスプレス:
`37[0-9]{2}¥s[0-9]{4}¥s[0-9]{4}¥s[0-9]{4}`
ディスカバー:
`6011¥s[0-9]{4}¥s[0-9]{4}¥s[0-9]{4}`
	一致するもの (ビザ): 4123 6453 2222 1746
	一致しないもの (ビザ):
	3124 5675 4400 4567、4123-6453-2222-1746
	同様に、マスターカードでは、5 から始まり、4 つの数字のサブセット間にスペースがある 16 個の数字のセットに一致します。アメリカンエキスプレスとディスカバーは同じですが、それぞれ 37 と 6011 から始まります。

日付 (REGEXP および SIMILAR TO): `([0-2][0-9]|30|31)/(0[1-9]|1[0-2])/[0-9]{4}`	**一致するもの**: 31/04/1999、15/12/4567 **一致しないもの**: 31/4/1999、31/4/99、1999/04/19、42/67/25456
Windows の絶対パス (REGEXP のみ): `([A-Za-z]:|¥¥¥¥)¥¥¥¥[[:alnum:][:whitespace:].!"#$%&''()+,.¥¥¥¥;=@¥[^_`{}~¥¥]-]*`	**一致するもの**: ¥¥server¥share¥file **一致しないもの**: ¥directory¥directory2、/directory2
電子メールアドレス (REGEXP のみ): `[[:word:]¥-.]+@[[:word:]¥-.]+¥.[[:alpha:]]{2,3}`	**一致するもの**: abc.123@def456.com、_123@abc.ca **一致しないもの**: abc@dummy、ab*cd@efg.hijkl
電子メールアドレス (REGEXP のみ): `.+@.+¥..+`	**一致するもの**: *@qrstuv@wxyz.12345.com、__1234^%@@abc.def.ghijkl **一致しないもの**: abc.123.*&ca、^%abcdefg123
HTML の 16 進カラーコード (REGEXP および SIMILAR TO): `[A-F0-9]{6}`	**一致するもの**: AB1234、CCCCCC、12AF3B **一致しないもの**: 123G45、12-44-CC
HTML の 16 進カラーコード (REGEXP のみ): `[A-F0-9]{2}¥s[A-F0-9]{2}¥s[A-F0-9]{2}`	**一致するもの**: AB 11 00、CC 12 D3 **一致しないもの**: SS AB CD、AA BB CC DD、1223AB

例	一致のサンプル
IP アドレス (REGEXP のみ): `((2(5[0-5]|[0-4][0-9])|1([0-9][0-9])|([1-9][0-9])|[0-9])¥.){3}(2(5[0-5]|[0-4][0-9])|1([0-9][0-9])|([1-9][0-9])|[0-9])`	**一致するもの**: 10.25.101.216 **一致しないもの**: 0.0.0, 256.89.457.02
Java コメント (REGEXP のみ): `/¥*.*¥*/|//[^¥n]*`	**一致するもの**: /* と */ の間にある Java コメント、または先頭に // が付く 1 行のコメント **一致しないもの**: a=1
通貨 (REGEXP のみ): `(¥+|-)?¥$[0-9]*¥.[0-9]{2}`	**一致するもの**: $1.00、-$97.65 **一致しないもの**: $1、1.00$、$-75.17
正の値、負の値、小数値 (REGEXP のみ): `(¥+|-)?[0-9]+(¥.[0-9]+)?`	**一致するもの**: +41、-412、2、7968412、41、+41.1、-3.141592653 **一致しないもの**: ++41、41.1.19、-+97.14
パスワード (REGEXP および SIMILAR TO): `[[:alnum:]]{4,10}`	**一致するもの**: abcd、1234、A1b2C3d4、1a2B3 **一致しないもの**: abc、*ab12、abcdefghijkl
パスワード (REGEXP のみ): `[a-zA-Z]¥w{3,7}`	**一致するもの**: AB_cd、A1_b2c3、a123_ **一致しないもの**: *&^g、abc、1bcd
電話番号 (REGEXP および SIMILAR TO): `([2-9][0-9]{2}-[2-9][0-9]{2}-[0-9]{4})|([2-9][0-9]{2}¥s[2-9][0-9]{2}¥s[0-9]{4})`	**一致するもの**: 519-883-6898、519 888 6898 **一致しないもの**: 888 6898、5198886898、519 883-6898
文章 (REGEXP のみ): `[A-Z0-9].*(¥.|¥?|!)`	**一致するもの**:Hello, how are you? **一致しないもの**: i am fine
文章 (REGEXP のみ): `[[:upper:]0-9].*[.?!]`	**一致するもの**:Hello, how are you? **一致しないもの**: i am fine
社会保障番号 (REGEXP および SIMILAR TO): `[0-9]{3}-[0-9]{2}-[0-9]{4}`	**一致するもの**: 123-45-6789 **一致しないもの**: 123 45 6789、123456789、1234-56-7891

例	一致のサンプル
URL (REGEXP のみ): `(http://)?www¥.[a-zA-Z0-9]+¥.[a-zA-Z]{2,3}`	**一致するもの**: http://www.sample.com、www.sample.com **一致しないもの**: http://sample.com、http://www.sample.comm

関連情報

正規表現の構文 [39 ページ]
LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]

1.1.6.9 式の互換性

SQL Anywhere では、アポストロフィで囲まれた文字列を定数式、二重引用符で囲まれた文字列を区切り識別子 (データベースオブジェクト用の名前) とする ANSI/ISO SQL 標準の表記規則を採用しています。

このセクションの内容:

quoted_identifier オプション [52 ページ]
quoted_identifier オプションを使用して、区切り文字列の解釈を制御します。デフォルトでは、quoted_identifier オプションは On に設定されています。

1.1.6.9.1 quoted_identifier オプション

quoted_identifier オプションを使用して、区切り文字列の解釈を制御します。デフォルトでは、quoted_identifier オプションは On に設定されています。

quoted_identifier オプションが Off の場合、SQL の予約語は識別子として使用できません。

オプションの設定

次の文では、quoted_identifier オプション設定を On に変更します。

```
SET quoted_identifier On;
```

次の文では、quoted_identifier オプション設定を Off に変更します。

```
SET quoted_identifier Off;
```

区切り文字列の互換性のある解釈

各 DBMS で quoted_identifier オプションが同じ値に設定されていれば、ANSI/ISO SQL 標準の規則またはデフォルトの Transact-SQL の規則のどちらでも使用できます。

例

quoted_identifier オプションを On (デフォルト設定) で動作するように選択した場合、SQL キーワード **user** を指定した次の文はどちらの DBMS でも有効です。

```
CREATE TABLE "user" ( col1 char(5) )
go
INSERT "user" ( col1 )
   VALUES ( 'abcde' )
go
```

quoted_identifier オプションを off に設定する場合、次の文はどちらの DBMS でも有効です。次の例では、Chin は文字列であり、識別子ではありません。

```
SELECT *
FROM GROUPO.Employees
WHERE Surname = "Chin"
go
```

関連情報

予約語 [6 ページ]

1.1.7 探索条件

探索条件は、WHERE 句、HAVING 句、CHECK 句、ジョインの ON フレーズ、または IF 式に指定された基準です。探索条件は、述部とも言います。

構文

```
search-condition :
expression comparison-operator expression
| expression comparison-operator { [ ANY | SOME ] | ALL } ( subquery )
| expression IS [ NOT ] DISTINCT FROM expression
| expression IS [ NOT ] NULL
| expression [ NOT ] BETWEEN expression AND expression
| expression [ NOT ] LIKE pattern [ ESCAPE expression ]
| expression [ NOT ] SIMILAR TO pattern [ ESCAPE escape-expression ]
| expression [ NOT ] REGEXP pattern [ ESCAPE escape-expression ]
| expression [ NOT ] IN ( expression , ... )
     | ( query-expression )
| NOT search-condition
| CONTAINS (column-name [,... ] , query-string )
| EXISTS ( query-expression )
```

```
| search-condition [ { AND | OR } search-condition ] [ ... ]
| ( search-condition )
| ( search-condition , estimate )
|  search-condition IS [ NOT ] { TRUE | FALSE | UNKNOWN }
| expression IS [ NOT ] OF( type-name [ ONLY ],... )
| trigger-operation
```

```
comparison-operator :
=
| >
| <
| >=
| <=
| <>
| !=
| !<
| !>
```

```
trigger-operation :
INSERTING
| DELETING
| UPDATING [ ( column-name-string ) ]
| UPDATE( column-name )
```

パラメータ

- ALL 探索条件
- ANY 探索条件と SOME 探索条件
- IS [NOT] DISTINCT FROM 探索条件
- BETWEEN 探索条件
- CONTAINS 探索条件
- EXISTS 探索条件
- LIKE 探索条件
- SIMILAR TO 探索条件
- REGEXP 探索条件
- IS OF `type-expression`、および IS NOT OF `type-expression`
  このタイプ述部は、空間ジオメトリのサポート用に追加されましたが、既存のデータ型にも使用できます。

備考

探索条件は、テーブル内からローのサブセットを選択するか、または IF 文などの制御文でフローの制御を決めるために使用します。

SQL では、すべての条件が TRUE、FALSE、または UNKNOWN のいずれかに評価されます。これは、3 値的論理といいます。比較される値のいずれかが NULL の場合、比較結果は UNKNOWN になります。

比較結果が TRUE のみの場合、ローは探索条件を満たします。条件が UNKNOWN または FALSE のローは、探索条件を満たしません。

サブクエリは、多数の探索条件で使用される式の重要なクラスを構成します。

LIKE 探索条件、SIMILAR TO 探索条件、REGEXP 探索条件はよく似ています。

前提条件

データベースに接続されている必要があります。

関連する動作

なし。

このセクションの内容:

次の表は、3 値的論理で SQL の論理演算子 AND、OR、NOT、IS がどのように機能するかを示します。

明示的な選択性推定 [82 ページ]
データベースサーバは、統計情報を基に各文の実行に最も効率的な方法を判別します。

関連情報

SQL 文の式 [32 ページ]
LIKE 探索条件 [63 ページ]
SIMILAR TO 探索条件 [69 ページ]
REGEXP 探索条件 [67 ページ]
NULL 特別値 [99 ページ]

1.1.7.1 探索条件内のサブクエリ

1 つのカラムと、0 もしくは 1 つのローを確実に返すサブクエリは、式の途中などカラム名を使用できる位置であれば SQL 文内のどこにでも使用できます。

たとえば、サブクエリがローを 1 つだけ返すかぎりは、式を比較条件内のサブクエリと比較できます。サブクエリ (カラムは 1 つだけ) がローを 1 つ返す場合、そのローの値は式と比較されます。サブクエリがローを返さない場合、サブクエリの値は NULL です。

1 つのカラムと任意の数のローを返すサブクエリは、IN、ANY、ALL、SOME の各探索条件で使用できます。任意の数のカラムとローを返すサブクエリは、EXISTS 探索条件で使用できます。

標準

ANSI/ISO SQL 標準

コア機能。

関連情報

比較演算子 [15 ページ]

1.1.7.2 ALL 探索条件

ALL 探索条件を使用して、値をセット内のすべての値と比較します。

構文

```
expression comparison-operator ALL ( subquery )
```

```
comparison-operator:
=
| >
| <
| >=
| <=
| <>
| !=
| !<
| !>
```

備考

ALL 探索条件では、サブクエリの結果セットの値が空のセットだった場合、探索条件は TRUE と評価されます。それ以外の場合は、次に示すように、`expression` の値とサブクエリが返す結果セットに応じて、TRUE、FALSE、または UNKNOWN になります。

式の値	サブクエリが返す結果セットに 1 つ以上の NULL が含まれている場合	サブクエリが返す結果セットに NULL がない場合
NULL	UNKNOWN	UNKNOWN
NOT NULL	式の値と比較した結果が FALSE になる値がサブクエリの結果セットに 1 つでもあると、探索条件は FALSE と評価されます。それ以外の場合は、UNKNOWN になります。	式の値と比較した結果が FALSE になる値がサブクエリの結果セットに 1 つでもあると、探索条件は FALSE と評価されます。それ以外の場合は、TRUE になります。

標準

ANSI/ISO SQL 標準

コア機能。

1.1.7.3 ANY 探索条件と SOME 探索条件

ANY または SOME 探索条件を使用して、値をセット内のすべての値と比較します。

構文

```
expression comparison-operator { ANY | SOME }( subquery )
```

```
comparison-operator:
=
| >
| <
| >=
| <=
| <>
| !=
| !<
| !>
```

備考

キーワード ANY と SOME は同義です。

ANY 探索条件と SOME 探索条件では、サブクエリの結果セットが空のセットだった場合、探索条件は FALSE と評価されます。それ以外の場合は、次に示すように、`expression` の値とサブクエリが返す結果セットに応じて、TRUE、FALSE、または UNKNOWN になります。

式の値	サブクエリが返す結果セットに 1 つ以上の NULL が含まれている場合	サブクエリが返す結果セットに NULL がない場合
NULL	UNKNOWN	UNKNOWN
NOT NULL	式の値と比較した結果が TRUE になる値がサブクエリの結果セットに 1 つでもあると、探索条件は TRUE と評価されます。それ以外の場合は、UNKNOWN になります。	式の値と比較した結果が TRUE になる値がサブクエリの結果セットに 1 つでもあると、探索条件は TRUE と評価されます。それ以外の場合は、FALSE になります。

等号演算子のある ANY 探索条件または SOME 探索条件は、`expression` がサブクエリ結果のいずれかの値と等しい場合は TRUE、expression が NULL ではなく、サブクエリ結果のいずれの値とも等しくなく、結果セットに NULL が含まれていない場合は FALSE と評価されます。

注記

= ANY または **= SOME** を使用することは、IN キーワードを使用することと同等です。

標準

ANSI/ISO SQL 標準

コア機能。

1.1.7.4 IS DISTINCT FROM および IS NOT DISTINCT FROM 探索条件

IS DISTINCT FROM および IS NOT DISTINCT FROM 探索条件 を使用して、値がセット内の値とは異なるかどうかを評価します。

構文

```
expression1 IS [ NOT ] DISTINCT FROM expression2
```

備考

IS DISTINCT FROM および IS NOT DISTINCT FROM 探索条件は、検索引数可能であり、TRUE または FALSE と評価されます。

IS NOT DISTINCT FROM 探索条件は、`expression1` が `expression2` と等しい場合、または両方の式が NULL の場合、TRUE と評価されます。これは、次のような 2 つの探索条件の組み合わせと同じです。

```
(expression1 = expression2) IS TRUE OR ( expression1 IS NULL AND expression2 IS
NULL )
```

IS DISTINCT FROM 構文は、逆の意味を持ちます。つまり、IS DISTINCT FROM は、`expression1` が `expression2` と等しくなく、少なくとも一方の式が NULL ではない場合、TRUE と評価されます。これは、次の探索条件と同じです。

```
NOT(( expression1 = expression2) IS TRUE OR ( expression1 IS NULL AND expression2
IS NULL ))
```

標準

ANSI/ISO SQL 標準

IS [NOT] DISTINCT FROM 述部は、ANSI/ISO SQL 標準に定義されています。IS DISTINCT FROM 述部は、機能 T151 の "DISTINCT predicate" です。IS NOT DISTINCT FROM 述部は、機能 T152 の "DISTINCT predicate with negation" です。

1.1.7.5 BETWEEN 探索条件

BETWEEN 探索条件を使用して、値が別のセットの値の間にあるかどうかを評価します。

構文

```
expression [ NOT ] BETWEEN start-expression AND end-expression
```

備考

BETWEEN 探索条件は、TRUE、FALSE、または UNKNOWN として評価できます。NOT キーワードがない場合は、`expression` が `start-expression` と `end-expression` との間にあれば、条件は TRUE と評価されます。NOT キーワードを使用すると探索条件の意味が逆になりますが、UNKNOWN は変わりません。

BETWEEN 探索条件は、次の 2 つの不等式の組み合わせと等価です。

```
[ NOT ] ( expression >= start-expression AND expression <= end-expression )
```

標準

ANSI/ISO SQL 標準

コア機能。

1.1.7.6 LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件

REGEXP、LIKE、SIMILAR TO の各探索条件は、どれもパターンと文字列を一致させようとする点で似ています。また、これらの 3 つはすべて、文字列内の部分文字列ではなく、文字列全体に一致させようとします。

これらの 3 つの探索条件の基本的な構文は似ています。

```
expression search-condition pattern
```

LIKE、REGEXP、SIMILAR TO: `pattern` の定義の違い

- REGEXP では、SIMILAR TO でサポートされる正規表現の構文のスーパーセットがサポートされています。また、他の製品と互換性を持たせるため、REGEXP 探索条件では構文拡張がいくつかサポートされています。また、REGEXP と

SIMILAR TO は、デフォルトのエスケープ文字が異なり、アンダースコア (_)、パーセント (%)、および脱字記号 (^) の処理が異なります。REGEXP の動作は Perl 5 とほとんど一致します (ただし、Perl の構文と演算子がサポートされていない箇所を除く)。

- `pattern` の LIKE 構文は単純で、少数のワイルドカードがサポートされていますが、正規表現構文は完全にはサポートされていません。
- `pattern` の SIMILAR TO 構文では、ANSI/ISO SQL 標準に定義されている正規表現の構文を使用した強力なパターン一致を実行できます。

LIKE、REGEXP、SIMILAR TO: 文字の比較の違い

比較の実行において、REGEXP の動作は LIKE や SIMILAR TO とは異なります。REGEXP の比較では、データベースサーバはデータベース文字セットのコードポイント値を比較に使用します。この動作は、Perl などでの正規表現の実装と同じです。

LIKE と SIMILAR TO の場合、データベースサーバはデータベース照合の等価性とソート順を比較に使用します。この動作は、データベースが > や = などの比較演算子を評価する方法と同じです。

文字の比較方法が違うため、REGEXP と LIKE および SIMILAR TO とでは一致や範囲評価の結果が異なります。

一致における相違点

REGEXP では、コードポイント値を使用するため、パターンに含まれるリテラルと一致するのは、厳密な意味で同じ文字の場合のみです。このため、REGEXP での一致は、データベース照合、大文字と小文字の区別、またはアクセント記号の区別といった要因には影響されません。たとえば、'a' との一致として 'A' が返されることはありません。

LIKE と SIMILAR TO ではデータベース照合が使用されるため、文字が等価かどうかを判断する際に、大文字と小文字やアクセント記号の区別に影響されます。たとえば、データベース照合で大文字と小文字やアクセント記号が区別されない場合、一致でも大文字と小文字やアクセント記号が区別されません。そのため、'a' との一致として 'A' が返されることがあり得ます。

範囲評価における相違点

REGEXP では範囲評価にコードポイントが使用されるため、文字が範囲内と判断されるのは、その文字のコードポイント値が、範囲の最初と最後のコードポイント値と同じかその間にある場合になります。たとえば、比較 `x REGEXP '[A-C]'` は、単一文字 `x` について、`CAST(x AS BINARY) >= CAST(A AS BINARY) AND CAST(x AS BINARY) <= CAST(C AS BINARY)` と同義です。

LIKE と SIMILAR TO では、範囲評価に照合のソート順が使用されるため、文字が範囲内と判断されるのは、その文字の照合における位置が、範囲の最初と最後の文字の位置と同じかその間にある場合になります。たとえば、比較 `x SIMILAR TO '[A-C]'` (`x` は 1 文字) は `x >= A AND x <= C` と同義であり、比較演算子は照合のソート順を使用して評価されます。

次の表は、LIKE、SIMILAR TO、REGEXP が評価した範囲 `'[A-C]'` に該当する文字を示しています。2 つのデータベースはどちらも 1252LATIN1 照合を使用しますが、最初のデータベースでは大文字と小文字が区別されず、2 番目では区別されます。

	LIKE/SIMILAR TO '[A-C]'	REGEXP '[A-C]'
demo.db (大文字と小文字の区別なし)	A、B、C、a、b、c、ª、À、Á、Â、Ã、Ä、Å、Æ、Ç、à、á、â、ã、ä、å、æ、ç	A、B、C

	LIKE/SIMILAR TO '[A-C]'	REGEXP '[A-C]'
`charsensitive.db`(大文字と小文字の区別あり)	A、B、C、b、c、À、Á、Â、Ã、Ä、Å、Æ、Ç、ç	A、B、C

この結果から次のことを確認できます。

- LIKE と SIMILAR TO は、範囲内のアクセント記号付き文字を含めます。
- LIKE と SIMILAR TO では、データベースで大文字と小文字が区別されるかどうかに応じて、該当する文字が変わります。具体的には、結果に範囲内の小文字はすべて含まれますが、大文字と小文字が区別されるデータベースでの検索にそのような動作は期待されません。
  同様に、大文字と小文字を区別するデータベースで、範囲内に含まれる文字に一貫性がないように見えることがあります。たとえば、大文字と小文字を区別するデータベースでの `SIMILAR TO '[A-C]'` の結果には、A、b、B、c、C が含まれますが、a は含まれません。これは、ソート順で a が A より前に位置しているためです。
- REGEXP は、データベースで大文字と小文字が区別されるかどうかに関係なく、A、B、C を返します。範囲に小文字も含めるには、その条件を範囲の定義に追加する必要があります。たとえば、`REGEXP '[a-cA-C]'` のように記述します。
- REGEXP の結果に含まれる文字のセットは、データベースで大文字と小文字が区別されるかどうかに関係なく、同じです。

データベースで使用されている照合、大文字と小文字の区別、アクセント記号の区別が、上記の例と異なる場合でも、データベースに接続して上記の文を実行することによって、同様のテストを実行し、LIKE、SIMILAR TO、REGEXP が返す内容を確認できます。

```
SELECT CHAR( row_num ) FROM RowGenerator WHERE CHAR( row_num ) LIKE '[A-C]';
SELECT CHAR( row_num ) FROM RowGenerator WHERE CHAR( row_num ) REGEXP '[A-C]';
SELECT CHAR( row_num ) FROM RowGenerator WHERE CHAR( row_num ) SIMILAR TO '[A-
C]';
```

このセクションの内容:

関連情報

正規表現の概要 [38 ページ]
正規表現の構文 [39 ページ]
正規表現の例 [49 ページ]

1.1.7.6.1 LIKE 探索条件

LIKE 探索条件を使用して、値がセット内の値と似ているかどうかを評価します。

構文

LIKE 探索条件の構文は次のとおりです。

```
expression [ NOT ] LIKE pattern [ ESCAPE escape-character ]
```

パラメータ

expression

検索される文字列。

pattern

`expression` を検索するためのパターン。

escape-character

アンダースコアやパーセント記号などの特殊文字をエスケープするために使用する文字。

備考

LIKE 探索条件は、`expression` を `pattern` に一致させようとし、TRUE、FALSE、または UNKNOWN と評価されます。

`expression` が `pattern` に一致すれば、探索条件は TRUE と評価されます (NOT が指定されない場合)。`expression` または `pattern` が NULL 値の場合、探索条件は UNKNOWN と評価されます。NOT キーワードを使用すると探索条件の意味が逆になりますが、UNKNOWN は変わりません。

`expression` は、CHAR または NCHAR の文字列として解釈されます。`expression` の内容全体が一致に使用されます。同様に、`pattern` は、CHAR または NCHAR の文字列として解釈され、次の表のサポートされているワイルドカードをいくつでも含めることができます。

ワイルドカード	一致するもの
_ (アンダースコア)	任意の 1 文字。たとえば、`a_` は ab と ac に一致しますが、a には一致しません。
% (パーセント記号)	0 個以上の文字からなる任意の文字列。たとえば、`bl%` は bl と bla に一致します。
[]	指定範囲内、または一連の指定文字の任意の 1 文字。たとえば、`T[oi]m` は Tom または Tim に一致します。

ワイルドカード	一致するもの
[^]	指定範囲、または一連の指定文字に含まれない任意の 1 文字。たとえば、M[^c] は Mb と Md に一致しますが、Mc には一致しません。

その他すべての文字は正確に一致しなければなりません。

たとえば、次のような探索条件は、文字 a で始まり、末尾から 2 つ目の文字が b の name のローの場合、TRUE になります。

```
... name LIKE 'a%b_'
```

escape-character を指定する場合は、シングルバイトの CHAR 文字または NCHAR 文字として評価される文字を指定します。pattern 内のパーセント記号、アンダースコア、左側の角カッコ、または別のエスケープ文字の前にエスケープ文字を置くことによって、特殊文字に特別な意味を持たせないようにできます。この方法でエスケープすると、パーセント記号はパーセント記号に一致し、アンダースコアはアンダースコアに一致します。

長さが最大 126 バイトのパターンがサポートされています。

LIKE 探索条件を使用するさまざまな方法

検索対象	例	その他の情報
一連の文字	LIKE 'sm[iy]th'	検索対象の一連の文字は、角カッコ内にリストして指定します。この例では、探索条件は *smith* と *smyth* に一致します。
範囲内の文字	LIKE '[a-r]ough'	検索対象の文字範囲は、角カッコ内に範囲を書いて指定します。範囲の始めと終わりの間にハイフンを書きます。この例では、探索条件は bough と rough に一致しますが、tough には一致しません。 文字の範囲 [a-z] は "a 以上 z 以下" と解釈され、データベースの照合では大なり演算と小なり演算が実行されます。 範囲は、まず小さい方の値、次に大きい方の値を指定してください。たとえば、[z-a] は何も一致しません。これは、[z-a] の範囲にいずれの文字も一致しないためです。

検索対象	例	その他の情報
文字範囲と一連の文字の結合	`... LIKE '[a-rt]ough'`	角カッコ内で、文字範囲と一連の文字を結合できます。この例では、`... LIKE '[a-rt]ough'` は bough、rough、tough に一致します。 パターン [a-rt] は、a ～ r の範囲に含まれる文字または t と一致する 1 文字と解釈されます。
範囲外の 1 文字	`... LIKE '[^a-r]ough'`	脱字記号 (^) は、検索から除外する文字範囲を指定します。この例では、`LIKE '[^a-r]ough'` は文字列 tough に一致しますが、文字列 rough や bough には一致しません。 脱字記号は、この記号以外のカッコ内の内容を否定します。たとえば、角カッコ [^a-rt] は、a ～ r の範囲に含まれない文字かつ t ではない 1 文字と解釈されます。
後続ブランクのある探索パターン	`'90 '`、`'90[ ]'`、および `'90_'`	探索パターンに後続ブランクが含まれる場合、データベースサーバではブランクが含まれる値に対してのみパターンが一致します。文字列へのブランクの埋め込みは行われません。たとえば、検索されている値が幅 3 文字以上の CHAR 型または VARCHAR 型カラムにあっても、'90 '、'90[ ]'、'90_' のパターンは、式 '90 ' には一致しますが、式 '90' には一致しません。

文字範囲と一連の文字の特殊なケース

角カッコ内の任意の 1 文字は、その文字を指しています。たとえば、`[a]` は、文字 a に一致します。`[^]` は脱字記号、`[%]` はパーセント記号 (パーセント記号は、このコンテキストではワイルドカードになりません)、`[_]` はアンダースコア文字に一致します。また、`[[]` は文字 [に一致します。

その他の特殊なケースには、次のものがあります。

- パターン `[a-]` は、文字 a または - に一致します。
- パターン `[]` は、一致する文字がないのでローを返すことはありません。
- パターン `[` または `[abp-q` は、閉じカッコが欠けているため構文エラーを返します。
- 角カッコ内ではワイルドカードは使用できません。パターン `[a%b]` は a、%、または b のいずれかを検出します。
- 脱字記号を使用しても、カッコ内の先頭になければ範囲を否定できません。パターン `[a^b]` は a、^、または b のいずれかを検出します。

大文字と小文字の区別と、比較の実行

データベース照合が大文字と小文字を区別する場合は、探索条件でも大文字と小文字を区別します。大文字と小文字を区別する照合で大文字と小文字を区別しない検索を実行するには、大文字と小文字を両方指定します。たとえば、次の探索条件では、文字列 Bough、rough、TOUGH が true と評価されます。

```
LIKE '[a-zA-Z][oO][uU][gG][hH]'
```

文字列単位で比較する等価演算子 (=) やその他の演算子とは異なり、比較が文字単位で実行されます。たとえば、UCA 照合 (照合が UCA に設定された CHAR または NCHAR) で比較が行われる場合、`'Æ'='AE'` は true ですが、`'Æ' LIKE 'AE'` は false です。

文字単位で比較する一致では、検索される式に含まれる各文字が 1 文字ごとに一致するか (照合の文字等価を使用します)、または LIKE 式のワイルドカードに一致する必要があります。

各国文字 (NCHAR) サポート

LIKE 探索条件は、CHAR と NCHAR の文字列の比較に使用できます。この場合、共通データ型を使用して比較が行われるように、文字セット変換が実行されます。次に、文字単位の比較が実行されます。

引用符で囲んだ値の前に N を付けることによって (たとえば、`expression LIKE N'pattern'`)、`expression` または `pattern` を NCHAR 文字列リテラルとして指定できます。また、CAST 関数を使用して、CHAR または NCHAR にパターンをキャストすることもできます (たとえば、`expression LIKE CAST(pattern AS datatype)`。

ブランクが埋め込まれたデータベース

ブランクが埋め込まれたデータベースの場合、`expression` と `pattern` の一致では、左から右の順に文字単位の比較が行われるため、LIKE パターンのセマンティックが変わることはありません。評価時に、`expression` または `pattern` の値に追加のブランクは埋め込まれません。このため、式 `a1` はパターン a1 に一致しますが、パターン 'a1 ' (a1 の後ろにスペースがある) または a1_ には一致しません。

標準

ANSI/ISO SQL 標準

LIKE 探索条件は、ANSI/ISO SQL 標準のコア機能です。ただし、ソフトウェアでは大文字と小文字を区別しない照合とブランク埋め込みがサポートされるため、この標準の動作とは微妙な違いがあります。

ソフトウェアでは、オプションの ANSI/ISO SQL 言語機能 F281 をサポートしています。これにより、パターンとエスケープ式を実行時に評価される任意の式にできます。機能 F281 を使用すると、`expression` を単純なカラム参照よりも複雑な式にすることもできます。

角カッコ [] に含まれる文字範囲と一連の文字の使用は標準にありません。

ソフトウェアでは、ANSI/ISO SQL 機能 T042 をサポートしています。これにより、LIKE 探索条件で LONG VARCHAR 値の文字列式を参照できます。

NCHAR の文字列式またはパターンを指定する LIKE 探索条件は、オプションの ANSI/ISO SQL 言語機能 F421 です。

関連情報

CHAR と NCHAR の比較 [187 ページ]
文字列リテラル [13 ページ]
CAST 関数 [データ型変換] [264 ページ]
REGEXP 探索条件 [67 ページ]
SIMILAR TO 探索条件 [69 ページ]

1.1.7.6.2 REGEXP 探索条件

パターンと文字列を照合します。

構文

```
expression [ NOT ] REGEXP pattern [ ESCAPE escape-expression ]
```

パラメータ

expression

検索される文字列。

pattern

`expression` を検索するための正規表現。

escape-expression

一致で使用されるエスケープ文字。デフォルトは円記号 (¥) です。

備考

REGEXP 探索条件は文字列全体に一致し、部分文字列には一致しません。部分文字列で文字列に一致させるには、残り部分と一致するワイルドカードで文字列を囲みます (`.*pattern.*`)。たとえば、`SELECT ... WHERE Description REGEXP 'car'` は car にのみ一致し、sportscar には一致しません。一方、`SELECT ... WHERE Description REGEXP '.*car'` は、car や sportscar など、car で終わるすべての文字列に一致します。または、REGEXP_SUBSTR 関数を使用するようにクエリを書き換えることができます。この関数は、文字列内の部分文字列を検索します。

部分文字クラスのみに一致させる場合は、外側の角カッコと、部分文字クラス用の角カッコを使用する必要があります。たとえば、`expression REGEXP '[[:digit:]]'` のように記述します。

データベース照合と一致

REGEXP を使用した場合に、パターンに含まれるリテラルと文字とが一致するのは、厳密な意味で同じ文字の場合 (コードポイント値が同じ場合) のみです。文字クラスによる範囲 (たとえば `'[A-F]'`) と一致する文字は、そのコードポイント値が、範囲の最初に指定されている文字 (A) のコードポイント値以上で、なおかつ、範囲で次に指定されている文字 (F) のコードポイント値以下の場合に限られます。

文字列単位で比較する等価演算子 (=) やその他の演算子とは異なり、比較が文字単位で実行されます。たとえば、UCA 照合 (照合が UCA に設定された CHAR または NCHAR) で比較が行われる場合、`'Æ'='AE'` は true ですが、`'Æ' REGEXP 'AE'` は false です。

各国文字 (NCHAR) サポート

REGEXP 探索条件は、CHAR と NCHAR の文字列の比較に使用できます。この場合、共通データ型を使用して比較が行われるように、文字セット変換が実行されます。次に、コードポイント単位の比較が実行されます。

引用符で囲んだ値の前に N を付けることによって (たとえば、`expression REGEXP N'pattern'`)、`expression` または `pattern` を NCHAR 文字列リテラルとして指定できます。また、CAST 関数を使用して、CHAR または NCHAR にパターンをキャストすることもできます (たとえば、`expression REGEXP CAST(pattern AS datatype)`。

標準

ANSI/ISO SQL 標準

REGEXP 探索条件は標準にはありませんが、ANSI/ISO SQL 標準の SQL 言語機能 F841 の LIKE_REGEX 探索条件とほぼ互換性があります。

ソフトウェアでは、SQL 機能 F281 をサポートしています。これにより、パターンとエスケープ式を実行時に評価される任意の式にできます。機能 F281 を使用すると、`expression` を単純なカラム参照よりも複雑な式にすることもできます。

ソフトウェアでは、SQL 機能 T042 をサポートしています。これにより、REGEXP 探索条件で LONG VARCHAR 値の文字列式を参照できます。

NCHAR の文字列式またはパターンを指定する REGEXP 探索条件は、機能 F421 です。

関連情報

CHAR と NCHAR の比較 [187 ページ]

文字列リテラル [13 ページ]
正規表現の概要 [38 ページ]
CAST 関数 [データ型変換] [264 ページ]
SIMILAR TO 探索条件 [69 ページ]
LIKE 探索条件 [63 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]
正規表現:特殊部分文字クラス [42 ページ]

1.1.7.6.3 SIMILAR TO 探索条件

パターンと文字列を照合します。

構文

```
expression [ NOT ] SIMILAR TO pattern [ ESCAPE escape-expression ]
```

パラメータ

expression

検索される式。

pattern

`expression` を検索するための正規表現。

escape-expression

一致で使用するエスケープ文字。デフォルトのエスケープ文字は NULL 文字で、文字列リテラル内では '¥x00' のように指定します。

正規表現の構文	意味
¥x	x と同じものに一致します。ここでは、エスケープ文字は円記号 (¥) と見なされます。たとえば、¥[は '[' に一致します。
x	任意の文字 (メタ文字以外) がそれ自体に一致します。たとえば、A は 'A' に一致します。

備考

部分文字列と文字列を一致させるには、パーセント記号のワイルドカードを使用します (%expression)。たとえば、`SELECT ... WHERE Description SIMILAR TO 'car'` は car にのみ一致し、sportscar には一致しません。一方、`SELECT ... WHERE Description SIMILAR TO '%car'` は、car や sportscar など、car で終わるすべての文字列に一致します。

部分文字クラスのみに一致させる場合は、外側の角カッコと、部分文字クラス用の角カッコを使用する必要があります。たとえば、`expression` SIMILAR TO '[[:digit:]]') のように指定します。

文字列単位で比較する等価演算子 (=) やその他の演算子とは異なり、比較が文字単位で実行されます。たとえば、UCA 照合 (照合が UCA に設定された CHAR または NCHAR) で比較が行われる場合、`'Æ'='AE'` は true ですが、`'Æ' SIMILAR TO 'AE'` は false です。

文字単位で比較して一致するには、検索される式に含まれる各文字が、SIMILAR TO パターンに含まれる 1 文字またはワイルドカード文字に一致する必要があります。

データベース照合と一致

SIMILAR TO では、文字が等価かどうかの判断や文字クラス範囲の評価に、照合が使用されます。たとえば、データベースで大文字と小文字やアクセント記号が区別されない場合、一致でも大文字と小文字やアクセント記号が区別されません。範囲も、照合のソート順を使用して評価されます。

各国文字 (NCHAR) サポート

SIMILAR TO 探索条件は、CHAR と NCHAR の文字列の比較に使用できます。この場合、共通データ型を使用して比較が行われるように、文字セット変換が実行されます。次に、文字単位の比較が実行されます。

引用符で囲んだ値の前に N を付けることによって (たとえば、`expression SIMILAR TO N'pattern'`)、`expression` または `pattern` を NCHAR 文字列リテラルとして指定できます。また、CAST 関数を使用して、CHAR または NCHAR にパターンをキャストすることもできます (たとえば、`expression SIMILAR TO CAST(pattern AS datatype)`。

標準

ANSI/ISO SQL 標準

SIMILAR TO 述部は、オプションの ANSI/ISO SQL 言語機能 T141 です。

関連情報

正規表現の概要 [38 ページ]
CHAR と NCHAR の比較 [187 ページ]
文字列リテラル [13 ページ]
CAST 関数 [データ型変換] [264 ページ]
REGEXP 探索条件 [67 ページ]
LIKE 探索条件 [63 ページ]
REGEXP_SUBSTR 関数 [文字列] [484 ページ]

正規表現:特殊部分文字クラス [42 ページ]

1.1.7.7 IN 探索条件

IN 探索条件を使用して、値がセット内で見つかるかどうかを評価します。

構文

```
expression [ NOT ] IN { ( query-expression ) | ( expression-list ) }
```

備考

IN 探索条件は、query-expression によって返された値のセットまたは expression-list で指定した値のセットと expression を比較します。NOT キーワードがない場合、IN 探索条件は次の規則に従って評価されます。

- expression が NULL でなく、少なくとも 1 つの値と等しい場合、TRUE です。
- expression が NULL で、値リストが空でない場合、または少なくとも 1 つの値が NULL で、expression が他の値のいずれとも等しくない場合、UNKNOWN です。
- expression が NULL で、query-expression が値を返さない場合、または expression が NULL でなく、いずれの値も NULL でなく、expression がいずれの値とも等しくない場合、FALSE です。

NOT キーワードを指定すると、評価結果の TRUE と FALSE が逆になります。

探索条件 expression IN (expression-list) は expression = ANY (expression-list) と同義です。

探索条件 expression NOT IN (expression-list) は expression <> ALL (expression-list) と同義です。

expression-list の式には、結果が単一のローと単一のカラムになるリテラル、変数、ホスト変数、またはクエリ式を指定できます。

標準

ANSI/ISO SQL 標準

コア機能。

1.1.7.8 CONTAINS 探索条件

CONTAINS 探索条件を使用して、値がセット内に含まれるかどうかを評価します。

構文

```
CONTAINS ( column-name [,...], contains-query-string )
```

```
contains-query-string :
simple-expression
| or-expression
```

```
simple-expression :
primary-expression
| and-expression
```

```
or-expression :
simple-expression { OR | | } contains-query-string
```

```
primary-expression :
basic-expression
| FUZZY " fuzzy-expression "
| and-not-expression
```

```
and-expression :
primary-expression [ AND | & ] simple-expression
```

```
and-not-expression :
primary-expression [ AND | & ] { NOT | - } basic-expression
```

```
basic-expression :
term
| phrase
| ( contains-query-string )
| near-expression
| before-expression
```

```
fuzzy-expression :
term
| fuzzy-expression term
```

```
term :
simple-term
| prefix-term
```

```
prefix-term :
simple-term*
```

```
phrase :
" phrase-string "
```

```
near-expression :
term NEAR [ [ min-distance ], max-distance ] term
| term { NEAR | ~ } term
```

```
before-expressions :
term BEFORE [ [ min-distance ] max-distance ] term
| term BEFORE term
```

```
phrase-string :
term
| phrase-string term
```

simple-term: 検索に使われる単一のインデックス単語を表す、ホワイトスペースと特殊記号で区切られた文字列

distance: 正の整数

パラメータ

and-expression

primary-expression と simple-expression の両方がテキストインデックスに存在する必要があることを指定します。

デフォルトでは、単語または式の間に演算子が指定されていない場合、and-expression と見なされます。たとえば、'a b' は、'a AND b' と解釈されます。

アンパサンド (&) は AND の代わりとして使用でき、式や単語のどちら側にも配置できます ('a &b' など)。

and-not-expression

primary-expression はテキストインデックスに存在する必要があるが、basic-expression はテキストインデックスに存在してはならないことを指定します。これは否定とも呼ばれています。

否定を示すのにハイフンを使用する場合、そのハイフンの左がスペース、右に単語が隣接していることが必要です。それ以外の場合、ハイフンは否定と解釈されません。たとえば、'a -b' は 'a AND NOT b' と同義ですが、'a - b' ではハイフンが無視され、この文字列は 'a AND b' と同義になります。'a-b' はフレーズ '"a b"' と同義です。

or-expression

少なくとも simple-expression または contains-query-string のいずれかがテキストインデックスに存在する必要があることを指定します。たとえば、'a|b' は、'a OR b' と解釈されます。

fuzzy-expression

指定した単語と似ている単語を検索します。あいまい一致は、NGRAM テキストインデックスに対してのみサポートされています。

near-expression

互いの距離が近い単語を検索します。これは近接検索とも呼ばれています。たとえば、'b NEAR[5] c' は、互いの距離が 5 語以下の b と c の出現を探します。単語の順序は重要ではなく、'b NEAR c' は 'c NEAR b' と同義です。

最大距離が指定されない場合、デフォルトの距離は 10 になります。最小距離が指定されない場合、デフォルトの距離は 1 になります。

クエリ 'apple NEAR[2, 10] tree' は次のドキュメントとは一致しません。

```
'apple grows on the tree'
```

```
'apple and tree'
'tree and apple'
```

ただし、このクエリは次のドキュメントとは一致しません。

```
'apple tree'
'tree apple'
```

NEAR の代わりとしてチルダ (~) を指定することもできます。チルダの使用は NEAR を距離なしで指定することに相当するため、デフォルトの最大 10 語および最小 1 語が適用されます。チルダを指定した場合は最大距離または最小距離を指定できません。常に 10 語になります。

NEAR 式を連ねて使用することはできません (たとえば `'a NEAR[1] b NEAR[1] c'` は無効です)。

before-expression

`before-expression` を使用して、別の単語の前にある単語を検索します。これは近接検索とも呼ばれています。照合するテキスト内の引数は、CONTAINS クエリ文字列で指定されたのと同じ順序で配置される必要があります。たとえば、`'apple BEFORE[2, 10] tree'` は次のドキュメントと一致します。

```
'apple grows on the tree'
'apple and tree'
```

ただし、このクエリは次のドキュメントとは一致しません。

```
'tree and apple'
'apple tree'
'tree apple'
```

次のクエリは上の例と同義です。

```
'apple BEFORE tree'
'apple BEFORE[10] tree'
'apple BEFORE[1, 10] tree'
```

最大距離が指定されない場合、デフォルトの距離は 10 になります。最小距離が指定されない場合、デフォルトの距離は 1 になります。

prefix-term

指定されたプレフィクスで始まる単語を検索します。たとえば、`'datab*'` は datab で始まる単語を検索します。これはプレフィクス検索とも呼ばれています。プレフィクス検索では、単語のアスタリスクより左の部分に対して一致が実行されます。

備考

CONTAINS 検索条件は、引数としてカラムリストと `contains-query-string` を取ります。これは、探索条件 (述部とも呼ばれる) を指定できる場所ならどこでも使用でき、TRUE または FALSE を返します。`contains-query-string` は、クエリ実行時に認識できる値を持つ、定数文字列または変数である必要があります。`contains-query-string` は、NULL や空の文字列にはできず、300 を超える有効な単語を指定することはできません。有効な単語とは、許可される単語長以内の長さで、STOPLIST に含まれていない単語です。`contains-query-string` に 300 を超える有効な単語がある場合は、エラーが返されます。

テキスト設定のために contains-query-string 内のすべての単語が削除される場合、CONTAINS 探索条件の結果は FALSE となります。

複数のカラムが指定された場合は、それらがどれも同じベーステーブルに解決される必要があります (テキストインデックスが複数のベーステーブルにわたることはありません)。ベーステーブルは FROM 句で直接参照できます。また、ビューまたは派生テーブルで使用することもできます。column-name がビューまたは派生テーブルのカラムである場合、次の条件が真であることを前提として、クエリ式のネストしたクエリブロック内に CONTAINS 探索条件が再帰的にプッシュされます。

- 元の CONTAINS 探索条件内で参照されるクエリ式のカラムはすべて、ネストしたクエリブロック内の単一のベーステーブルに解決されます。
- ネストしたクエリブロック内で複数のテキストブロックが使用されている場合、これらのテキストインデックス設定は同じである必要があります。
- ネストしたクエリブロックに TOP、LIMIT、FIRST、または Window 集合関数を含めることはできません。
- 元の CONTAINS は、論理和述部の WHERE 句内にある必要があります。元の CONTAINS 探索条件内で参照されるカラムは、同じビューまたは派生テーブルのカラムに解決される必要があります。

クエリ文字列に英数字以外の文字を使用する場合は、次の点に注意が必要です。

- アスタリスクが単語内にあると、エラーが返されます。
- 英数字以外の文字 (特殊文字を含む) を fuzzy-expression に使用しないでください。このような文字は、ホワイトスペースとして扱われ、単語区切りとして機能します。
- 可能であれば、特殊文字ではない英数字以外の文字をクエリ文字列に含めないでください。特殊文字ではない英数字以外の文字があると、それを含む単語がフレーズとして扱われ、その文字の位置で単語が区切られます。たとえば、'things we've done' は、'things "we ve" done' と解釈されます。

フレーズの内部にあっても特殊文字として解釈されるのは、アスタリスクのみです。それ以外の特殊文字がフレーズの内部にあると、ホワイトスペースとして扱われ、単語区切りとして機能します。

contains-query-string の解釈は、主に次の 2 つのステップで行われます。

手順 1: 演算子の解釈と優先度の適用

このステップでは、キーワードが演算子として解釈され、優先度の規則が適用されます。

手順 2: テキスト設定オブジェクトの設定の適用

このステップでは、テキスト設定オブジェクトの設定が単語に適用されます。たとえば、NGRAM テキストインデックスでは、単語は N-gram 表現に分解されます。このステップで、単語長の設定値を超えているクエリ単語またはストップリストに載っているクエリ単語が削除されます。

CONTAINS 探索条件における演算子の優先度

クエリの評価中、式は次の優先順位に従って評価されます。

1. FUZZY、NEAR
2. AND NOT
3. AND
4. OR

キーワードとしての BEFORE の処理

現在、BEFORE キーワードは演算子としてサポートされていません。たとえば、`CONTAINS(column-name, 'a before b')` と指定すると、エラーが返されます。代わりに NEAR キーワードを使用してクエリを作成します。

フレーズクエリの一部である場合は、単語 **before** を検索できます。たとえば、`CONTAINS(column-name, '"a before b"')` のように記述します。このクエリはフレーズ "a before b" を検索します。

アスタリスク (*) を使用できる構文

アスタリスクはプレフィクス検索で使用します。アスタリスクは、クエリ文字列の末尾、または、スペース、アンパサンド、パイプ記号、閉じカッコ、または終了引用符の直前に指定できます。それ以外の位置にアスタリスクを指定すると、エラーが返されます。

次の表は、アスタリスクの使用法として可能なものを示しています。

クエリ文字列	同義の文字列	解釈
`'th*'`		th で始まる単語を検索します。
`'th*&best'`	`'th* AND best'` および `'th* best'`	th で始まる単語と、単語 best を検索します。
`'th*|best'`	`'th* OR best'`	th で始まる単語、または単語 best を検索します。
`'very&(best|th*)'`	`'very AND (best OR th*)'`	単語 very と、単語 best または th で始まる単語を検索します。
`'"fast auto*"'`		auto で始まる単語が直後に続く、単語 fast を検索します。
`'"auto* price"'`		単語 price が直後に続く、auto で始まる単語を検索します。

注記

アスタリスクを含むクエリ文字列の解釈は、テキスト設定オブジェクトの設定によって異なります。

ハイフン (-) を使用できる構文

ハイフンは、単語や式の否定に使用でき、NOT と同義です。ハイフンが否定と解釈されるかどうかは、クエリ文字列内での位置によります。たとえば、ハイフンの直後に単語や式が続く場合、そのハイフンは否定と解釈されます。ハイフンが単語内に埋め込まれている場合は、そのままハイフンとして解釈されます。

否定を示すために使用する場合、ハイフンの直前にホワイトスペース、直後に式がある必要があります。

fuzzy-expression のフレーズ内で使用すると、ハイフンはホワイトスペースとして扱われ、単語区切りとして機能します。

次の表に、ハイフンを使用できる構文を示します。

クエリ文字列	同義の文字列	解釈
'the -best'	'the AND NOT best'、'the AND -best'、'the & -best'、'the NOT best'	単語 the と、best ではない単語を検索します。
'the -(very best)'	'the AND NOT (very AND best)'	単語 the と、very でも best でもない単語を検索します。
'the -"very best"'	'the AND NOT "very best"'	単語 the と、フレーズ very best ではないものを検索します。
'alpha-numerics'	'"alpha numerics"'	単語 numerics が直後に続く、単語 alpha を検索します。
'wild - west'	'wild west'、および'wild AND west'	単語 wild と、単語 west を検索します。

特殊文字を使用できる構文

次の表は、アスタリスクとハイフンを除くすべての特殊文字を使用できる構文を示します。

以下に示されている特殊文字は、フレーズ内に出現した場合は特殊文字とは見なされず、削除されます。

> **注記**
>
> 文字列リテラルの指定に関連する制約事項は、クエリ文字列にも該当します。たとえば、アポストロフィはエスケープする必要があります。

文字または構文	使用例と備考
アンパサンド (&)	アンパサンドは AND と同義で、次のような指定が可能です。 • 'a & b' • 'a &b' • 'a&b' • 'a& b'
パイプ記号 (|)	パイプ記号は OR と同義で、次のような指定が可能です。 • 'a|b' • 'a |b' • 'a | b' • 'a| b'

文字または構文	使用例と備考
二重引用符 (")	二重引用符は、順序と相対的な距離が重要な単語並びを含める場合に使用します。たとえば、クエリ文字列 **'learn "full text search"'** では、**full text search** はフレーズです。この例において、learn はこのフレーズの前後どちらにあっても、別のカラムにあってもかまいませんが (テキストインデックスが複数カラムに対して構築されている場合)、完全に一致するフレーズが 1 つのカラム内で見つかる必要があります。
丸カッコ ()	丸カッコは、式の評価順がデフォルトの順序とは違う場合に、その順序を指定するために使用します。たとえば、**'a AND (b|c)'** は "a と、b または c" と解釈されます。
チルダ (~)	チルダは NEAR[10] と同義です。クエリ文字列 **'full~text'** は **'full NEAR text'** と同義で、"単語 text から 10 語以内の距離にある単語 full" と解釈されます。 距離とチルダを同時に指定することはできません。
角カッコ [ ]	角カッコはキーワード NEAR と組み合わせて `distance` の指定に使用します。角カッコをそれ以外の目的で使用すると、エラーが返されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

文字列リテラル [13 ページ]
FROM 句 [1112 ページ]
sa_char_terms システムプロシージャ [1452 ページ]
sa_nchar_terms システムプロシージャ [1574 ページ]

1.1.7.9 EXISTS 探索条件

EXISTS 探索条件を使用して、値がセット内で見つかるかどうかを評価します。

構文

```
EXISTS ( subquery )
```

備考

EXISTS 探索条件は、サブクエリ結果にローが少なくとも 1 つあれば TRUE で、ローがなければ FALSE です。EXISTS 探索条件は、UNKNOWN にはなりません。

標準

ANSI/ISO SQL 標準

コア機能。

1.1.7.10 IS NULL および IS NOT NULL 探索条件

IS NULL 探索条件を使用して、セット内の値が NULL であるかどうかを評価します。

構文

```
expression IS [ NOT ] NULL
```

備考

NOT キーワードがない場合、expression が NULL 値なら IS NULL 探索条件は TRUE、それ以外の場合は FALSE です。NOT キーワードを使用すると探索条件の意味が逆になります。

標準

ANSI/ISO SQL 標準

コア機能。

1.1.7.11 真理値探索条件

IS TRUE 探索条件を使用して、条件が指定された値になるかどうかを評価します。

構文

```
IS [ NOT ] truth-value
```

備考

NOT キーワードがない場合、`condition` が指定された `truth-value` (TRUE、FALSE、UNKNOWN のいずれか) と評価されれば、検索条件は TRUE になります。それ以外の場合、値は FALSE です。NOT キーワードを使用すると探索条件の意味が逆になりますが、UNKNOWN は変わりません。

標準

ANSI/ISO SQL 標準

真理値探索条件は、オプションの ANSI/ISO SQL 言語機能 F571 によって構成されています。

1.1.7.12 3 値的論理

次の表は、3 値的論理で SQL の論理演算子 AND、OR、NOT、IS がどのように機能するかを示します。

AND 演算子

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR 演算子

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT 演算子

TRUE	FALSE	UNKNOWN
FALSE	TRUE	UNKNOWN

IS 演算子

IS	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE
UNKNOWN	FALSE	FALSE	TRUE

標準

ANSI/ISO SQL 標準

コア機能。真理値テスト (IS UNKNOWN など) **は、**SQL **言語機能** F571 **によって構成されています。**

関連情報

NULL **特別値** [99 **ページ**]

1.1.7.13 明示的な選択性推定

データベースサーバは、統計情報を基に各文の実行に最も効率的な方法を判別します。

データベースサーバは、それらの統計を自動的に収集、更新します。これらの統計は、データベースのシステムテーブル ISYSCOLSTAT に永久的に格納されます。ある文の処理中に収集された統計は、以降の文の効率的な実行方法を見いだすときに使用できます。

場合によっては、統計情報が不正確になったり、関連統計情報が使用不能になったりすることがあります。このような状況がもっとも発生しやすいのは、大量のデータが追加、更新、または削除されてから実行されたクエリが少ない場合です。このような場合は、CREATE STATISTICS 文を実行します。

特定の実行プランに問題がある場合は、オプティマイザに関するヒントを使用して、特定のインデックスの使用を要求できます。

ただし、特異な状況では、この方法では効果的でないことがあります。そのような場合、明示的な選択性推定を指定することでパフォーマンスを改善できることがあります。

オプティマイザは対象となる各テーブルについて、結果の一部となるローの数を推測します。条件の成功する確率がオプティマイザの推定とは異なることがあらかじめ判明している場合、ユーザ推定を明示的に探索条件として指定できます。

予測値はパーセントで表します。この値は、正の整数または小数です。

警告

進行ベースで使用される文には、できるかぎり明示的な推定を指定しないようにしてください。データが変更されると、明示的な推定が不正確になり、オプティマイザが誤って不適切なプランを選択することがあります。明示的な選択性推定を使用する場合は、数値が正確であることを確認してください。たとえば、0% または 100% の値を指定してインデックスを強制的に使用することはしないでください。

ユーザ推定を無効にするには、データベースオプション user_estimates を Off に設定します。user_estimates のデフォルト値は Override-Magic です。これは、オプティマイザが条件に MAGIC (デフォルト) 選択性値を使用する場合にのみ、ユーザ提供の選択性推定が使用されることを意味します。オプティマイザは、述部の選択性を正確に予測できない場合の最終手段として MAGIC 値を使用します。

例

次のクエリでは、ShipDate 値の 1% が 2001/06/30 より遅くなる予測値を出力します。

```
SELECT  ShipDate
   FROM  GROUPO.SalesOrderItems
WHERE ( ShipDate > '2001/06/30', 1 )
ORDER BY ShipDate DESC;
```

次のクエリでは、ローの 0.5% が条件を満たす予測値を出力します。

```
SELECT *
   FROM GROUPO.Customers c, GROUPO.SalesOrders o
WHERE (c.ID = o.CustomerID, 0.5);
```

小数を使用すると、ジョインや大きなテーブルのユーザ予測値はさらに正確になります。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE STATISTICS 文 [942 ページ]
FROM 句 [1112 ページ]

1.1.8 特別値

特別値は、式や、テーブル作成時のカラムのデフォルトに使用できます。

クエリに使用できる特別値もありますが、カラムのデフォルト値だけに使用できる特別値もあります。たとえば、AST USER、TIMESTAMP、および UTC TIMESTAMP は、デフォルト値だけに使用できます。

このセクションの内容:

CURRENT DATABASE 特別値 [85 ページ]
CURRENT DATABASE は、現在のデータベースの名前を返します。

CURRENT DATE 特別値 [85 ページ]
CURRENT DATE は、現在の年、月、日を返します。

CURRENT PUBLISHER 特別値 [87 ページ]
CURRENT PUBLISHER は、SQL Remote レプリケーション用データベースのパブリッシャユーザ ID を含む文字列を返します。

CURRENT REMOTE USER 特別値 [88 ページ]
現在の接続が SQL Remote の受信フェーズに属している場合、CURRENT REMOTE USER は、現在、この接続に適用されているメッセージを作成したリモートユーザのユーザ ID を返します。それ以外のすべての状況では、CURRENT REMOTE USER は NULL 値です。

CURRENT SERVER DATE 特別値 [88 ページ]
CURRENT DATE は、データベースサーバのタイムゾーン内の現在の年、月、日を返します。

CURRENT SERVER TIME 特別値 [89 ページ]
CURRENT SERVER TIME は、データベースサーバのタイムゾーン内の現在の時、分、秒 (小数位あり) を返します。

CURRENT SERVER TIMESTAMP 特別値 [90 ページ]
CURRENT TIMESTAMP は、CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。データベースサーバのタイムゾーン内の 年、月、日、時、分、秒、秒の小数位で構成されます。

CURRENT TIME 特別値 [90 ページ]
CURRENT TIME は現在の時、分、秒 (小数位あり) を返します。

CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT TIMESTAMP は、CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。年、月、日、時、分、秒、秒の小数位で構成されます。

CURRENT USER 特別値 [94 ページ]
CURRENT USER には、現在の接続のユーザ ID が含まれます。

CURRENT UTC TIMESTAMP 特別値 [95 ページ]
CURRENT UTC TIMESTAMP は、年、月、日、時、分、秒、秒の小数位、タイムゾーンで構成される協定世界時 (UTC: Coordinated Universal Time) を返します。

EXECUTING USER 特別値 [96 ページ]
現在有効なユーザを返す SQL 特別値。

INVOKING USER 特別値 [97 ページ]
現在のプロシージャを呼び出したユーザを返すか、プロシージャが実行されていない場合は現在ログインしているユーザを返す SQL 特別値。

LAST USER 特別値 [98 ページ]
LAST USER は、ローを最後に変更したユーザのユーザ ID です。

NULL 特別値 [99 ページ]
NULL 値は、不定または該当なしの値を指定します。

PROCEDURE OWNER 特別値 [102 ページ]
現在のプロシージャの所有者を返すか、プロシージャコンテキストの外部でクエリが実行された場合は NULL を返す SQL 特別値。

SESSION USER 特別値 [103 ページ]
現在ログインしているユーザを格納する SQL 特別値。

SQLCODE 特別値 [104 ページ]
SQLCODE は、最後に実行された SQL 文の処理を示します。

SQLSTATE 特別値 [105 ページ]
SQLSTATE は、最後に実行された SQL 文が成功、エラー、または警告条件になったかどうかを示します。

TIMESTAMP 特別値 [106 ページ]
TIMESTAMP デフォルト値は、テーブル内のローが最後に変更されたローカル日時の記録に使用します。

USER 特別値 [108 ページ]
USER には、現在の接続のユーザ ID が含まれます。

UTC TIMESTAMP 特別値 [109 ページ]
UTC TIMESTAMP デフォルト値は、テーブル内のローが最後に変更された協定世界時 (UTC: Coordinated Universal Time) の記録に使用します。

1.1.8.1 CURRENT DATABASE 特別値

CURRENT DATABASE は、現在のデータベースの名前を返します。

データ型

文字列

備考

UPDATE 操作時には、CURRENT DATABASE のデフォルト値の入ったカラムは変更されません。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

SQL 文の式 [32 ページ]

1.1.8.2 CURRENT DATE 特別値

CURRENT DATE は、現在の年、月、日を返します。

データ型

DATE

備考

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこの値が計算されます。

UPDATE 操作時には、CURRENT DATE のデフォルト値の入ったカラムは変更されません。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。この標準では、現在の日付を定義する特別登録を CURRENT_DATE と呼びます。このソフトウェアでは、CURRENT_DATE はサポートされていません。

例

オーストラリア東部標準時を作成します。

```
CREATE TIME ZONE NewSouthWales OFFSET '10:00'
STARTING 'Oct/Sun>=1' AT '2:00'
ENDING 'Apr/Sun>=1' AT '2:00';
```

デフォルトカラムを CURRENT DATE 特別値に設定したテーブルを作成します。他の 2 つのタイムゾーンで同じ値を挿入します。

```
CREATE OR REPLACE TABLE TEST( COL1 DATE DEFAULT CURRENT DATE,COL2 INT);
SET OPTION PUBLIC.time_zone='NewSouthWales';
INSERT INTO TEST(COL2) VALUES(1),(2);
SET OPTION PUBLIC.time_zone=;
INSERT INTO TEST(COL2) VALUES(3),(4);
SELECT * FROM TEST;
```

最初の 2 つの行は NewSouthWales タイムゾーンの現在の日付を使用して挿入され、最後の 2 つの行はサーバのタイムゾーンにおける現在の日付を使用して挿入されます。

関連情報

DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]

SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]

1.1.8.3 CURRENT PUBLISHER 特別値

CURRENT PUBLISHER は、SQL Remote レプリケーション用データベースのパブリッシャユーザ ID を含む文字列を返します。

データ型

文字列

備考

パブリッシャは PUBLIC.db_publisher オプションを使用して設定したり、GRANT PUBLISH 文や REVOKE PUBLISH 文を使用して設定します。

CURRENT PUBLISHER は文字データ型のカラムでデフォルト値として使用できます。

DEFAULT CURRENT PUBLISHER として定義されたカラムで更新または挿入操作を実行すると、カラムは CURRENT PUBLISHER の現在の値を使用して更新されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

GRANT PUBLISH 文 [SQL Remote] [1147 ページ]
REVOKE PUBLISH 文 [SQL Remote] [1276 ページ]

1.1.8.4 CURRENT REMOTE USER 特別値

現在の接続が SQL Remote の受信フェーズに属している場合、CURRENT REMOTE USER は、現在、この接続に適用されているメッセージを作成したリモートユーザのユーザ ID を返します。それ以外のすべての状況では、CURRENT REMOTE USER は NULL 値です。

データ型

文字列

備考

CURRENT REMOTE USER 特別値は、メッセージをデータベースに適用するときに、SQL Remote の受信フェーズによって設定されます。CURRENT REMOTE USER 特別値は、適用される操作が SQL Remote の受信フェーズによって適用されるかどうか、および、そのことが当てはまる場合に、適用される操作をどのリモートユーザが生成したかをトリガで判別する場合に、非常に役立ちます。

DEFAULT CURRENT REMOTE USER として定義されたカラムで更新または挿入操作を実行すると、カラムは CURRENT REMOTE USER の現在の値を使用して更新されます。

標準

ANSI/ISO SQL 標準

標準になし。

1.1.8.5 CURRENT SERVER DATE 特別値

CURRENT DATE は、データベースサーバのタイムゾーン内の現在の年、月、日を返します。

データ型

DATE

備考

UPDATE 操作時には、CURRENT SERVER DATE のデフォルト値の入ったカラムは変更されません。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。この標準では、現在の日付を定義する特別登録を CURRENT_SERVER_DATE と呼びます。このソフトウェアでは、CURRENT_SERVER_DATE はサポートされていません。

1.1.8.6 CURRENT SERVER TIME 特別値

CURRENT SERVER TIME は、データベースサーバのタイムゾーン内の現在の時、分、秒 (小数位あり) を返します。

データ型

TIME

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

UPDATE 操作時には、CURRENT SERVER TIME のデフォルト値の入ったカラムは変更されません。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。この標準では、現在の時刻を定義する特別登録を CURRENT_SERVER_TIME と呼びます。このソフトウェアでは、CURRENT_SERVER_TIME はサポートされていません。

1.1.8.7 CURRENT SERVER TIMESTAMP 特別値

CURRENT TIMESTAMP は、CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。データベースサーバのタイムゾーン内の 年、月、日、時、分、秒、秒の小数位で構成されます。

データ型

TIMESTAMP

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。この標準では、現在のタイムスタンプを定義する特別登録を CURRENT_TIMESTAMP と呼びます。

1.1.8.8 CURRENT TIME 特別値

CURRENT TIME は現在の時、分、秒 (小数位あり) を返します。

データ型

TIME

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこの値が計算されます。

UPDATE 操作時には、CURRENT TIME のデフォルト値の入ったカラムは変更されません。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。この標準では、現在の時刻を定義する特別登録を CURRENT_TIME と呼びます。このソフトウェアでは、CURRENT_TIME はサポートされていません。

例

オーストラリア東部標準時を作成します。

```
CREATE TIME ZONE NewSouthWales OFFSET '10:00'
STARTING 'Oct/Sun>=1' AT '2:00'
ENDING 'Apr/Sun>=1' AT '2:00';
```

デフォルトカラムを CURRENT TIME 特別値に設定したテーブルを作成します。他の 2 つのタイムゾーンで同じ値を挿入します。

```
CREATE OR REPLACE TABLE TEST( COL1 TIME DEFAULT CURRENT TIME,COL2 INT);
SET OPTION PUBLIC.time_zone='NewSouthWales';
INSERT INTO TEST(COL2) VALUES(1),(2);
SET OPTION PUBLIC.time_zone=;
INSERT INTO TEST(COL2) VALUES(3),(4);
SELECT * FROM TEST;
```

最初の 2 つの行は NewSouthWales タイムゾーンの現在の時間を使用して挿入され、最後の 2 つの行はサーバのタイムゾーンにおける現在の時間を使用して挿入されます。

関連情報

SQL 文の式 [32 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]

TIMESTAMP データ型 [168 ページ]
TIMESTAMP 特別値 [106 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.1.8.9 CURRENT TIMESTAMP 特別値

CURRENT TIMESTAMP は、CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。年、月、日、時、分、秒、秒の小数位で構成されます。

データ型

TIMESTAMP

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

DEFAULT TIMESTAMP とは異なり、DEFAULT CURRENT TIMESTAMP で宣言されたカラムにユニークな値を指定する必要はありません。ユニークな値にする必要がある場合、代わりに DEFAULT TIMESTAMP を使用してください。

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこの値が計算されます。

CURRENT TIMESTAMP が返す情報は、GETDATE 関数と NOW 関数が返す情報と同じです。

CURRENT_TIMESTAMP は、CURRENT TIMESTAMP と同じです。

> **注記**
>
> DEFAULT CURRENT TIMESTAMP と DEFAULT TIMESTAMP の主な違いは、DEFAULT CURRENT TIMESTAMP カラムは INSERT にのみ設定され (明示的に UPDATE に設定された場合を除く)、DEFAULT TIMESTAMP は INSERT と UPDATE の両方に設定されることです。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。この標準では、現在のタイムスタンプを定義する特別登録を CURRENT_TIMESTAMP と呼びます。

例

オーストラリア東部標準時を作成します。

```
CREATE TIME ZONE NewSouthWales OFFSET '10:00'
STARTING 'Oct/Sun>=1' AT '2:00'
ENDING 'Apr/Sun>=1' AT '2:00';
```

デフォルトカラムを CURRENT TIMESTAMP 特別値に設定したテーブルを作成します。他の 2 つのタイムゾーンで同じ値を挿入します。

```
CREATE OR REPLACE TABLE TEST( COL1 TIMESTAMP DEFAULT CURRENT TIMESTAMP,COL2 INT);
SET OPTION PUBLIC.time_zone='NewSouthWales';
INSERT INTO TEST(COL2) VALUES(1),(2);
SET OPTION PUBLIC.time_zone=;
INSERT INTO TEST(COL2) VALUES(3),(4);
SELECT * FROM TEST;
```

最初の 2 つの行は NewSouthWales タイムゾーンの現在の日時を使用して挿入され、最後の 2 つの行はサーバのタイムゾーンにおける現在の日時を使用して挿入されます。

関連情報

SQL 文の式 [32 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP 特別値 [106 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.1.8.10 CURRENT USER 特別値

CURRENT USER には、現在の接続のユーザ ID が含まれます。

データ型

文字列

備考

CURRENT USER は文字データ型のカラムでデフォルト値として使用できます。

UPDATE 時には、明示的に更新されないかぎり CURRENT USER のデフォルトが入ったカラムは変更されません。LAST USER のデフォルトは、ユーザによる更新の追跡に使用されることがあります。

CURRENT_USER は、CURRENT USER と同じです。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準内にありません。標準では、現在のユーザを定義する特別登録を CURRENT_USER と呼びます。

関連情報

SQL 文の式 [32 ページ]
LAST USER 特別値 [98 ページ]
USER 特別値 [108 ページ]

1.1.8.11 CURRENT UTC TIMESTAMP 特別値

CURRENT UTC TIMESTAMP は、年、月、日、時、分、秒、秒の小数位、タイムゾーンで構成される協定世界時 (UTC: Coordinated Universal Time) を返します。

データ型

TIMESTAMP WITH TIME ZONE

備考

この機能により、データが入力されたタイムゾーンに関係なく、入力データに一貫した時間基準を適用することができます。

UPDATE 操作時には、明示的に更新されないかぎり CURRENT UTC TIMESTAMP のデフォルト値が入ったカラムは変更されません。

UTC TIMESTAMP のデフォルトを使用して更新の UTC 時刻を追跡できます。

標準

ANSI/ISO SQL 標準

標準になし。ただし、TIMESTAMP WITH TIME ZONE データ型は、オプションの ANSI/ISO SQL 言語機能 F411 です。

関連情報

SQL 文の式 [32 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]

TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP 特別値 [106 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.1.8.12 EXECUTING USER 特別値

現在有効なユーザを返す SQL 特別値。

データ型

STRING

備考

EXECUTING USER、INVOKING USER、SESSION USER、および PROCEDURE OWNER を使用して、プロシージャとユーザ定義関数を実行できる (している) ユーザを確認します。特定のプロシージャの呼び出しがネストしているレイヤの数に応じて、および以前と現在のプロシージャが SQL SECURITY DEFINER と SQL SECURITY INVOKER のどちらであるかに基づいて、EXECUTING USER と INVOKING USER は変更を実行することができます (します)。

EXECUTING_USER は、EXECUTING USER と同義です。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

INVOKING USER 特別値 [97 ページ]
PROCEDURE OWNER 特別値 [102 ページ]
SESSION USER 特別値 [103 ページ]

1.1.8.13 INVOKING USER 特別値

現在のプロシージャを呼び出したユーザを返すか、プロシージャが実行されていない場合は現在ログインしているユーザを返す SQL 特別値。

データ型

STRING

備考

INVOKING USER、SESSION USER、EXECUTING USER、および PROCEDURE OWNER を使用して、プロシージャとユーザ定義関数を実行できる (している) ユーザを確認します。特定のプロシージャの呼び出しがネストしているレイヤの数に応じて、および以前と現在のプロシージャが SQL SECURITY DEFINER と SQL SECURITY INVOKER のどちらであるかに基づいて、INVOKING USER と EXECUTING USER は変更を実行することができます (します)。

INVOKING_USER は、INVOKING USER と同義です。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

EXECUTING USER 特別値 [96 ページ]
PROCEDURE OWNER 特別値 [102 ページ]
SESSION USER 特別値 [103 ページ]

1.1.8.14 LAST USER 特別値

LAST USER は、ローを最後に変更したユーザのユーザ ID です。

データ型

文字列

備考

LAST USER は文字データ型のカラムでデフォルト値として使用できます。

INSERT の場合、この定数は CURRENT USER と同じ効果があります。

UPDATE では、LAST USER のデフォルト値を持つカラムが明示的に変更されていなければ、現在のユーザ名に変更されます。

DEFAULT TIMESTAMP と結合すると、LAST USER のデフォルト値を使用して、ローを最後に変更したユーザと日時の両方を記録できます (ただし、別々のカラムに記録されます)。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE TABLE 文 [952 ページ]
CURRENT USER 特別値 [94 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
USER 特別値 [108 ページ]

1.1.8.15 NULL 特別値

NULL 値は、不定または該当なしの値を指定します。

構文

NULL

備考

NULL は、あらゆるデータ型の有効な値とは異なる特別値です。ただし、NULL 値はすべてのデータ型で使用できます。NULL は、情報が不足または不適切であることを表すために使用します。次に示すとおり、NULL が使用される 2 つのケースは、独立していて、性質が異なります。

状況	説明
欠落	フィールドに値はありますが、その値が不明です。
不適切	フィールドは、この特定のローには適していません。

SQL では、NOT NULL 制限を使用してカラムを作成できます。これらの特定のカラムに NULL を含めることはできません。

NULL 値によって、SQL に 3 値的論理の概念が導入されました。NULL 値をどのような値 (NULL 値を含む) や比較演算子を使用して比較しても、結果は "UNKNOWN" です。この場合、TRUE を返す唯一の探索条件は IS NULL 述部です。SQL では、WHERE 句の探索条件が TRUE と評価された場合のみ、ローが選択されます。UNKNOWN または FALSE と評価されたローは、選択されません。

NULL 値に対するカラム領域の使用量は、カラム当たり 1 ビットであり、領域は 8 ビット単位で割り付けられます。NULL のビット使用量は、NULL 値を許容するテーブルのカラム数に基づいて決められます。

IS [NOT] `truth-value` 句は、NULL 値があるローを選択するために使用します (`truth-value` は TRUE、FALSE、または UNKNOWN のいずれかです)。

カラム Salary に NULL が格納されている場合の例を次に示します。

条件	真理値	選択
Salary = NULL	UNKNOWN	NO
Salary <> NULL	UNKNOWN	NO
NOT (Salary = NULL)	UNKNOWN	NO
NOT (Salary <> NULL)	UNKNOWN	NO
Salary = 1000	UNKNOWN	NO
Salary IS NULL	TRUE	YES
Salary IS NOT NULL	FALSE	NO
Salary = `expression` IS UNKNOWN	TRUE	YES

2 つの異なるテーブルのカラムを比較する場合、同じ規則が適用されます。そのため、2 つのテーブルを結合すると、比較したカラムに NULL 値があるローは選択されません。

数値式で使用する場合も、NULL 値は特別な性質を持っています。NULL 値が含まれる数値式の結果は、すべて NULL 値になります。NULL 値を数値に加算しても結果は NULL 値であり、数値にはなりません。NULL を 0 として扱うには、ISNULL(expression, 0) 関数を使用します。

SQL クエリの作成で生じるエラーの多くは、NULL の性質によるものです。注意して、このような問題を避けるようにしてください。

セット演算子と DISTINCT 句

SQL では、探索条件内での NULL との比較の結果は UNKNOWN となります。ただし、2 つのローがそれぞれの重複であるかどうかを判別する場合、SQL は NULL を NULL と等しいものとして処理します。これらのセマンティックは、集合演算子 (UNION、INTERSECT、EXCEPT)、GROUP BY、WINDOW 句内 PARTITION、および SELECT DISTINCT に適用されます。

たとえば、テーブル T1 で、redundant というカラムのすべてのローに NULL が含まれる場合、次の文は 1 つのローを返します。

```
SELECT DISTINCT redundant FROM T1;
```

検索引数可能な探索条件 IS DISTINCT FROM および IS NOT DISTINCT FROM を使用することて、2 つの式が等しいかどうか、または両方の式が NULL かどうかを判別することもできます。

前提条件

データベースに接続されている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 言語

コア機能。

Transact-SQL

Adaptive Server Enterprise では、一部のコンテキストにおいて NULL 値との比較の処理が異なります。expression が等号または不等号を使用して変数または NULL リテラルと比較され、expression がベーステーブルまたはビューのカラムを参照する単純な式である場合、NULL = NULL が UNKNOWN ではなく TRUE を返す 2 値的論理を使用して

比較が実行されます。これらのセマンティックで考えられる比較とそれに相当する ANSI/ISO SQL 標準の比較のリストを以下に示します。

Transact-SQL の比較	ANSI/ISO SQL 標準
expression = NULL	expression IS NULL
expression != NULL	NOT (expression IS NULL)
expression = variable	expression = variable IS TRUE OR (expression IS NULL AND variable IS NULL)
expression != variable	expression != variable IS TRUE AND (NOT expression IS NULL OR NOT variable IS NULL)

SQL Anywhere では、ansinull オプションが OFF に設定されている場合、これらのセマンティックを実装して、Adaptive Server Enterprise の動作と一致させます。Open Client 接続と jConnect 接続では、ansinull オプションはデフォルトで OFF に設定されます。ANSI/ISO SQL 標準のセマンティックを保証するには、ansinull オプションを ON にリセットするか、等号比較の代わりに IS [NOT] NULL 述部を使用します。

SQL Anywhere のユニークインデックスは、エントリに NULL があってもよい点を除き、Adaptive Server Enterprise の場合と同じです。Adaptive Server Enterprise では、そのようなエントリをユニークインデックスに入れることはできません。

jConnect を使用する場合、tds_empty_string_is_null オプションは、空文字列が NULL 文字列または 1 つのブランク文字を含む文字列として返されるかどうかを制御できます。

例

次の INSERT 文は、Borrowed_book テーブルの date_returned カラムに NULL を挿入します。

```
INSERT INTO Borrowed_book ( date_borrowed, date_returned, book )
VALUES ( CURRENT DATE, NULL, '1234' );
```

関連情報

SQL 文の式 [32 ページ]
探索条件 [53 ページ]
IS DISTINCT FROM および IS NOT DISTINCT FROM 探索条件 [59 ページ]

1.1.8.16 PROCEDURE OWNER 特別値

現在のプロシージャの所有者を返すか、プロシージャコンテキストの外部でクエリが実行された場合は NULL を返す SQL 特別値。

データ型

STRING

備考

PROCEDURE OWNER、INVOKING USER、SESSION USER、および EXECUTING USER を使用して、プロシージャとユーザ定義関数を実行できる (している) ユーザを確認します。特定のプロシージャの呼び出しがネストしているレイヤの数に応じて、および以前と現在のプロシージャが SQL SECURITY DEFINER と SQL SECURITY INVOKER のどちらであるかに基づいて、EXECUTING USER と INVOKING USER は変更を実行することができます (します)。

PROCEDURE_OWNER は、PROCEDURE OWNER と同義です。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

EXECUTING USER 特別値 [96 ページ]
INVOKING USER 特別値 [97 ページ]
SESSION USER 特別値 [103 ページ]

1.1.8.17 SESSION USER 特別値

現在ログインしているユーザを格納する SQL 特別値。

データ型

STRING

備考

SESSION USER、INVOKING USER、EXECUTING USER、および PROCEDURE OWNER を使用して、プロシージャとユーザ定義関数を実行できる (している) ユーザを確認します。特定のプロシージャの呼び出しがネストしているレイヤの数に応じて、および以前と現在のプロシージャが SQL SECURITY DEFINER と SQL SECURITY INVOKER のどちらであるかに基づいて、INVOKING USER と EXECUTING USER は変更を実行することができます (します)。ただし、SESSION USER は常にログインユーザのままです。

SESSION_USER は SESSION USER と同義です。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

EXECUTING USER 特別値 [96 ページ]
INVOKING USER 特別値 [97 ページ]
PROCEDURE OWNER 特別値 [102 ページ]

1.1.8.18 SQLCODE 特別値

SQLCODE は、最後に実行された SQL 文の処理を示します。

データ型

符号付き INTEGER

備考

データベースサーバは、実行する各 SQL 文に対して、SQLSTATE と SQLCODE を設定します。SQLCODE は製品固有であり (たとえば、Mobile Link には固有の SQLCODE があります)、SQLSTATE についての追加情報を得る場合に使用できます。たとえば、100 以外の正の値は、製品固有の `warning` 条件を示します。負の値は、製品固有の `exception` 条件を示します。値 100 は、(たとえば、カーソルによってフェッチされた結果セットの最後で) "データがない" ことを示します。

SQLSTATE と SQLCODE は、各 SQLCODE が SQLSTATE に対応しており、各 SQLSTATE が 1 つ以上の SQLCODE に対応していることがあるという点で関連しています。

SQLCODE に関連付けられたエラー条件を返すには、ERRORMSG 関数を使用できます。

> **注記**
>
> SQLSTATE は、SQL 文の結果の優先ステータスインジケータです。

標準

ANSI/ISO SQL 標準

標準になし。SQLSTATE は、優先ステータスインジケータです。

関連情報

SQLSTATE 特別値 [105 ページ]
SQL 文の式 [32 ページ]
ERRORMSG 関数 [その他] [349 ページ]

1.1.8.19 SQLSTATE 特別値

SQLSTATE は、最後に実行された SQL 文が成功、エラー、または警告条件になったかどうかを示します。

データ型

文字列

備考

データベースサーバは、実行する各 SQL 文に対して、SQLSTATE と SQLCODE を設定します。SQLSTATE は、最後に実行された SQL 文が成功、警告、またはエラー条件になったかどうかを示す文字列です。

各 SQLSTATE は、すべてのプラットフォームに共通するエラーを表し、通常は製品共通の表現が含まれてます。SQLSTATE 値のフォーマットは、2 文字クラス値の後ろに 3 文字サブクラス値が続きます。クラス値とサブクラス値に関する SQLSTATE 適合のガイドラインは、ISO/ANSI SQL 標準に概説されています。

データベースサーバは、ISO/ANSI の SQLSTATE 規則に準拠しており、次の追加項目と例外項目があります。

クラスとサブクラス	条件
01WCx	文字セット変換に関する警告
38xxx	外部関数の例外
42Xxx	構文エラー: 式
42Rxx	構文エラー: 参照整合性 (たとえば、2 つ目のプライマリキーを作成しようとした場合)
42Wxx	構文エラー: 汎用
42Uxx	構文エラー: 重複したオブジェクト参照、定義されていないオブジェクト参照、またはあいまいなオブジェクト参照
42Zxx	アクセス違反
54Wxx	製品の制限超過
55Wxx	オブジェクトが操作を成功させるために必要なステータスにない
57xxx	リソース使用不可またはオペレータ介入
5Rxxx	SQL Remote のエラー
WBxxx	オンラインバックアップのエラー
WIxxx	データベースの内部エラー
WPxxx	プロシージャ、変数などでのエラー
WLxxx	ロードかアンロードまたはその両方のエラー

クラスとサブクラス	条件
WWxxx	SQL Anywhere 固有のその他のエラー/警告 (システム障害を含む)
WOxxx	リモートデータアクセス機能に関するエラー
WJxxx	JCS と JDBC 関連のエラー
WCxxx	文字変換のエラー
WXxxx	XML 関連のエラー
WTxxx	テスト関連のエラー

正常終了クラスは '00xxx' です (たとえば、'00000')。

SQLSTATE と SQLCODE は、各 SQLCODE が SQLSTATE に対応しており、各 SQLSTATE が 1 つ以上の SQLCODE に対応していることがあるという点で関連しています。

SQLSTATE に関連付けられたエラー条件を返すには、ERRORMSG 関数を使用できます。

標準

ANSI/ISO SQL 標準

値 '0' ～ '4' と 'A' ～ 'H' で始まる SQLSTATE クラス (最初の 2 文字) は、ANSI/ISO SQL 標準で定義されています。その他のクラスは実装依存です。同様に、値 '0' ～ '4' と 'A' ～ 'H' で始まるサブクラス値も、ANSI/ISO SQL 標準で定義されています。これらの範囲に含まれないサブクラス値は実装依存です。

関連情報

ERRORMSG 関数 [その他] [349 ページ]
SQLCODE 特別値 [104 ページ]
SQL 文の式 [32 ページ]

1.1.8.20 TIMESTAMP 特別値

TIMESTAMP デフォルト値は、テーブル内のローが最後に変更されたローカル日時の記録に使用します。

データ型

TIMESTAMP

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

カラムの宣言に DEFAULT TIMESTAMP が指定されている場合は、ローを挿入するとタイムスタンプのデフォルト値が割り付けられます。この値は、ローが更新されるたびに現在の日付と時刻に基づいて更新されます。

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこの値が計算されます。

DEFAULT TIMESTAMP で宣言されたカラムにはユニークな値が入ります。これにより、アプリケーションは、ほぼ同時に行われた同じローの更新を検出できます。現在のタイムスタンプ値が直前の値と同じ場合は、default_timestamp_increment オプションの値が加えられます。

default_timestamp_increment オプションを使用して、タイムスタンプ値を自動的にトランケートできます。これは、記録されるタイムスタンプ値の精度が低い他のデータベースソフトウェアとの互換性を維持する場合に便利です。

グローバル変数 @@dbts は、データベースの DEFAULT TIMESTAMP を使用して、いずれかのカラムに対して最後に生成された値に対応する TIMESTAMP 値を返します。データベースがシミュレートされたタイムゾーンを使用している場合、TIMESTAMP 値はそのタイムゾーンに相対したものになります。

注記

DEFAULT CURRENT TIMESTAMP と DEFAULT TIMESTAMP の主な違いは、DEFAULT CURRENT TIMESTAMP カラムは INSERT にのみ設定され (明示的に UPDATE に設定された場合を除く)、DEFAULT TIMESTAMP は INSERT と UPDATE の両方に設定されることです。

標準

ANSI/ISO SQL 標準

標準になし。

例

オーストラリア東部標準時を作成します。

```
CREATE TIME ZONE NewSouthWales OFFSET '10:00'
STARTING 'Oct/Sun>=1' AT '2:00'
ENDING 'Apr/Sun>=1' AT '2:00';
```

デフォルトカラムを CURRENT TIMESTAMP 特別値に設定したテーブルを作成します。他の 2 つのタイムゾーンで同じ値を挿入します。

```
CREATE OR REPLACE TABLE TEST( COL1 TIMESTAMP DEFAULT TIMESTAMP,COL2 INT);
SET OPTION PUBLIC.time_zone='NewSouthWales';
INSERT INTO TEST(COL2) VALUES(1),(2);
SET OPTION PUBLIC.time_zone=;
INSERT INTO TEST(COL2) VALUES(3),(4);
SELECT * FROM TEST;
```

最初の 2 つの行は NewSouthWales タイムゾーンの現在の日時を使用して挿入され、最後の 2 つの行はサーバのタイムゾーンにおける現在の日時を使用して挿入されます。

関連情報

CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.1.8.21 USER 特別値

USER には、現在の接続のユーザ ID が含まれます。

データ型

文字列

備考

USER は文字データ型のカラムでデフォルト値として使用できます。これは、CURRENT USER と同義です。

UPDATE 時には、明示的に更新されないかぎり USER のデフォルトが入ったカラムは変更されません。代わりに、LAST USER のデフォルトがユーザによる更新の追跡に使用されることがあります。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

SQL 文の式 [32 ページ]
CURRENT USER 特別値 [94 ページ]
LAST USER 特別値 [98 ページ]

1.1.8.22 UTC TIMESTAMP 特別値

UTC TIMESTAMP デフォルト値は、テーブル内のローが最後に変更された協定世界時 (UTC: Coordinated Universal Time) の記録に使用します。

データ型

TIMESTAMP WITH TIME ZONE

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

カラムの宣言に DEFAULT UTC TIMESTAMP が指定されている場合は、ローを挿入するとタイムスタンプのデフォルト値が割り付けられます。この値は、ローが更新されるたびに現在の UTC の日付と時刻に基づいて更新されます。

DEFAULT UTC TIMESTAMP で宣言されたカラムにはユニークな値が入ります。これにより、アプリケーションは、ほぼ同時に行われた同じローの更新を検出できます。現在の UTC タイムスタンプ値が直前の値と同じ場合は、default_timestamp_increment オプションの値が加えられます。

default_timestamp_increment オプションを使用して、UTC TIMESTAMP 値を自動的にトランケートできます。これは、記録されるタイムスタンプ値の精度が低い他のデータベースソフトウェアとの互換性を維持する場合に便利です。

> **注記**
>
> DEFAULT UTC TIMESTAMP は INSERT と UPDATE の両方で設定され、DEFAULT CURRENT UTC TIMESTAMP は INSERT で設定されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

SQL 文の式 [32 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.1.9 %TYPE および %ROWTYPE 属性

オブジェクトに対してデータ型を明示的に設定する以外にも、%TYPE および %ROWTYPE 属性を指定することによってデータ型を設定することもできます。

変数の作成または宣言時、値の変換時、テーブルの作成または変更時、およびプロシージャの作成時に、%TYPE および %ROWTYPE 属性を使用して、テーブル、ビュー、またはカーソルのカラムまたはローのデータ型に基づいてデータ型を定義します。%TYPE 属性はデータ型を、指定したオブジェクトのカラムのデータ型に設定します。一方、%ROWTYPE 属性はデータ型を、指定したオブジェクトのローのデータ型に設定します。

スキーマオブジェクトに対して %TYPE または %ROWTYPE を指定した場合、データベースサーバは、実際のデータ型情報をシステムテーブルから抽出します。たとえば、%TYPE 属性でテーブルカラムを指定する場合、データ型は ISYSTABCOL システムテーブルから抽出されます。

データ型が抽出され、オブジェクト (変数やカラムなど) が作成されたら、%TYPE および %ROWTYPE 属性で参照されるオブジェクトに対するこれ以上のリンクまたは依存性は存在しません。ただし、%TYPE および %ROWTYPE を使用してパラメータ

および戻り値を定義するプロシージャの場合、基礎となる参照先のオブジェクトが変更された場合は、プロシージャから異なる結果が戻される可能性があります。これは、プロシージャの作成時ではなく、プロシージャの実行時に %TYPE および %ROWTYPE が評価されるためです。

テーブルとビュー

%TYPE 属性を指定し、カラムのデータ型を、別のテーブルまたはビューのカラムのデータ型に設定します。次に例を示します。

- `myColumnName other-table-name.column-name%TYPE`

次の例の 2 番目の文は、myT2 テーブルを作成し、その myColumn カラムのデータ型を myT1 の last_name カラムのデータ型に設定します。null 入力可能性などの追加属性は適用されないため、myT2.myColumn には、myT1.last_name が持つ NOT NULL 制限と同じ制限は適用されません。

```
CREATE TABLE myT1
( first_name   CHAR(20),
   last_name   VARCHAR NOT NULL );
CREATE TABLE myT2
( myColumn myT1.last_name%TYPE );
```

プロシージャと関数

%TYPE または %ROWTYPE 属性を指定し、パラメータのデータ型を、指定したテーブルまたはビューのカラムまたはローのデータ型に設定します。

次の文は、プロシージャ DepartmentsCloseToCustomerLocation を作成し、その IN パラメータを Customers テーブル内の ID カラムのデータ型に設定します。

```
CREATE OR REPLACE PROCEDURE DepartmentsCloseToCustomerLocation( IN customer_ID
Customers.ID%TYPE )
  BEGIN
    DECLARE cust_rec Customers%ROWTYPE;
    SELECT City, State, Country
    INTO cust_rec.City, cust_rec.State, cust_rec.Country
    FROM Customers
    WHERE ID = customer_ID;
    SELECT Employees.Surname, Employees.GivenName, Departments.DepartmentName
    FROM Employees JOIN Departments
    ON Departments.DepartmentHeadID = Employees.EmployeeID
    WHERE Employees.City = cust_rec.City
    AND Employees.State = cust_rec.State
    AND Employees.Country = cust_rec.Country;
  END;
CALL DepartmentsCloseToCustomerLocation(158);
```

次の文は、fullname と呼ばれる関数を作成し、firstname および lastname パラメータのデータ型を Employees テーブルの Surname および Givenname カラムのデータ型に設定します。

```
CREATE OR REPLACE FUNCTION fullname(
   firstname Employees.Surname%TYPE,
   lastname Employees.GivenName%TYPE )
```

```
RETURNS LONG VARCHAR
BEGIN
   RETURN ( firstname || ' ' || lastname );
END;
SELECT fullname ( Surname, GivenName )FROM GROUPO.Employees;
```

値のキャストおよび変換

値を別のデータベースオブジェクトのデータ型にキャストまたは変換する場合は、%TYPE 属性を指定します。

次の文は、Employees テーブルの BirthDate カラム (DATE データ型) に定義されているデータ型に値をキャストします。

```
SELECT CAST ( '1966-10-30' AS Employees.BirthDate%TYPE );
```

ドメイン

%TYPE 属性を指定し、ドメインのデータ型を、指定したテーブルまたはビューのカラムのデータ型に設定します。

次の例では、次の一連の文のうち 2 番目の 2 つの CREATE DOMAIN 文は、Customers テーブルの Surname および GivenName カラムのデータ型に基づいてドメインを作成します。

```
CREATE DOMAIN identifier UNSIGNED INT
DEFAULT AUTOINCREMENT;
CREATE DOMAIN customers_surname Customers.Surname%TYPE;
CREATE DOMAIN customers_givenname Customers.GivenName%TYPE;

CREATE TABLE Customers3 (
   ID identifier PRIMARY KEY,
   SurName customers_surname,
   GivenName customers_givenname
);
```

変数

%TYPE 属性を指定し、変数のデータ型を、指定したテーブル、ビュー、またはカーソルのカラムのデータ型に設定します。%TYPE を使用する場合、参照先のオブジェクトからはデータ型のみが抽出されます。デフォルト値、制約、および NULL の使用の可否などの他の属性は含まれず、個別に指定する必要があります。基礎となるテーブルスキーマの変更に応じてアプリケーションを調整できるようにするには、%TYPE 属性を使用して、カラムデータと同じ型を使用して変数を宣言します。

次の例では、ProductID という新しい変数を作成し、%TYPE 属性を使用してそのデータ型を、Products テーブルの ID カラムのデータ型に設定します。

```
CREATE VARIABLE ProductID Products.ID%TYPE;
```

%ROWTYPE 属性を指定し、一連のカラムのデータ型を、指定したテーブル、ビュー、またはカーソルのローのデータ型に設定します。たとえば、%ROWTYPE 属性を使用して、ローまたは配列値を格納できる変数を定義します。

%ROWTYPE を指定する場合、デフォルト値、制約、および NULL の使用の可否などの他の属性は抽出に含まれません。

次の例では、%ROWTYPE 属性を使用して変数 @a_product を作成し、Products テーブルから変数にデータを挿入します。

```
CREATE OR REPLACE PROCEDURE CheckStock (
    IN @id Products.ID%TYPE
)
   BEGIN
    DECLARE @a_product Products%ROWTYPE;
    SET (@a_product).ID = 200;
   END;
```

また、次の例の 2 番目の DECLARE 文 に示すように、%TYPE 属性を使用して、変数のデータ型を別の変数の型に設定することもできます。

```
DECLARE cust_rec Customers%ROWTYPE;
DECLARE cust_rec2 cust_rec%TYPE;
```

このセクションの内容:

1.1.9.1 %TYPE 属性構文

変数の作成または宣言時、またはテーブル、ビュー、プロシージャ、および関数の作成または変更時に、データ型を指定したオブジェクトまたは変数内のカラムのデータ型に設定します。また、データ型間のキャストに使用することもできます。

構文

```
type-source%TYPE
| TYPE OF ( type-source )
type-source :
   [ owner. ]{ table-name
   | view-name }.column-name
   | variable-name
   | variable-name.field-name
```

パラメータ

table-name

テーブルの名前。

view-name

有効なビュー (マテリアライズドビューを含む) の名前。マテリアライズドビューの初期化も行う必要があります。

variable-name

変数の名前。

column-name

カラムの名前。

備考

プロシージャ (パラメータおよび戻り値)、テーブル、ビュー、およびドメインを作成または変更する場合、%TYPE 仕様内で参照されるオブジェクトは永続オブジェクトである必要があります。変数、カーソル、またはテンポラリテーブルなどのテンポラリオブジェクトを参照すると、エラーが返されます。

%TYPE が IS OF 検索式、FROM 句の WITH `hint` 式、または CAST または CONVERT 関数に指定されている場合、参照される項目は永続オブジェクトである必要があります。相関名または抽出テーブルを指定すると、エラーが返されます。

%TYPE が指定されている場合、デフォルト値、制約、および NULL の使用の可否などの他の属性は継承される定義の一部ではなく、個別に指定する必要があります。

変数を定義または宣言する場合、`type-source` の識別子部分が次の 1 つである場合、識別子部分は引用符で囲む必要があります。

- IN
- OUT
- INOUT
- DYNAMIC
- SCROLL
- NO
- INSENSITIVE
- SENSITIVE
- TIMESTAMP
- # で始まる名前

たとえば、次の文は、DYNAMIC と呼ばれる変数を宣言します。次に、var1 と呼ばれる別の変数を宣言し、そのデータ型を DYNAMIC (INT) のデータ型に設定します。DYNAMIC は引用符で囲む必要があるキーワードの 1 つであるため、引用符で囲む必要があるため、前後に引用符が配置されます。

```
BEGIN
    DECLARE dynamic INT;
    DECLARE var1 "DYNAMIC"%TYPE;
    SET var1 = 1;
    MESSAGE var1;
END
```

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例に加えて、%TYPE 属性の指定をサポートしている SQL 文および関数のマニュアルにも例があります。

次の文は、プロシージャ DepartmentsCloseToCustomerLocation を作成し、%TYPE 属性を使用して、その IN パラメータを Customers テーブル内の ID カラムのデータ型に設定します。

```
CREATE OR REPLACE PROCEDURE DepartmentsCloseToCustomerLocation( IN customer_ID
Customers.ID%TYPE )
  BEGIN
    DECLARE cust_rec Customers%ROWTYPE;
    SELECT City, State, Country
    INTO cust_rec.City, cust_rec.State, cust_rec.Country
    FROM Customers
    WHERE ID = customer_ID;
    SELECT Employees.Surname, Employees.GivenName, Departments.DepartmentName
    FROM Employees JOIN Departments
    ON Departments.DepartmentHeadID = Employees.EmployeeID
    WHERE Employees.City = cust_rec.City
    AND Employees.State = cust_rec.State
    AND Employees.Country = cust_rec.Country;
  END;
CALL DepartmentsCloseToCustomerLocation(158);
```

次の文は、Employees テーブルの BirthDate カラム (DATE データ型) に定義されているデータ型に値をキャストします。

```
SELECT CAST ( '1966-10-30' AS Employees.BirthDate%TYPE );
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
DECLARE 文 [1010 ページ]

1.1.9.2 %ROWTYPE 属性構文

データ型を、指定したテーブル、ビュー、テーブル参照変数、またはカーソル内のローの複合データ型に設定します。

構文

```
rowtype-source%ROWTYPE
| ROWTYPE OF ( rowtype-source )
rowtype-source :
   [ owner. ]{ table-name | view-name }
   | cursor-name
   | TABLE REF ( table-reference-variable )
```

パラメータ

table-name

テーブルの名前。

table-name を指定する場合、%ROWTYPE 変数のデータ型は、table-name 内のカラムのデータ型で構成されます。

view-name

有効なビュー (マテリアライズドビューを含む) の名前。マテリアライズドビューの初期化も行う必要があります。

view-name を指定する場合、%ROWTYPE 変数のデータ型は、view-name 内のカラムのデータ型で構成されます。

cursor-name

カーソルの名前。

cursor-name を指定する場合、%ROWTYPE 変数のデータ型は、カーソルの select 項目のデータ型で構成されます。

table-reference-variable

テーブル参照変数の名前。

テーブル参照変数を指定する場合、%ROWTYPE 変数のデータ型は、table-reference-variable で参照されるテーブルのカラムのデータ型で構成されます。

備考

%ROWTYPE 変数を作成する場合、デフォルト値、制約、および NULL の使用の可否などの他の属性は継承される定義の一部ではなく、個別に指定する必要があります。

テーブル、ビュー、またはカーソル参照を使用して **%ROWTYPE** を指定する際の制限

プロシージャ、ビュー、およびドメインを作成または変更する場合、%ROWTYPE 仕様内で参照されるオブジェクトは永続オブジェクトである必要があります。テンポラリテーブルを参照すると、エラーが返されます。

ロー変数を宣言するときに、%ROWTYPE 構成体の引数がまだオープンされていないカーソルである場合、基礎となるオブジェクトが変更されている場合、カーソルのスキーマがオープン時に異なる可能性があります。すでにオープンしているカーソルに基づいてロー変数を宣言する方が安全です。

%ROWTYPE が IS OF 検索式、FROM 句の WITH hint 式、または CAST または CONVERT 関数に指定されている場合、参照される項目は永続オブジェクトである必要があります。テンポラリテーブル、相関名、または抽出テーブルを指定すると、エラーが返されます。

rowtype-source がカーソルを参照する場合、カーソル内の項目の名前 (カラム名など) は簡易名またはエイリアスである必要があります。カーソル内の select list 項目を正常に抽出できない場合、エラーが返されます。

変数を定義または宣言する場合、rowtype-source の識別子部分が次の 1 つである場合、識別子部分は引用符で囲む必要があります。

- IN
- OUT
- INOUT
- DYNAMIC
- SCROLL
- NO
- INSENSITIVE
- SENSITIVE
- TIMESTAMP
- 先頭に # が付く識別子

テーブル参照変数を使用して **%ROWTYPE** を指定する際の制限 **(TABLE REF (table-reference-variable) %ROWTYPE):**

プロシージャ、ビュー、およびドメインを作成または変更する場合、テーブル参照変数を使用した %ROWTYPE の指定はサポートされていません。同様に、IS OF 検索式、FROM 句の WITH hint 式、または CAST または CONVERT 関数でも、テーブル参照変数を使用した %ROWTYPE の指定はサポートされていません。

rowtype-source がテーブル参照変数を参照する場合、%ROWTYPE の処理時にはテーブル参照変数がすでに初期化されている必要があります。バッチまたはプロシージャ内の文で TABLE REF (table-reference-variable) %ROWTYPE が使用される場合、テーブル参照変数に値が割り当てられたかテーブル参照変数がパラメータとして渡された後に、文を別の BEGIN...END ブロック内でネストする必要があります。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例に加えて、%ROWTYPE 属性の指定をサポートしている SQL 文および関数のマニュアルにも例があります。

次の例では、ItemsForSale という新しい変数を作成し、%ROWTYPE 属性を使用してそのデータ型を、Products テーブルに定義されているカラムで構成される複合データ型に設定します。

```
CREATE VARIABLE ItemsForSale Products%ROWTYPE;
```

次の文は、cust_rec という変数を宣言し、そのデータ型を Customers テーブルのローの複合データ型に設定します。

```
CREATE OR REPLACE PROCEDURE DepartmentsCloseToCustomerLocation( IN customer_ID
Customers.ID%TYPE )
  BEGIN
    DECLARE cust_rec Customers%ROWTYPE;
    SELECT City, State, Country
    INTO cust_rec.City, cust_rec.State, cust_rec.Country
    FROM Customers
    WHERE ID = customer_ID;
    SELECT Employees.Surname, Employees.GivenName, Departments.DepartmentName
    FROM Employees JOIN Departments
    ON Departments.DepartmentHeadID = Employees.EmployeeID
    WHERE Employees.City = cust_rec.City
    AND Employees.State = cust_rec.State
    AND Employees.Country = cust_rec.Country;
  END;
CALL DepartmentsCloseToCustomerLocation(158);
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
DECLARE 文 [1010 ページ]

1.1.10 SQL 変数

サポートされる変数は、スコープ (接続、データベース、グローバル) によってグループ化できます。

変数が作成されると、デフォルトが指定されていない限り、初期値は NULL に設定されます。この値は、SET 文、UPDATE 文、または SELECT 文を INTO 句とともに使用することにより、後で変更できます。

変数は COMMIT または ROLLBACK 文の影響を受けません。

接続スコープ変数

接続スコープ変数は、接続のコンテキストで設定および使用されます。これらの変数は、他の接続では使用できません。接続スコープ変数には 2 種類あります。すなわち、**connection-level** と **local** (**declared** とも呼ばれます) です。また、TABLE REF 型の接続スコープ変数を作成してテーブルに対する参照を保持することもできます。これらは、テーブル参照変数と呼ばれます。

接続レベル変数

接続レベル変数は、CREATE VARIABLE 文を使用して作成されます。通常、接続によって実行される任意のプロシージャで値を使用可能にするために使用されます。

接続レベル変数は、接続の間、または DROP VARIABLE 文を使用して変数が明示的に削除されるまでの間のみ保持されます。

ローカル (宣言された) 変数

ローカル変数は、BEGIN...END ブロック内で DECLARE 文を使用して作成されます。通常、ローカル変数が宣言されたものと同じ複合文内で値を格納および変更するために使用されます。ローカル変数値は、BEGIN...END ブロックのコンテキストの外部では使用できません。

ローカル変数は、宣言された BEGIN...END ブロックの実行中のみ保持されます。ローカル変数は削除することもできます。

データベーススコープ変数

データベーススコープ変数は、(接続の代わりに) データベースのコンテキスト内で使用されます。これは、接続間で値を共有する優れた方法です。使用目的は、変更頻度が低い小さな共有値を格納することです。変更頻度が高い値を格納することは、アプリケーションのパフォーマンスに影響を与える可能性があるため、お奨めしません。データベーススコープ変数の初期値は、データベースが再起動した後も保持されます (つまり、これらの初期値に対する変更は、データベースの再起動後は保持されません)。データベーススコープ変数は接続スコープ変数やグローバル変数と同じ方法で使用できますが、データ型 ROW、ARRAY、または TABLE REF を使用して定義することはできません。

ユーザが所有するデータベーススコープ変数

ユーザがデータベーススコープ変数を所有している場合、このユーザのみがこの変数から選択したりこの変更を更新できます。これらの処理は、接続とは関係なく実行できます。

また、データベーススコープ変数はロールが所有することもできます。ただし、ロールが所有するデータベーススコープ変数には、ストアドプロシージャ、ユーザ定義関数、およびロールが所有するイベントを介してのみアクセスできます。

PUBLIC が所有するデータベーススコープ変数

PUBLIC が所有するデータベーススコープ変数は、すべてのユーザおよび接続が使用できます。ただし、これは、ユーザが正しいシステム権限を持つことが前提です。

データベーススコープ変数をアクセスまたは管理するには、変数を所有する主体 (自分、別のユーザ、または PUBLIC) によって異なります。次の表に、データベーススコープ変数のアクセスまたは管理に必要な権限をまとめます。

表 1: データベーススコープ変数の管理に必要な権限

アクション	所有者	必要な権限
データベーススコープ変数の作成	自分	CREATE DATABASE VARIABLE または MANAGE ANY DATABASE VARIABLE
データベーススコープ変数の作成	別のユーザ	MANAGE ANY DATABASE VARIABLE
データベーススコープ変数の作成	PUBLIC	MANAGE ANY DATABASE VARIABLE

アクション	所有者	必要な権限
データベーススコープ変数の更新	自分	不要
データベーススコープ変数の更新	別のユーザ	不可能
データベーススコープ変数の更新	PUBLIC	UPDATE PUBLIC DATABASE VARIABLE
データベーススコープ変数からの選択	自分	なし
データベーススコープ変数からの選択	別のユーザ	不可能
データベーススコープ変数からの選択	PUBLIC	SELECT PUBLIC DATABASE VARIABLE
データベーススコープ変数の削除	自分	不要
データベーススコープ変数の削除	別のユーザ	MANAGE ANY DATABASE VARIABLE
データベーススコープ変数の削除	PUBLIC	MANAGE ANY DATABASE VARIABLE

グローバル変数

グローバル変数は、データベースのコンテキストで使用されます。ただし、データベースサーバのみが設定することができます。グローバル変数を直接設定することはできませんが、一部のグローバル変数はユーザアクティビティに応じて間接的に設定されます。たとえば、一部のグローバル変数 (@@identity など) は、接続に固有の情報を保持します。その他の変数 (@@connections など) は、すべての接続に共通の値を保持します。

グローバル変数は、名前の先頭に付けられた 2 つのアットマーク (@) で、その他の変数と視覚的に区別されます。たとえば、@@error や @@rowcount はグローバル変数です。

同じ名前を持つ変数とエイリアス

同じ名前を持つ変数とエイリアスが含まれる文を使用することは可能です。参照の解決方法を分かりやすく理解できるように、識別子を処理する際にデータベースサーバが辿る順序を次に示します。

1. クエリの SELECT リストで指定した任意のエイリアスと一致。
2. 任意の参照先テーブルのカラム名と一致。
3. 名前が変数であると仮定。

標準

ANSI/ISO SQL 標準

SQL ストアドプロシージャまたは関数内で DECLARE 文を使用して宣言される変数は、ANSI/ISO SQL 標準では SQL 言語機能 P002 の "Computational completeness" としてサポートされています。CREATE VARIABLE、DROP VARIABLE、およびグローバル変数は、ANSI/ISO SQL 標準内にありません。

このセクションの内容:

@@identity 変数には、現在の接続によって IDENTITY カラム、DEFAULT AUTOINCREMENT カラム、または DEFAULT GLOBAL AUTOINCREMENT カラムに挿入された最新の値が設定されます。最新の挿入が、このようなカラムがないテーブルに対して行われた場合は 0 になります。

関連情報

CREATE VARIABLE 文 [992 ページ]
DROP VARIABLE 文 [1085 ページ]
SET 文 [1325 ページ]
DECLARE 文 [1010 ページ]
UPDATE 文 [1391 ページ]
SYSDATABASEVARIABLE システムビュー [1859 ページ]

1.1.10.1 @@identity グローバル変数

@@identity 変数には、現在の接続によって IDENTITY カラム、DEFAULT AUTOINCREMENT カラム、または DEFAULT GLOBAL AUTOINCREMENT カラムに挿入された最新の値が設定されます。最新の挿入が、このようなカラムがないテーブルに対して行われた場合は 0 になります。

@@identity の値は、接続に固有です。文が複数のローを挿入した場合、@@identity には最後に挿入されたローの IDENTITY 値が反映されます。文が IDENTITY カラムがないテーブルに影響を及ぼすと、@@identity に 0 が設定されます。

INSERT または SELECT INTO 文の実行に失敗したり、@@identity の値が保持されるトランザクションのロールバックがあっても、@@identity の値には影響を及ぼしません。挿入された文のコミットが失敗しても、@@identity には IDENTITY カラムに最後に挿入された値が保持されます。

@@identity とトリガ

挿入の実行によって、参照整合性アクションが起きたり、トリガが起動されたりすると、@@identity はスタックのように動作します。たとえば、テーブル T1 (IDENTITY カラムまたは AUTOINCREMENT カラムがあるテーブル) に対する挿入によってトリガが起動し、テーブル T2 (IDENTITY カラムまたは AUTOINCREMENT カラムがあるテーブル) にローが 1 つ挿入された場合、挿入を実行したアプリケーションやプロシージャには、T1 に挿入された値が返されます。トリガ内では、T2 への挿入前には T1 の値が、挿入後には T2 の値が @@identity に入ります。トリガが両方の値にアクセスする場合、トリガはそれらの値をローカル変数にコピーできます。

標準

ANSI/ISO SQL 標準

グローバル変数は標準にありません。

1.1.11 コメント

コメントは、SQL 文または文ブロックに説明テキストを付加するために使用します。データベースサーバは、コメントを実行しません。

次のコメントインジケータがサポートされています。

-- (二重ハイフン)

データベースサーバは、この行の残りの文字を無視します。これは、ANSI/ISO SQL 標準のコメントインジケータです。

Interactive SQL や *SQL Central* のプロシージャとファンクションウィンドウの *SQL* タブで、テキストを選択して Ctrl + マイナス記号 (-) キーを押すと、このコメントインジケータを追加したり削除することができます。

SQL コメントインジケータが、選択したテキストの各行の先頭に追加されます。テキストが選択されていない場合、コメントインジケーターは現在の行の先頭に追加されます。

// (二重スラッシュ)

二重スラッシュは、二重ハイフンと同じ意味です。

Interactive SQL や *SQL Central* のプロシージャとファンクションウィンドウの *SQL* タブで、テキストを選択して Ctrl + スラッシュ (/) キーを押すと、このコメントインジケータを追加したり削除することができます。

SQL コメントインジケータが、選択したテキストの各行の先頭に追加されます。テキストが選択されていない場合、コメントインジケーターは現在の行の先頭に追加されます。

/* ...*/ (スラッシュ ― アスタリスク)

2 つのコメントマーカの間にある文字は、すべて無視されます。2 つのコメントマーカは、同じ行にあっても、別の行にあってもかまいません。このスタイルで示されたコメントはネストできますが、ネストされたコメントはバランスが取れている必要があります。コメントブロックに挿入されるすべてのコメントには、開始コメントマーカの単一インスタンスを含めないでください。このスタイルは、**C** スタイルコメントとも呼ばれます。

例

次に、二重ハイフンのコメントの使用例を示します。

```
CREATE FUNCTION fullname ( firstname CHAR(30),
         lastname CHAR(30))
RETURNS CHAR(61)
-- fullname concatenates the firstname and lastname
-- arguments with a single space between.
BEGIN
   DECLARE name CHAR(61);
   SET name = firstname || ' ' || lastname;
   RETURN ( name );
END;
```

次に、C スタイルコメントの使用例を示します。

```
/* Lists the names and employee IDs of employees
   who work in the sales department. */
CREATE VIEW SalesEmployees AS
   SELECT EmployeeID, Surname, GivenName
```

```
    FROM GROUPO.Employees
    WHERE DepartmentID = 200;
```

標準

ANSI/ISO SQL 標準

コメントでの二重ハイフンの使用は、ANSI/ISO SQL 標準のコア機能です。C スタイルの囲む形式のコメント (/* ...*/) は、SQL 言語機能 T351 です。二重スラッシュのコメント (//) は ソフトウェアではサポートされていますが、標準にありません。

1.1.12 名前付きパラメータ

CALL 文、EXECUTE 文 (Transact-SQL)、DML 文の FROM 句、および TRIGGER EVENT 文から参照される関数およびプロシージャは、位置パラメータと名前付きパラメータをサポートしています。使用可能なパラメータの任意のサブセットを任意の順序で指定できます。

名前付きパラメータは、CALL 文での場合を除いて関数とともには使用できません。名前付きパラメータは、ABS、COMPRESS、DAYNAME などの組み込み関数とともには使用できません。次の名前付きパラメータの構文がサポートされています。

- `parameter-name = parameter-value`
- `parameter-name => parameter-value`

例

次の例では = を使用して名前付きパラメータを指定しています。

```
CALL sa_conn_properties( connidparm = 1 );
```

次の例では、システムプロシージャ sp_remote_exported_keys を呼び出すときに名前付きパラメータを使用し、リモートサーバ RemoteSA 上で HR が所有する Employees テーブルの外部キーの関係に関する情報を返します。

```
CALL sp_remote_exported_keys(
     @server_name => 'RemoteSA',
     @table_owner => 'HR',
     @table_name => 'Employees' );
```

次の例では、TRIGGER EVENT 文で名前付きパラメータを使用します。

```
CREATE EVENT ev_TimePlace
HANDLER BEGIN
  MESSAGE 'ev_TimePlace - was triggered at ' ||  event_parameter( 'what_time' )
      || ' in ' || event_parameter( 'what_place' );
END;
TRIGGER EVENT ev_TimePlace( what_time => string( current timestamp ), what_place
=> 'Waterloo' );
```

名前付きパラメータは、CALL 文での場合を除いて関数とともには使用できません。次はその例です。

```
CREATE OR REPLACE FUNCTION PLUS( val1 INTEGER DEFAULT 0, val2 INTEGER DEFAULT 0 )
RETURNS INTEGER
BEGIN
    RETURN val1 + val2;
END
CREATE VARIABLE rslt INTEGER;
rslt = CALL PLUS( val1=1, val2=99 );
SELECT rslt;
```

名前付きパラメータは、一般的な式内の関数とともには使用できません。次に、使用する必要がある構文の例を示します。

```
SELECT PLUS( 1, 99 );
```

標準

ANSI/ISO SQL 標準

= 演算子は標準にありません。

関連情報

CALL 文 [756 ページ]
EXECUTE IMMEDIATE 文 [SP] [1090 ページ]
EXECUTE 文 [ESQL] [1093 ページ]
EXECUTE 文 [T-SQL] [1096 ページ]
FROM 句 [1112 ページ]
TRIGGER EVENT 文 [1365 ページ]

1.2 SQL データ型

このソフトウェアでは、多くの SQL データ型がサポートされています。

このセクションの内容:

ビット配列データ型 [151 ページ]
ビット配列は文字列に似ていますが、使用される要素は文字ではなくビットデータ (0 と 1) です。通常、ビット配列は、ブール値の文字列を保持するのに使用されます。

日付と時刻データ型 [153 ページ]
日付の値は 4 桁の西暦で出力でき、日付の内部記憶領域には、年の値の世紀にあたる部分が常に明示的に含まれています。

バイナリデータ型 [172 ページ]
バイナリデータ型は、データベースに認識されないイメージやその他の種類の情報を含めたバイナリデータを格納します。

複合データ型 ROW および ARRAY [178 ページ]
複合データ型は、0 個以上の要素で構成される値であり、要素ごとに特定のデータ型の値が含まれます。現在のところ、複合データ型 ROW および ARRAY のみがサポートされています。

TABLE REF データ型 [180 ページ]
TABLE REF データ型は、ベーステーブル、テンポラリテーブル、またはビューへの参照を格納します。このデータ型は、接続スコープ変数でのみ使用できます。

空間データ型 [184 ページ]
多くの空間データ型がサポートされます。これらのデータ型のマニュアルは、空間 SQL API のマニュアルと同じ場所にあります。

ドメイン [184 ページ]
ドメインは、適切な位置に精度や小数点以下の桁数を含み、さらにオプションとしてデフォルト値や CHECK 条件などを含んでいる、組み込みデータ型のエイリアスです。ドメインには、通貨データ型のように事前に定義されたものもありますが、ユーザが独自のドメインを追加することもできます。

データ型の比較 [185 ページ]
データ型の異なる引数の間で比較 (= など) を行う場合、1 つのデータ型で比較操作ができるように、1 つ以上の引数を変換します。

データ型変換 [192 ページ]
型の変換は自動的に発生することもあれば、CAST または CONVERT 関数を使って明示的に型変換が要求されることもあります。次の関数を使うことによっても、強制的に型変換を実行できます。

1.2.1 文字データ型

文字データ型は、文字、数字、記号などの文字列を格納します。

何種類かの文字データ型と、それらのデータ型を使用して定義されたいくつかのドメインがあります。

CHAR、VARCHAR、LONG VARCHAR

シングルバイトまたはマルチバイトの文字セットに格納される文字データ。多くの場合、データベースに格納されるプライマリ言語に最も対応するデータ型が選択されます。

NCHAR、NVARCHAR、LONG NVARCHAR

UTF-8 Unicode のエンコーディングに格納される文字データ。データベースに格納されているプライマリ言語に関係なく、これらのデータ型を使用してすべての Unicode コードポイントを格納できます。

TEXT、UNIQUEIDENTIFIERSTR、XML

他の文字データ型に基づくドメイン。

Ultra Light: Ultra Light は、シングルバイトまたはマルチバイトの文字セットに格納される CHAR、VARCHAR、LONG VARCHAR データ型をサポートしており、多くの場合、データベースに格納されるプライマリ言語に最も対応するデータ型が選択されます。

格納

すべての文字データ値は同じ方法で格納されます。デフォルトでは、1 つの値に最大で 128 バイトを格納できます。値が 128 バイトを超える場合、4 バイトのプレフィクスがデータベースページに格納され、全体の値は他の 1 つまたは複数のデータベースページに格納されます。このデフォルトサイズを制御するには、CREATE TABLE 文の INLINE 句と PREFIX 句を使用します。

Ultra Light: 固定文字型 (VARCHAR など) はローに埋め込まれ、LONG 文字型 (LONG VARCHAR など) は個別に格納されます。大きな固定型のカラム多数で構成されるテーブルを作成するときは、ページサイズを考慮します。1 行全体が 1 ページに収まる必要があります。また、固定文字カラム型は 1 行で格納します。たとえば、ページサイズを 1000 にして作成されたデータベースでは、1000 よりも大きな文字値はページに収まらないため保持できません。

このセクションの内容:

関連情報

CREATE TABLE 文 [952 ページ]

1.2.1.1 CHAR データ型

CHAR データ型は、32767 バイトまでの文字データを格納します。

構文

```
CHAR [ ( max-length [ CHAR | CHARACTER ] ) ]
```

Ultra Light:

```
CHAR [ ( max-length) ]
```

パラメータ

max-length

文字列の最大長。バイト長のセマンティックを使用する場合 (長さの部分に CHAR または CHARACTER を指定しない場合)、長さはバイト単位になります。また、範囲は 1 ～ 32767 にします。長さを指定しない場合、値は 1 になります。

文字長のセマンティックを使用する場合 (長さの部分に CHAR または CHARACTER を指定する場合)、長さは文字単位になります。また、`max-length` を指定します。`max-length` は最大 32767 文字までです。

備考

マルチバイト文字は CHAR 型に格納できますが、文字長のセマンティックを使用していなければ、宣言される長さは文字数ではなくバイト数です。

CHAR は CHARACTER と指定することもできます。どの構文を使用する場合でも、データ型は CHAR と記述されます。

セマンティック上、CHAR は VARCHAR と同じですが、型は異なります。CHAR は可変長型です。他のリレーショナルデータベース管理システムでは、CHAR は固定長型であり、データには `max-length` バイトまでブランクが埋め込まれて格納されます。SQL Anywhere では格納される文字データにブランクを埋め込みません。

カラムがどのように記述されるかは、クライアントインタフェース、使用される文字セット、および文字長セマンティックが使用されるかどうかによって異なります。たとえば、Embedded SQL では、記述される長さはクライアント文字セットの最大バイト数となります。記述される長さが 32767 バイトを超える場合、カラムは DT_LONGVARCHAR 型として記述されます。次の表は、Embedded SQL の例と、DESCRIBE を実行した場合に返される結果を示します。

記述される型	データベースの文字セット	クライアントの文字セット	DESCRIBE の結果
CHAR(10)	Windows-1252	Windows-1252	DT_FIXCHAR 長さ 10
CHAR(10)	UTF-8	UTF-8	DT_FIXCHAR 長さ 10
CHAR(10)	Windows-1252	UTF-8	DT_FIXCHAR 長さ 30
CHAR(20000)	Windows-31J	UTF-8	DT_LONGVARCHAR
CHAR(10 CHAR)	Windows-1252	Windows-1252	DT_FIXCHAR 長さ 10
CHAR(10 CHAR)	UTF-8	UTF-8	DT_FIXCHAR 長さ 40

ODBC では、odbc_distinguish_char_and_varchar オプションに応じて、CHAR は SQL_CHAR または SQL_VARCHAR として記述されます。

Ultra Light の備考

CHAR はドメインであり、VARCHAR として実装されます。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準とは互換性がありません。この標準では、文字長セマンティックがデフォルトですが、このソフトウェアではバイト長セマンティックがデフォルトです。大文字と小文字を区別しない照合のサポートとソフトウェアのブランク埋め込みに対するサポートのために、SQL 標準との間に、一部不整合があります。

ANSI/ISO SQL 標準では、SQL 言語機能 T061 として、明示的な文字長またはバイト長のセマンティックがサポートされています。

関連情報

VARCHAR データ型 [135 ページ]
LONG VARCHAR データ型 [129 ページ]
NCHAR データ型 [130 ページ]

1.2.1.2 LONG NVARCHAR データ型

LONG NVARCHAR データ型は、任意の長さの Unicode 文字データを格納します。

構文

LONG NVARCHAR

備考

最大サイズは 2 GB から 1 バイトを差し引いた値です (2^{31} - 1)。

文字は UTF-8 を使用して格納されます。各文字には 1 ～ 4 バイト必要です。LONG NVARCHAR に格納できる最大文字数は 5 億を超え、格納される文字の長さによっては 20 億を超えることもあります。

Embedded SQL クライアントで LONG NVARCHAR カラムに DESCRIBE が実行される場合、db_change_nchar_charset 関数が呼び出されたかどうかに応じて、返されるデータ型は DT_LONGVARCHAR または DT_LONGNVARCHAR になります。

ODBC では、LONG NVARCHAR 式は SQL_WLONGVARCHAR として記述されます。

標準

ANSI/ISO SQL 標準

LONG NVARCHAR を使用して各国の文字列を宣言できますが、この機能は標準ではありません。

関連情報

NCHAR データ型 [130 ページ]
NVARCHAR データ型 [132 ページ]
LONG VARCHAR データ型 [129 ページ]

1.2.1.3 LONG VARCHAR データ型

LONG VARCHAR データ型は、任意の長さの文字データを格納します。

構文

LONG VARCHAR

備考

最大バイトサイズは 2 GB から 1 バイトを差し引いた値です (2^{31} - 1)。

LONG VARCHAR データに、または LONG VARCHAR データから文字列をキャストできます。LONG VARCHAR データを連結することはできません。

SQL 文の条件 (WHERE 句の条件など) は、LONG VARCHAR カラムでは実行できません。LONG VARCHAR カラムでは、INSERT、UPDATE、DELETE の各操作が許可されます。LONG VARCHAR カラムは、SELECT クエリの結果セットに含めることもできます。

マルチバイト文字は LONG VARCHAR として格納できますが、長さは文字数ではなくバイト数です。

Ultra Light: LONG VARCHAR 型に対してインデックスを作成することはできません。LONG VARCHAR 型は LENGTH 関数と CAST 関数でのみ使用できます。

標準

ANSI/ISO SQL 標準

ラージオブジェクトサポートは SQL 言語機能 T041 です。

関連情報

CHAR データ型 [127 ページ]
VARCHAR データ型 [135 ページ]
LONG NVARCHAR データ型 [129 ページ]

1.2.1.4 NCHAR データ型

NCHAR データ型は、32767 文字までの Unicode 文字データを格納します。

構文

```
NCHAR [ ( max-length ) ]
```

パラメータ

max-length

文字列の文字単位での最大長。長さは、1 ～ 32767 の範囲内である必要があります。長さを指定しない場合、値は 1 になります。

備考

文字は UTF-8 エンコーディングを使用して格納されます。格納に必要な最大バイト数は、`max-length` を 4 倍した値です。ただし、通常、実際に使用される格納バイト数はより少ない値です。

たとえば、デザレット文字の Yee (U+10437) を UTF-8 でエンコーディングするには、4 バイト必要です。Interactive SQL を使用して次の SQL クエリを実行すると、文字 Yee が表示されます。

```
SELECT CAST(0xF09090B7 as NCHAR(1));
```

NCHAR は NATIONAL CHAR または NATIONAL CHARACTER と指定することもできます。どの構文を使用する場合でも、データ型は NCHAR と記述されます。

セマンティック上、NCHAR は NVARCHAR と同じですが、型は異なります。NCHAR は可変長型として扱われ、格納時にカラムにブランクが埋め込まれることはありません。

Embedded SQL クライアントが NCHAR カラムに対して DESCRIBE を実行した場合、db_change_nchar_charset 関数が呼び出されたかどうかに応じて、返されるデータ型は DT_FIXCHAR または DT_NFIXCHAR になります。

また、Embedded SQL クライアントで NCHAR カラムに DESCRIBE が実行される場合、返される長さは、クライアントの NCHAR 文字セットの最大バイト長です。たとえば、西ヨーロッパ言語の文字セット cp1252 を NCHAR 文字セットに使用する Embedded SQL クライアントの場合、NCHAR(10) カラムは、長さ 10 (10 文字の文字ごとに最大で 1 バイトを乗じた値) の DT_NFIXCHAR 型として記述されます。日本語の文字セット cp932 を使用する Embedded SQL クライアントの場合、同じ NCHAR(10) カラムは長さ 20 (10 文字の文字ごとに最大で 2 バイトを乗じた値) の DT_NFIXCHAR 型として記述されます。記述された長さが 32767 バイトを超える場合、カラムは DT_LONGNVARCHAR 型として記述されます。

ODBC では、NCHAR は、バイト長 (オクテット長) が 32767 より小さい場合 SQL_WCHAR と記述され、それ以外の場合 SQL_WLONGVARCHAR と記述されます。たとえば、NCHAR(8192) は最大 32768 バイトの記憶領域が必要であり、2147483647 オクテット長の SQL_WLONGVARCHAR と記述されます。

標準

ANSI/ISO SQL 標準

各国の文字のサポートは、機能 F421 です。

関連情報

CHAR データ型 [127 ページ]
NVARCHAR データ型 [132 ページ]
LONG NVARCHAR データ型 [129 ページ]

1.2.1.5 NTEXT データ型

NTEXT データ型は、任意の長さの Unicode 文字データを格納します。

構文

```
NTEXT
```

備考

NTEXT はドメインです。LONG NVARCHAR として実装されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

LONG NVARCHAR データ型 [129 ページ]
TEXT データ型 [134 ページ]

1.2.1.6 NVARCHAR データ型

NVARCHAR データ型は、32767 文字までの Unicode 文字データを格納します。

構文

```
NVARCHAR [ ( max-length ) ]
```

パラメータ

max-length

文字列の文字単位での最大長。長さは、1 ～ 32767 の範囲内である必要があります。長さを指定しない場合、値は 1 になります。

備考

文字は UTF-8 エンコーディングを使用して格納されます。格納に必要な最大バイト数は `max-length` を 4 倍した値ですが、通常、実際に使用される格納バイト数はより少ない値です。

たとえば、デザレット文字の Yee (U+10437) を UTF-8 でエンコーディングするには、4 バイト必要です。Interactive SQL を使用して次の SQL クエリを実行すると、文字 Yee が表示されます。

```
SELECT CAST(0xF09090B7 as NVARCHAR(1));
```

NVARCHAR は、CHAR VARYING、NATIONAL CHAR VARYING、または NATIONAL CHARACTER VARYING と指定することもできます。どの構文を使用する場合でも、データ型は NVARCHAR と記述されます。

Embedded SQL クライアントで NVARCHAR カラムに DESCRIBE が実行される場合、db_change_nchar_charset 関数が呼び出されたかどうかに応じて、返されるデータ型は DT_VARCHAR または DT_NVARCHAR になります。

また、Embedded SQL クライアントで NVARCHAR カラムに DESCRIBE が実行される場合、返される長さは、クライアントの NCHAR 文字セットの最大バイト長です。たとえば、西ヨーロッパ言語の文字セット cp1252 を NCHAR 文字セットに使用する Embedded SQL クライアントの場合、NVARCHAR(10) カラムは、長さ 10 (10 文字の文字ごとに最大で 1 バイトを乗じた値) の DT_NVARCHAR 型として記述されます。日本語の文字セット cp932 を使用する Embedded SQL クライアントの場合、同じ NCHAR(10) カラムは長さ 20 (10 文字の文字ごとに最大で 2 バイトを乗じた値) の DT_NVARCHAR 型として記述されます。記述された長さが 32767 バイトを超える場合、カラムは DT_LONGNVARCHAR 型として記述されます。

ODBC では、NVARCHAR は、バイト長 (オクテット長) が 32767 より小さい場合 SQL_WVARCHAR と記述され、それ以外の場合 SQL_WLONGVARCHAR と記述されます。たとえば、NVARCHAR(8192) は最大 32768 バイトの記憶領域が必要であり、2147483647 オクテット長の SQL_WLONGVARCHAR と記述されます。

標準

ANSI/ISO SQL 標準

各国の文字のサポートは、SQL 言語機能 F421 です。

関連情報

NCHAR データ型 [130 ページ]
LONG NVARCHAR データ型 [129 ページ]
VARCHAR データ型 [135 ページ]

1.2.1.7 TEXT データ型

TEXT データ型は、任意の長さの文字データを格納します。

構文

TEXT

備考

TEXT はドメインです。LONG VARCHAR として実装されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

LONG VARCHAR データ型 [129 ページ]
NTEXT データ型 [132 ページ]

1.2.1.8 UNIQUEIDENTIFIERSTR データ型

UNIQUEIDENTIFIERSTR は、CHAR(36) として実装されたドメインです。

構文

UNIQUEIDENTIFIERSTR

備考

Microsoft SQL Server の uniqueidentifier カラムをマッピングするとき、リモートデータアクセスに使用されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

STRTOUUID 関数 [文字列] [548 ページ]

1.2.1.9 VARCHAR データ型

VARCHAR データ型は、32767 バイトまでの文字データを格納します。

構文

```
VARCHAR [ ( max-length [ CHAR | CHARACTER ] ) ]
```

Ultra Light:

```
VARCHAR [ ( max-length) ]
```

パラメータ

max-length

文字列の最大長。このデフォルト値は 1 です。

バイト長のセマンティックを使用する場合 (長さの部分に CHAR または CHARACTER を指定しない場合)、長さはバイト単位になります。また、範囲は 1 ～ 32767 にします。文字長のセマンティックを使用する場合 (長さの部分に CHAR または CHARACTER を指定する場合)、長さは文字単位になります。また、`max-length` を指定します。`max-length` は最大 32767 文字までです。

Ultra Light: Ultra Light データベースはバイト長のセマンティックのみをサポートしています。英語以外の文字は最大 3 バイトの記憶領域が必要になることがあります。

備考

マルチバイト文字は VARCHAR として格納できますが、宣言される長さは文字数ではなくバイト数です。

Ultra Light: Ultra Light はデータを可能な限りコンパクトにします。VARCHAR 値が `max-length` で指定されたバイト数を必要としない場合は、値を格納するために必要なバイト数のみが使用されます。式を評価するときのテンポラリ文字値の最大長は 2048 バイトです。

> ⚠ 警告
>
> **Ultra Light:**
>
> `max-length` がページサイズを超えている VARCHAR カラムを使用してテーブルを作成することはできますが、そのページサイズを超える長さの値を挿入するとエラーが発生します。

ODBC では、VARCHAR は SQL_VARCHAR と記述されます。VARCHAR は CHAR VARYING または CHARACTER VARYING と指定することもできます。どの構文を使用する場合でも、データ型は VARCHAR と記述されます。セマンティック上、VARCHAR は CHAR と同じですが、型は異なります。SQL Anywhere では、VARCHAR は可変長型です。他のリレーショナルデータベース管理システムでは、VARCHAR は固定長型であり、データには `max-length` バイトまでブランクが埋め込まれて格納されます。SQL Anywhere では格納される文字データにブランクを埋め込みません。VARCHAR カラムがどのように記述されるかは、クライアントインタフェース、使用される文字セット、および文字長セマンティックが使用されるかどうかによって異なります。たとえば、Embedded SQL では、記述される長さはクライアント文字セットの最大バイト数となります。記述される長さが 32767 バイトを超える場合、カラムは DT_LONGVARCHAR 型として記述されます。次の表は、Embedded SQL の例と、DESCRIBE を実行した場合に返される結果を示します。

記述される型	データベースの文字セット	クライアントの文字セット	DESCRIBE の結果
VARCHAR(10)	Windows-1252	Windows-1252	DT_VARCHAR 長さ 10
VARCHAR(10)	UTF-8	UTF-8	DT_VARCHAR 長さ 10
VARCHAR(10)	Windows-1252	UTF-8	DT_VARCHAR 長さ 30
VARCHAR(20000)	Windows-31J	UTF-8	DT_LONGVARCHAR
VARCHAR(10 CHAR)	Windows-1252	Windows-1252	DT_VARCHAR 長さ 10
VARCHAR(10 CHAR)	UTF-8	UTF-8	DT_VARCHAR 長さ 40

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準とは互換性がありません。この標準では、文字長セマンティックがデフォルトですが、このソフトウェアではバイト長セマンティックがデフォルトです。大文字と小文字を区別しない照合のサポートと、このソフトウェアによるブランク埋め込みに対するサポートのために、SQL 標準との間に、一部不整合があります。

ANSI/ISO SQL 標準では、SQL 言語機能 T061 として、明示的な文字長またはバイト長のセマンティックがサポートされています。

関連情報

CHAR データ型 [127 ページ]

LONG VARCHAR データ型 [129 ページ]
NVARCHAR データ型 [132 ページ]

1.2.1.10 XML データ型

XML データ型は、任意の長さの文字データを格納し、XML ドキュメントを格納します。

構文

XML

備考

最大サイズは 2 GB から 1 バイトを差し引いた値です ($2^{31} - 1$)。

リレーショナルデータから要素内容を生成する場合、XML データ型は引用符で囲まれません。

XML データ型は、文字列と相互にキャスト可能なすべてのデータ型と、相互にキャストできます。文字列が XML にキャストされるときに、整形式かどうかはチェックされません。

Embedded SQL クライアントアプリケーションで、XML カラムに DESCRIBE が実行された場合、LONG VARCHAR と記述されます。

標準

ANSI/ISO SQL 標準

XML データ型は、SQL 言語機能 X010 です。

1.2.2 数値データ型

数値データ型は数値データを格納します。

NUMERIC データ型、DECIMAL データ型、さまざまな INTEGER データ型は真数値データ型と呼ぶこともあります。これと対照的なのが、概数値データ型である FLOAT、DOUBLE、REAL です。

真数値データ型は、精度と小数点以下の桁数の値を指定できる型です。一方、概数値データ型は事前に定義された方法で格納されます。真数値データだけが、算術演算後に指定した最小有効桁数に対して正確性が保証されます。

1 より小さいデータ型の長さと精度は使用できません。

互換性

NUMERIC と DECIMAL のデータ型にデフォルトの精度と位取りの設定を使用する場合は注意が必要です。他のデータベースソリューションでは設定が異なる場合があるためです。デフォルトの精度は 30、デフォルトの位取りは 6 です。

FLOAT (p) データ型は、p の値によって、REAL か DOUBLE のどちらかの同義語になります。SQL Anywhere では、カットオフ値はプラットフォームによって異なりますが、どのプラットフォームでもカットオフ値は 16 以上です。

Transact-SQL の identity カラムでは、scale = 0 の NUMERIC データ型のみ使用できます。NUMERIC データ型と DECIMAL データ型では、デフォルトの精度と位取り設定を使用しないでください。これは、SQL Anywhere と Adaptive Server Enterprise で設定内容が異なるためです。SQL Anywhere では、デフォルトの精度は 30、デフォルトの位取りは 6 です。Adaptive Server Enterprise では、デフォルトの精度は 18、デフォルトの位取りは 0 です。

このセクションの内容:

1.2.2.1 BIGINT データ型

BIGINT データ型は、8 バイトの記憶領域を必要とする整数である BIGINT を格納します。

構文

```
[ UNSIGNED ] BIGINT
```

備考

BIGINT データ型は真数値データ型です。精度は算術演算の後で保存されます。

BIGINT 値には 8 バイトの記憶領域が必要です。

BIGINT 値の範囲は、-2^{63} ～ $2^{63} - 1$、または -9223372036854775808 ～ 9223372036854775807 です。

UNSIGNED BIGINT 値の範囲は、0 ～ $2^{64} - 1$、または 0 ～ 18446744073709551615 です。

デフォルトでは、このデータ型は符号付きです。

文字列を BIGINT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効な範囲外の値である場合、文字列に不正な文字が含まれる場合、または文字列を整数値として復号化できない場合にエラーが返されます。

標準

ANSI/ISO SQL 標準

SQL 言語機能 T071。

MySQL

UNSIGNED キーワードは BIGINT の後ろに配置することもできます。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
BIT データ型 [139 ページ]
INTEGER データ型 [144 ページ]
SMALLINT データ型 [147 ページ]
TINYINT データ型 [148 ページ]

1.2.2.2 BIT データ型

BIT データ型は、1 つのビット (0 または 1) を格納します。

構文

```
BIT
```

備考

BIT は、0 または 1 の値を格納できる整数型です。

デフォルトでは、BIT データ型には NULL を入力できません。

文字列を BIT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。値が 0 または 1 でない場合はエラーが返されます。

BIT 値には 1 バイトの記憶領域が必要です。

Ultra Light: BIT 値には 1 ビットの記憶領域が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
BIGINT データ型 [138 ページ]
INTEGER データ型 [144 ページ]
SMALLINT データ型 [147 ページ]
TINYINT データ型 [148 ページ]

1.2.2.3 DECIMAL データ型

DECIMAL データ型は、総桁数の `precision` と小数点以下の桁数の `scale` を持つ 10 進数です。

構文

```
DECIMAL [ ( precision [ , scale ] ) ]
```

パラメータ

precision

式の桁数を指定する整数式。1 ～ 127 の範囲。デフォルト設定値は 30 です。

scale

小数点以下の桁数を指定する整数式。0 ～ 127 の範囲。scale 値は precision 値以下にする必要があります。デフォルト設定値は 6 です。

デフォルト値はデータベースオプションの設定によって変更します。

Ultra Light: デフォルト値は該当する作成パラメータによって変更します。

備考

DECIMAL データ型は真数値データ型です。精度は、算術演算の後、最小の有効桁数まで保存されます。

10 進数を格納するために必要なバイト数は、次のように計算できます。

```
2 + INT(((precision - scale) + 1) / 2) + INT((scale + 1) / 2);
```

INT 関数は引数の整数部分です。記憶領域の計算の基準は、カラムの中で使われている最大精度と小数点以下の桁数ではなく、格納されている値を使います。

使用する精度が 20 以下で小数点以下の桁数が 0 の場合、代わりに整数データ型のいずれか (BIGINT、INTEGER、SMALLINT、または TINYINT) を使用できることがあります。整数値では、有効桁数が同じくらいの NUMERIC 値や DECIMAL 値よりも少ない記憶領域が使用されます。整数値の操作 (フェッチ、挿入など) や算術演算子は、通常、NUMERIC 値や DECIMAL 値の操作よりも速く実行されます。

> **注記**
>
> DECIMAL データ型のカラムまたは変数を作成するとき、これらのカラムまたは変数の精度または位取りがデータベースの精度および位取りの設定を超過する場合は、値はデータベースの設定に合わせてトランケートされます。このため、DECIMAL として定義されているカラムまたは変数の値がトランケートされる場合は、精度と位取りがデータベースのオプション設定を超過していないかどうかを確認します。

DECIMAL は DEC と指定することもできます。どの構文を使用する場合でも、データ型は DECIMAL と記述されます。セマンティック上、DECIMAL は NUMERIC と同じです。

標準

ANSI/ISO SQL 標準

コア機能。

関連情報

数値関数 [210 ページ]

集合関数 [200 ページ]
FLOAT データ型 [143 ページ]
REAL データ型 [146 ページ]
DOUBLE データ型 [142 ページ]
NUMERIC データ型 [145 ページ]

1.2.2.4 DOUBLE データ型

DOUBLE データ型は、倍精度の浮動小数点数を格納します。

構文

DOUBLE

備考

DOUBLE データ型は概数値データ型であり、算術演算後に丸め誤差が出ます。概数値という DOUBLE 値の性質により、通常、DOUBLE 値を比較するときは等号を使用するクエリを避けてください。

DOUBLE 値には 8 バイトの記憶領域が必要です。

値の範囲は -1.79769313486231e+308 ～ 1.79769313486231e+308 です。2.22507385850721e-308 がほぼ 0 です。DOUBLE として保持される値の有効桁数は厳密には 15 桁です。15 桁を超えると丸め誤差が出る可能性があります。

標準

ANSI/ISO SQL 標準

コア機能

関連情報

数値関数 [210 ページ]
数値セットの変換 [196 ページ]
集合関数 [200 ページ]
FLOAT データ型 [143 ページ]
REAL データ型 [146 ページ]
DECIMAL データ型 [140 ページ]

NUMERIC データ型 [145 ページ]

1.2.2.5 FLOAT データ型

FLOAT データ型は、単精度または倍精度の浮動小数点数を格納します。

構文

```
FLOAT [ ( precision ) ]
```

パラメータ

precision

仮数部 (常用対数の小数部) の桁数を指定する整数式。たとえば、5.63428 という数値の仮数は 0.63428 です。IEEE 標準 754 による浮動小数点精度は、次のとおりです。

提供されている精度値	10 進数精度	等価の SQL データ型	記憶サイズ
1〜24	7 桁	REAL	4 バイト
25-53	15 桁	DOUBLE	8 バイト

備考

FLOAT (`precision`) データ型を使用してカラムを作成した場合、すべてのプラットフォーム上でカラムが少なくとも指定の最小精度の値を持つことが保証されます。REAL と DOUBLE の場合、プラットフォームに依存しない最小精度は保証されません。

`precision` を指定しない場合、FLOAT データ型は単精度の浮動小数点数です。これは REAL データ型と等価で、4 バイトの記憶領域を必要とします。

`precision` を指定する場合、FLOAT データ型は指定した精度の値によって、単精度か倍精度のいずれかになります。REAL になるか DOUBLE になるかは、プラットフォームによって異なります。単精度の FLOAT 値は、4 バイトの記憶領域を必要とし、倍精度の FLOAT 値は、8 バイトを必要とします。

FLOAT データ型は概数値データ型です。算術演算後の丸め誤差がでます。概数値という FLOAT 値の性質により、FLOAT 値を比較するときは等号を使用するクエリを避けてください。

標準

ANSI/ISO SQL 標準

コア機能。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
DOUBLE データ型 [142 ページ]
REAL データ型 [146 ページ]
DECIMAL データ型 [140 ページ]
NUMERIC データ型 [145 ページ]

1.2.2.6 INTEGER データ型

INTEGER データ型は、4 バイトの記憶領域を必要とする整数を格納します。

構文

```
[ UNSIGNED ] INTEGER
```

備考

INTEGER データ型は真数値データ型で、精度は算術演算の後で保存されます。

UNSIGNED を指定した場合、整数に負の数を割り当てることはできません。デフォルトでは、このデータ型は符号付きです。

INTEGER 値の範囲は、-2^{31} ～ $2^{31} - 1$、または -2147483648 ～ 2147483647 です。

UNSIGNED INTEGER 値の範囲は、0 ～ $2^{32} - 1$、または 0 ～ 4294967295 です。

文字列を INTEGER に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効な範囲外の値である場合、文字列に不正な文字が含まれる場合、または文字列を整数値として復号化できない場合にエラーが返されます。

標準

ANSI/ISO SQL 標準

コア機能。ただし、UNSIGNED キーワードは標準にありません。
MySQL

UNSIGNED キーワードは INTEGER の後ろに配置することもできます。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
BIGINT データ型 [138 ページ]
BIT データ型 [139 ページ]
SMALLINT データ型 [147 ページ]
TINYINT データ型 [148 ページ]

1.2.2.7 NUMERIC データ型

NUMERIC データ型は、総桁数の `precision` と小数点以下の桁数の `scale` を持つ 10 進数を格納します。

構文

```
NUMERIC [ ( precision [ , scale ] ) ]
```

パラメータ

precision

式の桁数を指定する整数式。1 ～ 127 の範囲。デフォルト設定値は 30 です。

scale

小数点以下の桁数を指定する整数式。0 ～ 127 の範囲。scale 値は precision 値以下にする必要があります。デフォルト設定値は 6 です。

備考

NUMERIC データ型は真数値データ型です。精度は、算術演算の後、最小の有効桁数まで保存されます。

Ultra Light: NUMERIC はドメインであり、DECIMAL として実装されます。

10 進数を格納するために必要なバイト数は、次のように計算できます。

```
2 + INT( (BEFORE+1)/2 ) + INT( (AFTER+1)/2 );
```

INT 関数は引数の整数部分であり、BEFORE と AFTER は小数点の前後の有効桁数です。記憶領域の計算の基準は、カラムの中で使われている最大精度と小数点以下の桁数ではなく、格納されている値を使います。

使用する精度が 20 以下で小数点以下の桁数が 0 の場合、代わりに整数データ型のいずれか (BIGINT、INTEGER、SMALLINT、または TINYINT) を使用できることがあります。整数値では、有効桁数が同じくらいの NUMERIC 値や DECIMAL 値よりも少ない記憶領域が使用されます。整数値の操作 (フェッチ、挿入など) や算術演算子は、通常、NUMERIC 値や DECIMAL 値の操作よりも速く実行されます。

セマンティック上、NUMERIC は DECIMAL と同じです。

注記

NUMERIC データ型のカラムまたは変数を作成するとき、これらのカラムまたは変数の精度または位取りがデータベースの精度および位取りの設定を超過する場合は、値はデータベースの設定に合わせてトランケートされます。このため、NUMERIC として定義されているカラムまたは変数の値がトランケートされる場合は、精度と位取りがデータベースのオプション設定を超過していないかどうかを確認します。

標準

ANSI/ISO SQL 標準

scale オプションが 0 に設定されている場合、ANSI/ISO SQL と互換性があります。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
数値セットの変換 [196 ページ]
FLOAT データ型 [143 ページ]
REAL データ型 [146 ページ]
DOUBLE データ型 [142 ページ]
DECIMAL データ型 [140 ページ]

1.2.2.8 REAL データ型

REAL データ型は、4 バイトで格納される単精度の浮動小数点数を格納します。

構文

```
REAL
```

備考

REAL データ型は概数値データ型であり、算術演算後に丸め誤差が出ます。概数値という REAL 値の性質により、通常、REAL 値を比較するときは等号を使用するクエリを避けてください。

REAL 値には 4 バイトの記憶領域が必要です。

値の範囲は -3.402823e+38 ～ 3.402823e+38 です。1.175494351e-38 がほぼ 0 です。REAL として保持される値は、厳密には 7 有効桁数ですが、7 桁以上になると丸め誤差が出ます。

標準

ANSI/ISO SQL 標準

コア機能。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
DOUBLE データ型 [142 ページ]
FLOAT データ型 [143 ページ]
DECIMAL データ型 [140 ページ]
NUMERIC データ型 [145 ページ]

1.2.2.9 SMALLINT データ型

SMALLINT データ型は、2 バイトの記憶領域を必要とする整数を格納します。

構文

```
[ UNSIGNED ] SMALLINT
```

備考

SMALLINT データ型は真数値データ型です。精度は算術演算の後で保存されます。2 バイトの記憶領域を必要とします。

SMALLINT 値の範囲は、-2^{15} ～ $2^{15} - 1$、または -32768 ～ 32767 です。

UNSIGNED SMALLINT 値の範囲は、0 ～ 2^{16} - 1、または 0 ～ 65535 です。

文字列を SMALLINT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効な範囲外の値である場合、文字列に不正な文字が含まれる場合、または文字列を整数値として復号化できない場合にエラーが返されます。

標準

ANSI/ISO SQL 標準

標準と互換性があります。ただし、UNSIGNED キーワードは標準にありません。

MySQL

UNSIGNED キーワードは SMALLINT の後ろに配置することもできます。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
BIGINT データ型 [138 ページ]
BIT データ型 [139 ページ]
INTEGER データ型 [144 ページ]
TINYINT データ型 [148 ページ]

1.2.2.10 TINYINT データ型

TINYINT データ型は、1 バイトの記憶領域を必要とする符号なし整数を格納します。

構文

TINYINT

備考

TINYINT データ型は真数値データ型です。精度は算術演算の後で保存されます。

TINYINT 値の範囲は、0 ～ 2^{8} - 1、または 0 ～ 255 です。

文字列を TINYINT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変

換しようとしたデータ型の有効な範囲外の値である場合、文字列に不正な文字が含まれる場合、または文字列を整数値として復号化できない場合にエラーが返されます。

Embedded SQL では、TINYINT カラムを CHAR または UNSIGNED CHAR と定義された変数にフェッチしないでください。そうした場合、カラムの値を文字列に変換し、最初のバイトをプログラムの変数に割り当てようとすることになるからです。TINYINT カラムは 2 バイトまたは 4 バイト整数にフェッチしてください。TINYINT 値を C で書かれたアプリケーションからデータベースに送るには、C 変数の型に INTEGER を指定してください。

Ultra Light: Embedded SQL では、TINYINT カラムを CHAR と定義された変数にフェッチしないでください。そうした場合、カラムの値を文字列に変換し、最初のバイトをプログラムの変数に割り当てようとすることになるからです。TINYINT カラムは 2 バイトまたは 4 バイト整数にフェッチしてください。TINYINT 値を C で書かれたアプリケーションからデータベースに送るには、C 変数の型に INTEGER を指定してください。

標準

ANSI/ISO SQL 標準

標準になし。

MySQL

UNSIGNED キーワードは TINYINT の前に配置しても後ろに配置しても構いませんが、この型は必ず符号なしであるため、UNSIGNED 変更子は効力を持ちません。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
BIGINT データ型 [138 ページ]
BIT データ型 [139 ページ]
INTEGER データ型 [144 ページ]
SMALLINT データ型 [147 ページ]

1.2.3 通貨データ型

通貨データ型は、通貨データを格納するために使用します。

このセクションの内容:

1.2.3.1 MONEY データ型

MONEY データ型は、通貨データを格納します。

構文

MONEY

備考

MONEY は NUMERIC(19,4) として実装されるドメインです。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
SMALLMONEY データ型 [150 ページ]

1.2.3.2 SMALLMONEY データ型

SMALLMONEY データ型は、100 万通貨単位未満の通貨データを格納します。

構文

SMALLMONEY

備考

SMALLMONEY は NUMERIC(10,4) として実装されるドメインです。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

数値関数 [210 ページ]
集合関数 [200 ページ]
MONEY データ型 [150 ページ]

1.2.4 ビット配列データ型

ビット配列は文字列に似ていますが、使用される要素は文字ではなくビットデータ (0 と 1) です。通常、ビット配列は、ブール値の文字列を保持するのに使用されます。

サポートされるビット配列データ型には、VARBIT と LONG VARBIT があります。

このセクションの内容:

LONG VARBIT データ型 [151 ページ]
LONG VARBIT データ型は、任意の長さのビット配列を格納します。

VARBIT データ型 [152 ページ]
VARBIT データ型は、長さが 32767 未満のビット配列を格納するときに使用します。

1.2.4.1 LONG VARBIT データ型

LONG VARBIT データ型は、任意の長さのビット配列を格納します。

構文

LONG VARBIT

備考

任意の長さのビット (1 と 0) 配列または 32767 ビットを超えるビット配列を格納するときに使用します。

LONG VARBIT は LONG BIT VARYING と指定することもできます。どの構文を使用する場合でも、データ型は LONG VARBIT と記述されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

ビット配列の変換 [194 ページ]
ビット配列関数 [202 ページ]
集合関数 [200 ページ]
BIT データ型 [139 ページ]
VARBIT データ型 [152 ページ]

1.2.4.2 VARBIT データ型

VARBIT データ型は、長さが 32767 未満のビット配列を格納するときに使用します。

構文

```
VARBIT [ (max-length ) ]
```

パラメータ

max-length

ビット配列のバイト単位での最大長。長さは、1 ～ 32767 の範囲内である必要があります。長さを指定しない場合、値は 1 になります。

備考

VARBIT は BIT VARYING と指定することもできます。どの構文を使用する場合でも、データ型は VARBIT と記述されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

ビット配列の変換 [194 ページ]
ビット配列関数 [202 ページ]
集合関数 [200 ページ]
BIT データ型 [139 ページ]
LONG VARBIT データ型 [151 ページ]
ビット処理演算子 [31 ページ]

1.2.5 日付と時刻データ型

日付の値は 4 桁の西暦で出力でき、日付の内部記憶領域には、年の値の世紀にあたる部分が常に明示的に含まれています。

算出した値が別の世紀にかかるかどうかにかかわらず、日付に関する有効な算術演算や論理演算に対して常に正しい値を返します。

このセクションの内容:

日付と時刻が格納されるしくみ [154 ページ]
日付と時刻は、いずれかのデータ型を使用してデータベースに格納されます。

日付と時刻をデータベースに送信する方法 [154 ページ]
日付と時刻はどのインタフェースを使用しても文字列としてデータベースに送信できます。さらに、ODBC または OLE DB を使用してバイナリ値として送信したり (たとえば、ODBC TIMESTAMP_STRUCT 構造を使用)、Embedded SQL を使用して SQLDATETIME 構造として送信したりすることもできます。

日付と時刻のデータベースからの取得 [160 ページ]
日付と時刻は、どのインタフェースを使用してもデータベースから文字列として取得できます。さらに、ODBC または OLE DB を使用してバイナリ値として取得したり (たとえば、ODBC TIMESTAMP_STRUCT 構造を使用)、Embedded SQL を使用して SQLDATETIME 構造として取得したりすることもできます。

DATE データ型 [161 ページ]
DATE データ型は、年、月、日などの暦日を格納します。

DATETIME データ型 [162 ページ]
DATETIME は、日付と時刻の情報を格納します。

DATETIMEOFFSET データ型 [164 ページ]
DATETIMEOFFSET データ型は TIMESTAMP WITH TIME ZONE のエイリアスであり、日付、時刻、およびタイムゾーン情報の格納に使用されます。

SMALLDATETIME データ型 [166 ページ]
SMALLDATETIME は、TIMESTAMP として実装されるドメインで、日付と時刻の情報を格納するために使用します。SMALLDATETIME は Transact-SQL の型です。

TIME データ型 [167 ページ]
TIME データ型は、時、分、秒、秒以下で構成される時刻を格納します。

TIMESTAMP データ型 [168 ページ]
TIMESTAMP データ型は、年、月、日、時、分、秒、秒以下 (小数第 6 位まで) で構成される時刻を格納します。

TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
TIMESTAMP WITH TIME ZONE データ型は、タイムゾーンオフセット付きの時刻を格納します。

1.2.5.1 日付と時刻が格納されるしくみ

日付と時刻は、いずれかのデータ型を使用してデータベースに格納されます。

データ型	内容	記憶領域	指定できる値の範囲
DATE	暦日 (年、月、日)	4 バイト	0001-01-01 ～ 9999-12-31 の日付。
TIME	時刻 (時、分、秒、秒の小数部分 (小数点第 6 位まで))	8 バイト	00:00:00.000000 ～ 24:00:00.000000 の時刻。
TIMESTAMP	日の暦日と時刻 (年、月、日、時、分、秒、秒の小数部分 (小数点第 6 位まで))	8 バイト	0001-01-01 ～ 9999-12-31 の日付 (TIMESTAMP の時と分の精度については、1600-02-28 23:59:59 より前と 7911-01-01 00:00:00 より後の部分は削除されます)。
TIMESTAMP WITH TIME ZONE	暦日、時刻、タイムゾーンオフセット (年、月、日、時、分、秒、秒の小数部分 (小数点第 6 位まで)、時間単位と分単位のタイムゾーンオフセット)	10 バイト	0001-01-01 ～ 9999-12-31 の日付 (TIMESTAMP WITH TIME ZONE の時と分の精度については、1600-02-28 23:59:59 より前と 7911-01-01 00:00:00 より後の部分は削除されます)。-14:59 ～ +14:59 のゾーンオフセット。

1.2.5.2 日付と時刻をデータベースに送信する方法

日付と時刻はどのインタフェースを使用しても文字列としてデータベースに送信できます。さらに、ODBC または OLE DB を使用してバイナリ値として送信したり (たとえば、ODBC TIMESTAMP_STRUCT 構造を使用)、Embedded SQL を使用して SQLDATETIME 構造として送信したりすることもできます。

タイムゾーンオフセット付きの日付と時刻は、データベースに文字列としてのみ送信できます。

このセクションの内容:

日付形式 [155 ページ]
日付を文字列 (DATE データ型用) として、または文字列の一部 (TIMESTAMP データ型または TIMESTAMP WITH TIME ZONE データ型用) としてデータベースに送信する場合、文字列は、国際標準である ISO 8601 によって記述されるものなど、さまざまな方法で指定できます。

時間形式 [157 ページ]
時刻は、24 時間管理システムを使って ISO 8601 形式で指定できます。

時刻付きの日付形式 [158 ページ]
ISO 8601 では、日付と時刻で、スペースまたは文字 T を使用して組み合わせることができます。

タイムゾーン形式 [159 ページ]
ISO 8601 は、また、日文字列の日付と時刻にタイムゾーンオフセットを追加することを許可します。

関連情報

日付と時刻の比較 [189 ページ]
DATE データ型 [161 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.2.5.2.1 日付形式

日付を文字列 (DATE データ型用) として、または文字列の一部 (TIMESTAMP データ型または TIMESTAMP WITH TIME ZONE データ型用) としてデータベースに送信する場合、文字列は、国際標準である ISO 8601 によって記述されるものなど、さまざまな方法で指定できます。

日付は、次の ISO 8601 形式の 1 つで指定できます。

暦日

暦日の形式は YYYY-MM-DD であり、YYYY はグレゴリオ暦の年、MM は 01 (1 月) から 12 (12 月) までの月、DD は 01 から 31 までの日になります。たとえば、'2010-04-01' は、2010 年 4 月 1 日を表します。ISO 8601 は区切り文字を要求しません。ISO 8601 は区切り文字を要求しません。そのため、'20100401' も 2010 年 4 月 1 日を表します。

ISO 暦日	形式	例
基本	YYYYMMDD	20100401
拡張	YYYY-MM-DD	2010-04-01

週日

もう 1 つの ISO 日付形式は週日の形式です。この形式は YYYY-Www-D であり、YYYY はグレゴリオ暦の年、W は文字 W、ww は 01 (第 1 週) から 52 または 53 (最終週) までの年内の暦週、D は 1 (月曜日) から 7 (日曜日) までの週内の曜日です。たとえば、'2010-W13-4' は、2010 年の 13 番目の週の 4 番目の日 (2010 年 4 月 1 日) を表します。ISO

8601 は区切り文字を要求しません。そのため、'2010W134' も 2010 年の 13 番目の週の 4 番目の日を表します。精度を下げた場合、1 桁目を表現から省略できます ('2010W13' は 2010 年 3 月 29 日を表します)。

ISO 週日	形式	例
Basic	YYYYWwwD	2010W134
拡張	YYYY-Www-D	2010-W13-4

年内の日の番号

最後の ISO 日付形式は、年内の日の番号です。形式は YYYY-DDD であり、YYYY はグレゴリオ暦の年、DDD はその暦年内で第何日目かを指します。たとえば、'2010-091' は、2010 年 4 月 1 日を表します。ISO 8601 は区切り文字を要求しません。ISO 8601 は区切り文字を要求しません。たとえば、'2010091'も 2010 年 4 月 1 日を表します。年内の日の番号の最大値は、閏年用の 366 です。たとえば、'2008366' は、2008 年の最後の日 (2008 年 12 月 31 日) を表します。

ISO 年内の日の番号	形式	例
Basic	YYYYDDD	2010091
拡張	YYYY-DDD	2010-091

これら以外の日付形式もサポートされています。SQL Anywhere は、日付を含む文字列を非常に柔軟に解釈します。あいまいである場合は、date_order および nearest_century データベースオプション設定に従って日付値の解釈が行われます。たとえば、date_order 設定に従い、'02/05/2002' は、データベースサーバの解釈によって、5 月 2 日 (DMY) とすることも、2 月 5 日 (MDY) とすることも、無効な値 (YMD) とすることもできます。

nearest_century 設定は、2 桁の年の値を 20 世紀と 21 世紀のどちらとして解釈するかを決定します。たとえば、文字列 '02/05/10' では、date_order 設定が 02 と 10 のどちらを年として解釈するのかを決定し、nearest_century 設定が 02 で 1902 年と 2002 年のどちらを表すのか、または 10 で 1910 年と 2010 年のどちらを表すのかを決定します。nearest_century オプションの値は、2 桁の年数の解釈に影響します。nearest_century より少ない値には 2000 が加算され、その他の値にはすべて 1900 が加算されます。このオプションのデフォルト値は 50 です。そのため、デフォルトで 50 年は 1950 年、49 年は 2049 年と解釈されます。

次の表は、指定された date_order 設定と 50 の nearest_century 設定を使って、2010 年の 4 月 1 日をどのように指定できるのかを示しています。

date_order	形式	例
YMD	YYYY/MM/DD	2010/04/01
YMD	YY/MM/DD	2001/10/04
MDY	MM/DD/YYYY	2010-04-01
MDY	MM/DD/YY	2010-04-01
DMY	DD/MM/YYYY	2010/01/04
DMY	DD/MM/YY	2010/01/04

ISO 8601 形式はあいまいではなく、date_order と nearest_century のユーザ設定の影響を受けないので、ISO 8601 を使用することをお奨めします。

また、日付は月の名前を使って指定することもできます。たとえば、'2010 April 01'、'April 1, 2010'、'1 April 2010' などがあります。年があいまいに指定された場合、date_order オプションを使って、年と日の部分を分解できます。そのため、'01

April 10' は、date_order が 'YMD' のときは 2001 年 4 月 10 日と解釈され、date_order が 'DMY' のときは 2010 年 4 月 1 日と解釈されます。

日付の中の年の値の範囲は 0001 ～ 9999 です。最小日付は 0001-01-01 です。

文字列に部分的な日付指定だけがある場合、デフォルト値を使って日付が満たされます。次のデフォルトを使います。

年

年が何も指定されない場合 ('April 1' など) は、現在の年が使用されます。

月

年と月が何も指定されない ('23:59:59' など) の場合は、現在の月が使用されます。また、年が指定された場合 ('2010' など) は、01 が使用されます。

日

年と月が何も指定されない ('23:59:59' など) の場合は、現在の日が使用されます。また、月が指定された場合 ('April' など) は、01 が使用されます。

次の例では、現在の日付から日付値が構築されます。

```
SELECT CAST('23:59:59' AS TIMESTAMP);
```

関連情報

日付と時刻の比較 [189 ページ]
DATE データ型 [161 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.2.5.2.2 時間形式

時刻は、24 時間管理システムを使って ISO 8601 形式で指定できます。

これは hh:mm:ss であり、hh は真夜中から経過した完全な時間数、mm は時間の始めからの完全な分数、ss は分のはじめからの完全な秒数です。たとえば、'23:59:59' は真夜中より 1 秒前の時刻を表します。

ISO 8601 標準では、秒と分を省略できます。たとえば、'23:59' は真夜中より 60 秒前の時刻を表します。

ISO 8601 標準は、また、秒単位に小数を含めることができます。小数の秒は、カンマ (,) またはピリオド (.) を使って指定されます。秒の小数は、最大小数第 6 位まで格納されます。たとえば、'23:59:59,500000' と '23:59:59.500000' は、どちらも真夜中より 1/2 秒前の時刻を表します。分または時間の小数はサポートされません。

ISO 8601 は、時刻が日付仕様に含まれるときには、コロン区切り文字を必要としません。たとえば、'235959' は真夜中より 1 秒前の時刻を表します。

日の最大時刻は '24:00:00' です。これは真夜中を表します。日付と組み合わせると、これは真夜中、または次の日の 00:00:00 を表します。たとえば、'2010-04-01 24:00:00' は '2010-04-02 00:00:00' と同じです。

ISO 時刻	形式	例
基本 (日付あり)	hhmmss.ssssss	20100401 235959.500000
基本 (日付あり)	hhmmss,ssssss	20100401 235959,500000
拡張	hh:mm:ss.ssssss	23:59:59.500000
拡張	hh:mm:ss,ssssss	23:59:59,500000

ISO 以外の AM と PM の指示子もサポートされています。たとえば、'11:59:59 PM' は '23:59:59' と同じです。

AM/PM	形式	例
AM	hh:mm:ss.ssssss AM	11:59:59.500000 AM
AM	hh:mm:ss,ssssss AM	11:59:59,500000 AM
PM	hh:mm:ss.ssssss PM	11:59:59.500000 PM
PM	hh:mm:ss,ssssss PM	11:59:59,500000 PM

関連情報

日付と時刻の比較 [189 ページ]
DATE データ型 [161 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.2.5.2.3 時刻付きの日付形式

ISO 8601 では、日付と時刻で、スペースまたは文字 T を使用して組み合わせることができます。

たとえば、'2010-04-01 23:59:59' と '2010-04-01T23:59:59' はどちらも 2010 年 4 月 1 日の真夜中よりも 1 秒前を表します。ハイフンとコロンの区切り文字は省略できます。たとえば、'20100401T235959' も同じ日付と時刻を表します。また、この形式の拡張機能として、日付/時刻セパレータの省略もサポートされています。たとえば、'20100401235959' も上と同じ日付と時刻を表します。

日付の時刻の形式の基本と拡張の組み合わせがサポートされます。たとえば、'20100401T23:59:59' は基本形式と拡張形式の両方を組み合わせています。

関連情報

日付と時刻の比較 [189 ページ]
DATE データ型 [161 ページ]

TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.2.5.2.4 タイムゾーン形式

ISO 8601 は、また、日文字列の日付と時刻にタイムゾーンオフセットを追加することを許可します。

形式は次のいずれかになります。

Z

(ズールー) 日付と時刻は、協定世界時 (UTC: Coordinated Universal Time) 単位になります。たとえば、'2010-04-01 23:00:00Z' は、2010 年 4 月 1 日の協定世界時 11:00 PM を表します。

+hh:mm

指定された日付と時刻は、UTC より、指定された時間と分だけ進んでいます。たとえば、'2010-04-01 23:00:00+04:00' は、UTC から 4 時間東のタイムゾーンでの 2010 年 4 月 1 日の 11:00 PM を表します。

-hh:mm

指定された日付と時刻は、UTC より、指定された時間と分だけ遅れています。たとえば、'2010-04-01 23:00:00-05:00' は、UTC から 5 時間西のタイムゾーンでの 2010 年 4 月 1 日の 11:00 PM を表します。

分が 0 の場合、タイムゾーンオフセットで指定する必要はありません。また、スペースをタイムゾーンオフセットの前に含めることもできます。たとえば、'2010-04-01 23:00:00 -03:30' は、UTC から 3 時間半西のタイムゾーンでの 2010 年 4 月 1 日の 11:00 PM を表します。

ISO タイムゾーン	形式	例
基本	Z	20100401 235959Z
基本	+hhmm	20100401 235959+0400
基本	+hh	20100401 235959+04
基本	-hhmm	20100401 235959-0500
基本	-hh	20100401 235959-05
Basic	T を小数に使用	20100401T235959.50-0330
拡張	Z	2010-04-01 23:59:59Z
拡張	+hh:mm	2010-04-01 23:59:59+04:00
拡張	-hh:mm	2010-04-01 23:59:59-05:00
拡張	T を小数に使用	2010-04-01T23:59:59.50-03:30

日付、時刻、タイムゾーンの形式の基本と拡張の組み合わせがサポートされます。たとえば、'20100401T23:59:59-05' は基本形式と拡張形式の両方を組み合わせています。

関連情報

日付と時刻の比較 [189 ページ]
DATE データ型 [161 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.2.5.3 日付と時刻のデータベースからの取得

日付と時刻は、どのインタフェースを使用してもデータベースから文字列として取得できます。さらに、ODBC または OLE DB を使用してバイナリ値として取得したり (たとえば、ODBC TIMESTAMP_STRUCT 構造を使用)、Embedded SQL を使用して SQLDATETIME 構造として取得したりすることもできます。

タイムゾーンオフセット付きの日付と時刻は、データベースから文字列としてのみ取得できます。

(タイムゾーンオフセット付きかどうかにかかわらず) 日付または時刻を文字列として検索する場合は、データベースオプション date_format、time_format、timestamp_format、および timestamp_with_time_zone_format によって指定されている形式で検索します。

日付については次の算術演算子を使います。

timestamp + integer

日付またはタイムスタンプに指定日数を加算します。
timestamp - integer

日付またはタイムスタンプから指定日数を減算します。
date - date

2 つの日付またはタイムスタンプの間の日数を計算します。
date + time

与えられた日付と時刻を結合するタイムスタンプを作成します。

閏年

データベースサーバでは、広く認められているアルゴリズムを使用して閏年の決定を行います。このアルゴリズムを使用すると、4 で割り切れる年は閏年と見なされます。1900 年などの世紀にあたる年の場合は、400 で割り切れれば閏年となります。

すべての閏年が適切に処理されます。たとえば、次の SQL 文は "火曜日" という値を返します。

```
SELECT DAYNAME('2000-02-29');
```

データベースサーバは閏年 2000 年 2 月 29 日を日付として認め、この日付を考慮して曜日を判断します。

ただし、次の文はデータベースサーバでは拒否されます。

```
SELECT DAYNAME('2001-02-29');
```

エラーメッセージ「値 '2001-02-29' を日付に変換できません」が表示されます。これは、2 月 29 日が 2001 年には存在しないためです。

関連情報

日付および時刻関数 [204 ページ]
SET OPTION 文 [1310 ページ]

1.2.5.4 DATE データ型

DATE データ型は、年、月、日などの暦日を格納します。

構文

DATE

備考

DATE 値には 4 バイトの記憶領域が必要です。

アプリケーションが文字列として DATE 値を取得するときのフォーマットは、date_format オプションの設定によって決まります。たとえば、2010 年 7 月 19 日を表す DATE 値は、date_format オプション設定に従い、アプリケーションに 2010/07/19 または Jul 19, 2010 として返されます。

Ultra Light: アプリケーションが文字列として DATE 値を取得するときのフォーマットは、date_format 作成パラメータによって決まります。たとえば、2010 年 7 月 19 日を表す DATE 値は、date_format 作成パラメータ設定に従い、アプリケーションに 2010/07/19 または Jul 19, 2010 として返されます。

標準

ANSI/ISO SQL 標準

標準に含まれる機能。

Transact-SQL

Adaptive Server Enterprise によってサポートされます。

関連情報

日付および時刻関数 [204 ページ]
日付形式 [155 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.2.5.5 DATETIME データ型

DATETIME は、日付と時刻の情報を格納します。

構文

DATETIME

備考

DATETIME は Transact-SQL の型です。

Ultra Light: DATETIME はドメインであり、TIMESTAMP として実装されます。

アプリケーションが文字列として DATETIME 値を取得するときの形式は、timestamp_format オプションの設定によって決まります。たとえば、DATETIME 値 2010/04/01T23:59:59.999999 は、timestamp_format オプション設定に従い、アプリケーションに 2010/04/01 23:59:59 または April 1, 2010 23:59:59.999999 として返されます。

DATETIME 値には 8 バイトの記憶領域が必要です。

DATETIME データ型では、DATE 型と同じく 0001 ～ 9999 年の日付を格納でき、さらに 1600-02-28 23:59:59 ～ 7911-01-01 00:00:00 も格納できます。DATETIME 値では、この範囲より前または後の時間および分の部分は保持されません。

注記

DATETIME 値の精度が下がると、分または秒に関連する組み込み関数が無効な結果を生成します。

DATETIME 値を DATETIMEOFFSET に変換する場合、接続の time_zone_adjustment 設定が結果のタイムゾーンオフセットに使用されます。つまり、値はその接続に "ローカル" なものであると見なされます。DATETIMEOFFSET 値を DATETIME に変換すると、オフセットは廃棄されます。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

Adaptive Server Enterprise によって使用されるのは、TIMESTAMP ではなく DATETIME です。Adaptive Server Enterprise の DATETIME 型では、1753 年 1 月 1 日から 9999 年 12 月 31 日までの日付がサポートされ、値の時刻の部分でサポートされている精度はより低くなっています。SQL Anywhere では、DATETIME は、これらの制約なしに TIMESTAMP として実装されます。SQL Anywhere と Adaptive Server Enterprise の間でデータを移行する場合は、これらの違いに注意してください。

関連情報

日付および時刻関数 [204 ページ]
日付と時刻をデータベースに送信する方法 [154 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATE データ型 [161 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.2.5.6 DATETIMEOFFSET データ型

DATETIMEOFFSET データ型は TIMESTAMP WITH TIME ZONE のエイリアスであり、日付、時刻、およびタイムゾーン情報の格納に使用されます。

構文

DATETIMEOFFSET

備考

DATETIMEOFFSET 値は、年、月、日、時、分、秒、秒の小数位、および協定世界時 (UTC: Coordinated Universal Time) との差異を示す分数を含んでいます。秒以下は 6 桁まで格納されます。

アプリケーションが文字列として DATETIMEOFFSET 値を取得するときの形式は、timestamp_with_time_zone_format オプションの設定によって決まります。たとえば、DATETIMEOFFSET 値 2010/04/01T23:59:59.999999-6:00 は、timestamp_with_time_zone_format オプション設定に従い、アプリケーションに 2010/04/01 23:59:59 -6:00 または April 1, 2010 23:59:59.999999 -06:00 として返されます。

DATETIMEOFFSET 値には 10 バイトの記憶領域が必要です。

DATETIMEOFFSET データ型では、DATE 型と同じく 0001 ～ 9999 年の日付を格納でき、さらに 1600-02-28 23:59:59 ～ 7911-01-01 00:00:00 も格納できます。DATETIMEOFFSET 値では、この範囲より前または後の時間および分の部分は保持されません。

DATETIMEOFFSET は計算カラムまたはマテリアライズドビューには使用しないでください。時刻を制御する time_zone_adjustment オプションの値が、場所に基づく接続と日付によって異なるためです。

2 つの DATETIMEOFFSET 値が UTC において同じ時刻を表す場合、適用されている TIME ZONE オフセットに関係なく同じ時刻と見なされます。たとえば、次の文では、結果が同じと見なされるため、Yes が返されます。

```
IF CAST('2009-07-15 08:00:00 -08:00' AS DATETIMEOFFSET) =
CAST('2009-07-15 11:00:00 -05:00' AS DATETIMEOFFSET) THEN
SELECT  'Yes'
ELSE
SELECT 'No'
END IF;
```

DATETIMEOFFSET 値からタイムゾーンオフセットを省略すると、タイムスタンプが日付と時刻を標準時間または夏時間のいずれで表していても、クライアントの現在の UTC オフセットがデフォルト設定されます。たとえば、クライアントが東部標準タイムゾーンに属しており、夏時間が有効なときに次の文を実行した場合、大西洋標準タイムゾーン (UTC から -4 時間) に該当するタイムゾーンでタイムスタンプが返されます。

```
SELECT CAST('2009/01/30 12:34:55' AS DATETIMEOFFSET);
```

DATETIMEOFFSET 値をタイムゾーン付きではないタイムスタンプと比較することはお奨めしません。これは、クライアントのデフォルトのタイムゾーンオフセットは、クライアントの地理的な場所および日付によって異なるためです。

クライアントの現在のタイムゾーンオフセットを分単位で判別するには、次の文を実行します。

```
SELECT CONNECTION_PROPERTY( 'TimeZoneAdjustment' );
```

> **注記**
>
> TimeZoneAdjustment 接続プロパティは、Ultra Light データベースではサポートされません。

標準

ANSI/ISO SQL 標準

DATETIMEOFFSET の特定の使用方法は、標準にはありません。ANSI/ISO SQL 標準との互換性が必要な場合は、TIMESTAMP WITH TIME ZONE を使用します。TIMESTAMP WITH TIME ZONE 型は、オプションの ANSI/ISO SQL 言語機能 F411 です。

関連情報

日付および時刻関数 [204 ページ]
日付と時刻をデータベースに送信する方法 [154 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATETIME データ型 [162 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.2.5.7 SMALLDATETIME データ型

SMALLDATETIME は、TIMESTAMP として実装されるドメインで、日付と時刻の情報を格納するために使用します。SMALLDATETIME は Transact-SQL の型です。

構文

SMALLDATETIME

備考

なし

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

SMALLDATETIME は Adaptive Server Enterprise によってサポートされます。Adaptive Server Enterprise の SMALLDATETIME 型では、1900 年 1 月 1 日から 2079 年 6 月 6 日までの日付がサポートされ、値の時刻の部分でサポートされている精度はより低くなっています。SQL Anywhere では、SMALLDATETIME は、これらの制約なしに TIMESTAMP として実装されます。SQL Anywhere と Adaptive Server Enterprise の間でデータを移行する場合は、これらの違いに注意してください。

関連情報

日付と時刻をデータベースに送信する方法 [154 ページ]
日付および時刻関数 [204 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATETIME データ型 [162 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]

ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.2.5.8 TIME データ型

TIME データ型は、時、分、秒、秒以下で構成される時刻を格納します。

構文

TIME

備考

TIME 値には 8 バイトの記憶領域が必要です

ODBC を使用するときに、(ODBC TIME_STRUCT 構造体を使用して) バイナリ値として送信または取得される TIME 値は、時、分、秒の精度に制限されます。秒の小数部分は構造体には含まれません。そのため、精度を高めることを希望する場合は、TIME 値を文字列として送信または取得してください。アプリケーションが文字列として TIME 値を取得するときのフォーマットは、time_format オプションの設定によって決まります。たとえば、TIME 値 23:59:59.999999 は、time_format オプションの設定に従い、23:59:59、23:59:59.999、または 23:59:59.999999 としてアプリケーションに返されます。

Ultra Light: アプリケーションが文字列として TIME 値を取得するときの形式は、time_format 作成パラメータによって決まります。たとえば、TIME 値 23:59:59.999999 は、time_format 作成パラメータの設定に従い、23:59:59、23:59:59.999、または 23:59:59.999999 としてアプリケーションに返されます。

標準

ANSI/ISO SQL 標準

標準に含まれる機能。
Transact-SQL

TIME データ型は Adaptive Server Enterprise によってサポートされます。ただし、Adaptive Server Enterprise では、マイクロ秒の精度 (6 桁) ではなく、ミリ秒の精度 (3 桁) がサポートされます。SQL Anywhere と Adaptive Server Enterprise の間でデータを移行する場合は、これらの違いに注意してください。TIME 値を移行するには、Adaptive Server Enterprise の BIGTIME データ型を使用します。

関連情報

時間形式 [157 ページ]
日付および時刻関数 [204 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATETIME データ型 [162 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.2.5.9 TIMESTAMP データ型

TIMESTAMP データ型は、年、月、日、時、分、秒、秒以下 (小数第 6 位まで) で構成される時刻を格納します。

構文

TIMESTAMP

備考

TIMESTAMP 値には 8 バイトの記憶領域が必要です。

アプリケーションが文字列として TIMESTAMP 値を取得するときのフォーマットは、timestamp_format オプションの設定によって決まります。たとえば、TIMESTAMP 値 2010/04/01T23:59:59.999999 は、timestamp_format オプション設定に従い、アプリケーションに 2010/04/01 23:59:59 または April 1, 2010 23:59:59.999999 として返されます。

Ultra Light: アプリケーションが文字列として TIMESTAMP 値を取得するときのフォーマットは、timestamp_format 作成パラメータによって決まります。たとえば、TIMESTAMP 値 2010/04/01T23:59:59.999999 は、timestamp_format 作成パラメータに従い、アプリケーションに 2010/04/01 23:59:59 または April 1, 2010 23:59:59.999999 として返されます。

TIMESTAMP データ型では、DATE 型と同じく 0001 ～ 9999 年の日付を格納できますが、実質的に有効な範囲は 1600-02-28 23:59:59 ～ 7911-01-01 00:00:00 です。TIMESTAMP 値では、この範囲より前または後の時間および分の部分は保持されません。

> **注記**
>
> TIMESTAMP 値の精度が下がると、分または秒に関連する組み込み関数が無効な結果を生成します。

TIMESTAMP 値を TIMESTAMP WITH TIME ZONE に変換する場合、その接続の time_zone_adjustment 設定が、結果のタイムゾーンオフセットに使用されます。つまり、値はその接続に "ローカル" なものであると見なされます。TIMESTAMP WITH TIME ZONE 値を TIMESTAMP に変換すると、オフセットは廃棄されます。

Ultra Light: TIMESTAMP 値を TIMESTAMP WITH TIME ZONE に変換する場合、システムのローカルタイムゾーンオフセットが最終結果に使用されます。

標準

ANSI/ISO SQL 標準

標準と互換性があります。

Transact-SQL

Adaptive Server Enterprise は、TIMESTAMP 値に対して DATETIME 型を使用します。

関連情報

日付と時刻をデータベースに送信する方法 [154 ページ]
日付および時刻関数 [204 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATETIME データ型 [162 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

UTC TIMESTAMP 特別値 [109 ページ]

1.2.5.10 TIMESTAMP WITH TIME ZONE データ型

TIMESTAMP WITH TIME ZONE データ型は、タイムゾーンオフセット付きの時刻を格納します。

構文

TIMESTAMP WITH TIME ZONE

備考

TIMESTAMP WITH TIME ZONE 値は、年、月、日、時、分、秒、秒の小数位、および協定世界時 (UTC: Coordinated Universal Time) との差異を示す分数を含んでいます。秒以下は 6 桁まで格納されます。

注記

デフォルトでは、TIMESTAMP WITH TIME ZONE 値はデータベースの現在のタイムゾーンです。

アプリケーションが文字列として TIMESTAMP WITH TIME ZONE 値を取得するときのフォーマットは、timestamp_with_time_zone_format オプションの設定によって決まります。たとえば、TIMESTAMP WITH TIME ZONE 値 2010/04/01T23:59:59.999999-6:00 は、timestamp_with_time_zone_format オプションの設定に従い、アプリケーションに 2010/04/01 23:59:59 -06:00 または April 1, 2010 23:59:59.999999 -06:00 として返されます。

Ultra Light: アプリケーションが文字列として TIMESTAMP WITH TIME ZONE 値を取得するときのフォーマットは、timestamp_with_time_zone_format 作成パラメータの設定によって決まります。たとえば、TIMESTAMP WITH TIME ZONE 値 2010/04/01T23:59:59.999999-6:00 は、timestamp_with_time_zone_format 作成パラメータに従い、アプリケーションに 2010/04/01 23:59:59 -06:00 または April 1, 2010 23:59:59.999999 -06:00 として返されます。

TIMESTAMP WITH TIME ZONE 値には 10 バイトの記憶領域が必要です。

TIMESTAMP WITH TIME ZONE データ型では、DATE 型と同じく 0001 ～ 9999 年の日付を格納でき、さらに 1600-02-28 23:59:59 ～ 7911-01-01 00:00:00 も格納できます。TIMESTAMP WITH TIME ZONE 値では、この範囲より前または後の時間および分の部分は保持されません。

計算カラムまたはマテリアライズドビューに TIMESTAMP WITH TIME ZONE を使用しないでください。時刻を制御する time_zone_adjustment オプションの値は、場所と日付に基づいて、接続によって異なるためです。

2 つの TIMESTAMP WITH TIME ZONE 値が UTC において同じ時刻を表す場合、適用されている TIME ZONE オフセットに関係なく同じ時刻と見なされます。たとえば、次の文では、結果が同じと見なされるため、Yes が返されます。

```
IF CAST('2009-07-15 08:00:00 -08:00' AS TIMESTAMP WITH TIME ZONE) =
    CAST('2009-07-15 11:00:00 -05:00' AS TIMESTAMP WITH TIME ZONE) THEN
  SELECT  'Yes'
ELSE
  SELECT 'No'
END IF;
```

Ultra Light:

```
SELECT IF CAST('2009-07-15 08:00:00 -08:00' AS TIMESTAMP WITH TIME ZONE) =
    CAST('2009-07-15 11:00:00 -05:00' AS TIMESTAMP WITH TIME ZONE)
    THEN 'Yes'
    ELSE 'No'
END IF;
```

TIMESTAMP WITH TIME ZONE 値からタイムゾーンオフセットを省略すると、タイムスタンプが日付と時刻を標準時間または夏時間のいずれで表していても、クライアントの現在の UTC オフセットがデフォルト設定されます。たとえば、クライアントが東部標準タイムゾーンに属しており、夏時間が有効なときに次の文を実行した場合、大西洋標準タイムゾーン (UTC から -4 時間) に該当するタイムゾーンでタイムスタンプが返されます。

```
SELECT CAST('2009/01/30 12:34:55' AS TIMESTAMP WITH TIME ZONE)
```

TIMESTAMP WITH TIME ZONE とその他のデータ型の比較

TIMESTAMP WITH TIME ZONE 値をタイムゾーン付きではないタイムスタンプと比較することはおすすめしません。これは、クライアントのデフォルトのタイムゾーンオフセットは、クライアントの地理的な場所および日付によって異なるためです。

クライアントの現在のタイムゾーンオフセットを分単位で判別するには、次の文を実行します。

```
SELECT CONNECTION_PROPERTY( 'TimeZoneAdjustment' );
```

TIMESTAMP WITH TIME ZONE とその他のデータ型の間の変換

TIMESTAMP 値を TIMESTAMP WITH TIME ZONE に変換する場合、その接続の time_zone_adjustment 設定が、結果のタイムゾーンオフセットに使用されます。つまり、値はその接続に "ローカル" なものであると見なされます。
TIMESTAMP WITH TIME ZONE 値を TIMESTAMP に変換すると、オフセットは廃棄されます。文字列型、日付型、または時刻型以外の型との間の変換はサポートされていません。

Ultra Light: TIMESTAMP 値を TIMESTAMP WITH TIME ZONE に変換する場合、クライアントのタイムゾーンが結果のタイムゾーンオフセットに使用されます。つまり、値はその接続に "ローカル" なものであると見なされます。
TIMESTAMP WITH TIME ZONE 値を TIMESTAMP に変換すると、オフセットは廃棄されます。文字列型、日付型、または時刻型以外の型との間の変換はサポートされていません。

標準

ANSI/ISO SQL 標準

TIMESTAMP WITH TIME ZONE 文は、オプションの ANSI/ISO SQL 言語機能 F411 の一部です。

関連情報

日付と時刻の比較 [189 ページ]
日付と時刻をデータベースに送信する方法 [154 ページ]
日付および時刻関数 [204 ページ]
CURRENT TIME 特別値 [90 ページ]

CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATETIME データ型 [162 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.2.6 バイナリデータ型

バイナリデータ型は、データベースに認識されないイメージやその他の種類の情報を含めたバイナリデータを格納します。

このセクションの内容:

BINARY データ型 [173 ページ]
BINARY データ型は、指定された最大長 (バイト単位) のバイナリデータを格納します。

IMAGE データ型 [174 ページ]
IMAGE データ型は、任意の長さのバイナリデータを格納します。

LONG BINARY データ型 [174 ページ]
LONG BINARY データ型は、任意の長さのバイナリデータを格納します。

UNIQUEIDENTIFIER データ型 [175 ページ]
UNIQUEIDENTIFIER データ型は、UUID (GUID とも呼ばれる) の値を格納します。

VARBINARY データ型 [177 ページ]
VARBINARY データ型は、指定された最大長 (バイト単位) のバイナリデータを格納します。

1.2.6.1 BINARY データ型

BINARY データ型は、指定された最大長 (バイト単位) のバイナリデータを格納します。

構文

```
BINARY [ ( max-length ) ]
```

パラメータ

max-length

値のバイト単位での最大長。長さを指定しない場合、値は 1 になります。

長さは 1 ～ 32767 の範囲内で指定する必要があります。

備考

比較時に、BINARY 値はバイトごとに正確に比較されます。この比較方法は CHAR データ型とは異なります。CHAR の場合、データベースの照合順を使用して値が比較されます。

1 つのバイナリ文字列が他の文字列のプレフィクスである場合、短い文字列は長い文字列よりも文字数が少ないと見なされます。

CHAR 値とは異なり、文字セットの変換時に BINARY 値は変換されません。

セマンティック上、BINARY は VARBINARY と同じです。可変長型です。他のデータベース管理システムでは、BINARY は固定長型です。

Ultra Light: BINARY はドメインであり、VARBINARY として実装されます。

標準

ANSI/ISO SQL 標準

SQL 言語機能 T021。

関連情報

文字列関数 [213 ページ]
VARBINARY データ型 [177 ページ]

LONG BINARY データ型 [174 ページ]
ビット処理演算子 [31 ページ]

1.2.6.2 IMAGE データ型

IMAGE データ型は、任意の長さのバイナリデータを格納します。

構文

IMAGE

備考

IMAGE は LONG BINARY として実装されるドメインです。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

文字列関数 [213 ページ]
LONG BINARY データ型 [174 ページ]

1.2.6.3 LONG BINARY データ型

LONG BINARY データ型は、任意の長さのバイナリデータを格納します。

構文

LONG BINARY

備考

最大サイズは 2 GB から 1 バイトを差し引いた値です ($2^{31} - 1$)。

LONG BINARY データに、または LONG BINARY データから文字列をキャストできます。LONG BINARY データを連結することはできません。

SQL 文の条件 (WHERE 句の条件など) は、LONG BINARY カラムでは実行できません。LONG BINARY カラムでは、INSERT、UPDATE、DELETE の各操作が許可されます。LONG BINARY カラムは、SELECT クエリの結果セットに含めることもできます。

Ultra Light: LONG BINARY 型に対してインデックスを作成することはできません。LONG BINARY 型は LENGTH 関数と CAST 関数でのみ使用できます。

標準

ANSI/ISO SQL 標準

LONG BINARY データ型は、SQL 言語機能 T021 "BINARY and VARBINARY data types" と T041 "Basic LOB data type support" によって構成されています。

関連情報

BINARY データ型 [173 ページ]
VARBINARY データ型 [177 ページ]

1.2.6.4 UNIQUEIDENTIFIER データ型

UNIQUEIDENTIFIER データ型は、UUID (GUID とも呼ばれる) の値を格納します。

構文

UNIQUEIDENTIFIER

備考

UNIQUEIDENTIFIER データ型は、通常は、ローをユニークに識別する UUID (ユニバーサルユニーク識別子) 値を保持するために、プライマリキーまたはその他のユニークカラムに使用されます。NEWID 関数では、あるコンピュータで生成される UUID 値が他のコンピュータで生成される UUID と一致しないように UUID 値が生成されます。したがって、NEWID を使用して生成された UNIQUEIDENTIFIER 値は、同期環境でキーとして使用できます。

例:

```
CREATE TABLE T1 (
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    c1 INT );
```

Ultra Light の場合:

```
CREATE TABLE T1 (
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    c1 INT );
```

UUID 値は、GUID (グローバルユニーク識別子) 値とも呼ばれます。UUID 値にはハイフンが含まれます。これは他の RDBMS と互換性を持たせるためです。

このデフォルト設定を変更するには、uuid_has_hyphens オプションを Off にします。

Ultra Light: このデフォルト設定を変更するには UUIDTOSTR 関数と STRTOUUID 関数を使用します。

UNIQUEIDENTIFIER 値は、必要に応じて文字列値とバイナリ値の間で自動的に変換されます。

UNIQUEIDENTIFIER 値は、BINARY(16) として格納されますが、クライアントアプリケーションには BINARY(36) として示されます。このため、クライアントが値を文字列としてフェッチした場合に、結果に対して十分な領域が割り付けられることが保証されます。

SQL Anywhere の場合、ODBC クライアントアプリケーションでは、uniqueidentifier 値が SQL_GUID 型として示されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

文字列関数 [213 ページ]
NEWID 関数 [その他] [447 ページ]
UUIDTOSTR 関数 [文字列] [584 ページ]
STRTOUUID 関数 [文字列] [548 ページ]

1.2.6.5 VARBINARY データ型

VARBINARY データ型は、指定された最大長 (バイト単位) のバイナリデータを格納します。

構文

```
VARBINARY [ ( max-length ) ]
```

パラメータ

max-length

値のバイト単位での最大長。長さを指定しない場合、値は 1 になります。

長さは 1 ～ 32767 の範囲内で指定する必要があります。

備考

比較時に、VARBINARY 値はバイトごとに正確に比較されます。この比較方法は CHAR データ型とは異なります。CHAR の場合、データベースの照合順を使用して値が比較されます。

VARBINARY 値は、文字セットの変換時に変換されません。

VARBINARY は BINARY VARYING と指定することもできます。どの構文を使用する場合でも、データ型は VARBINARY と記述されます。1 つのバイナリ文字列が他の文字列のプレフィクスである場合、短い文字列は長い文字列よりも文字数が少ないと見なされます。

Ultra Light: 1 つのバイナリ文字列が他の文字列のプレフィクスである場合、他の文字列と比較するときに他よりも短い文字列はゼロ (0) が埋め込まれていると見なされます。式を評価するときのテンポラリ文字値の最大長は 2048 バイトです。

標準

ANSI/ISO SQL 標準

SQL 言語機能 T021、"BINARY and VARBINARY data types"。

関連情報

文字列関数 [213 ページ]
BINARY データ型 [173 ページ]

LONG BINARY データ型 [174 ページ]
ビット処理演算子 [31 ページ]

1.2.7 複合データ型 ROW および ARRAY

複合データ型は、0 個以上の要素で構成される値であり、要素ごとに特定のデータ型の値が含まれます。現在のところ、複合データ型 ROW および ARRAY のみがサポートされています。

複合データ型は固有のデータ型ではありません。1 つ以上のデータ型をローに組み合わせる方法の定義を含むコンストラクタです。ROW 型または ARRAY 型を定義する際、ロー (配列は一連のローです) を構成する各要素の数値型とデータ型を定義します。

ROWS と ARRAYS を使用すると、リストの構造やその値のデータ型が定義されるため、より効率的にリストを保管できます。また、UNNEST 演算子を使用することで、リスト要素の作成を容易にします。この場合、直接作成、二重角括弧を使用した作成、または結果セットとしての作成が可能です。VARCHAR カラム内の区切られた文字列としてリストを保管し、sa_split_list で解析する場合は、ARRAY データ型の使用を検討してください。ARRAYS は、それぞれの間に関連のあるさまざまなオブジェクトを保管する場合に便利です。ROWS は、1 つのオブジェクトと関連する複数の値を保管する場合に便利です。

ROW 型または ARRAY 型のドメインの作成

ROW 型または ARRAY 型のドメインを作成できます。たとえば、次の文は、MyRow と呼ばれるドメインを作成し、その複合型を 2 つの整数値として定義します。

```
CREATE DOMAIN MyRow ROW( a INT, b INT );
```

ROW ドメインまたは ARRAY ドメインを作成すると、文の内部で繰り返し定義する代わりに、ローまたは配列の型を参照できます。

ARRAY 型の宣言

ARRAY 型は同種の順序付けされたコレクションであり、その全部または一部を SQL ストアドプロシージャまたはファンクションの引数として渡すことができます。

ARRAY 型は最大 640 万個の要素で構成できます。1 つの ARRAY は宣言されたタイプの長さ 0 の ARRAY に初期化され、各要素は NULL になります。

ARRAY コンストラクタが ARRAY 値を作成するため、その配列はクエリで処理したり、SQLL ストアドプロシージャまたはファンクションの引数として渡すことができます。

データベースサーバは ARRAY 型の値を宣言するために、次の構文をサポートしています。

```
DECLARE variable-name element-type-name ARRAY [ ( maximum-size ) ]
```

```
DECLARE variable-name ARRAY [ ( maximum-size ) ] OF element-type-name
```

maximum-size を省略した場合、その配列には最大 640 万個の要素を含めることができます。ARRAY 型の変数は、他のすべての非複合型の変数のように NULL に初期化されません。その代わり、長さ 0 の配列に初期化されます。

ARRAY 型はベーステーブルまたはテンポラリテーブルにカラムとして格納できないため、以下ではサポートされません。

- ビュー定義の最も外側の SELECT リスト
- クライアントに返される最上位レベルの SELECT ブロックまたはクエリ式
- ベーステーブル
- テンポラリテーブル
- Embedded SQL の FETCH 文

> **注記**
>
> プロシージャが結果セットで ROW 型または ARRAY 型のカラムを返した場合、プロシージャには RESULT カラムが必要です。ない場合はエラーが返されます。必要に応じて、Watcom SQL 構文を使用するようにプロシージャを変更します。

次の例は、5 つの整数の配列を宣言する方法を示しています。

```
DECLARE NewArray INTEGER ARRAY( 5 );
```

次の例は、Oracle の構文と互換性のある 5 つの整数の配列を宣言する別の方法を示しています。

```
DECLARE NewArray ARRAY( 5 ) OF INTEGER;
```

次の例は、New2DArray に 10 個の要素が含まれ、それぞれが整数の 5 要素配列となる 2 次元配列を宣言する方法を示しています。

```
DECLARE New2DArray INTEGER ARRAY( 5 ) ARRAY( 10 );
```

次の例は、New2DArray に 10 個の要素が含まれ、それぞれが整数の 5 要素配列となる、Oracle の構文と互換性のある 2 次元配列を宣言する別の方法を示しています。

```
DECLARE New2DArray ARRAY( 10 ) OF ARRAY( 5 ) OF INTEGER;
```

ROW 型の宣言

ROW 型は 1 つのロー型記述子によって表され、その記述子は ROW 型の各フィールドのフィールド記述子で構成されます。ROW 型は 45000 フィールドに制限されます (テーブル内のカラム数と同じ制限)。

ROW 型の変数は宣言されたタイプの ROW に初期化され、各フィールドは NULL に初期化されます。

ROW は、型が異なる可能性のあるフィールドのグループで構成される、構造化された型の構造をサポートしています。ROW 型は上位のロー型の一部となる可能性があり、したがって他のロー型や配列を含む複雑な構造体となることができます。

次の例は、変数 student を宣言し、4 つの異なるフィールドを持つ構造化された型として定義する方法を示しています。

```
DECLARE student ROW( studentID INTEGER,
    student_first_name VARCHAR( 40 ),
    student_last_name VARCHAR( 50 ),
    student_address LONG VARCHAR );
```

次の例は、ROW を完全な構造体として割り当てる方法を示しています。

```
DECLARE employee ROW( empID INTEGER,
    address ROW( street_address LONG VARCHAR,
        city VARCHAR( 50 ),
        province VARCHAR( 30 ),
        country VARCHAR( 40 )
    )
);
DECLARE temp_address ROW( street_address LONG VARCHAR,
    city VARCHAR( 50 ),
    province VARCHAR( 30 ),
    country VARCHAR( 40 )
);
SET temp_address = employee.address;
```

関連情報

ARRAY コンストラクタ [複合] [234 ページ]
ROW コンストラクタ [複合] [510 ページ]
UNNEST 配列演算子 [26 ページ]

1.2.8 TABLE REF データ型

TABLE REF データ型は、ベーステーブル、テンポラリテーブル、またはビューへの参照を格納します。このデータ型は、接続スコープ変数でのみ使用できます。

構文

TABLE REF 型の変数の宣言

```
DECLARE table-ref-variable TABLE REF
  [ { DEFAULT | = } TABLE REF ( [ owner.]table-name ) ]
```

TABLE REF 型の変数の作成

```
CREATE VARIABLE table-ref-variable TABLE REF
  [ { DEFAULT | = } TABLE REF ( [ owner.]table-name ) ]
```

TABLE REF 型の変数の設定

```
SET table-ref-variable = TABLE REF ( table-name )
```

DML 文における TABLE REF 型の変数の参照

```
TABLE REF ( table-ref-variable ) AS correlation-name
```

関数またはプロシージャにおけるパラメータとしての TABLE REF 型の変数の参照

```
TABLE REF ( table-ref-variable )
```

パラメータ

table-ref-variable テーブル参照変数の有効な識別子。

table-name ベーステーブルの名前。オプションで、指定の一部として所有者を含めます (GROUPO.Employees など)。これは、ベーステーブルとビューに推奨されます。

備考

テーブル参照変数 (TABLE REF 型の変数) では、変更時に操作するテーブルの名前がまだ定義されていない場合でも、プロシージャと関数を定義することができます。

DML 文で TABLE REF 型の変数を参照するときは、結果の相関名を指定する必要があります。

ステートメントでテーブル参照変数を指定すると、ステートメントの実行前にテーブルがすぐに参照されます。

テーブル参照変数は、作成者だけがアクセスできます。

テーブル参照変数を作成しても、変数および基本となるテーブルの間の依存関係は作成されないため、テーブル参照変数により参照されるテーブルで引き続き DDL 文を作成できます。

テーブルが削除された場合、そのテーブルを参照するテーブル参照変数が無効になります。無効なテーブル参照変数を使用しようとすると、エラーが返されます。

テーブル参照変数を使用して指定されたテーブルで動作する文を実行するときは、変数により参照される基本となるテーブルでの適切な権限が必要です。

テーブル参照変数の使用には、次のような制限があります。

- 変数が NULL 値に解決される場合、テーブル参照変数を SELECT 文または DML 文で使用することはできません。
- テーブル参照変数を使用して、DDL 文でテーブルを指定することはできません。
- テーブル参照変数を、ベーステーブル、テンポラリテーブル、またはビューでカラムとして使用することはできません。
- テーブル参照変数を、最上位の SELECT ブロック、またはクライアントに返されるクエリ式で使用することはできません。
- テーブル参照変数を、パラメータの共通のスーパータイプを必要とする組み込み関数内の他の型の変数と組み合わせることはできません。
- テーブル参照変数を並べ替えたり、計算または比較 (等号および不等号を除く) の一部として使用したりすることはできません。

テーブル参照変数の機能は、間接識別子機能と重複します。どちらも、テーブルを間接的に参照する方法です。ただし、テーブル参照は作成時に解決され、有効な参照のままです。一方、間接参照はその間接参照を参照する文が実行されたときに解決されるため、有効な参照ではなくなる可能性があります。

さらに、テーブル参照は、現在のコンテキストでアクセス可能なテーブルへのアクセスを提供することができますが、間接識別子はできません。たとえば、ベーステーブルと同じ名前のローカルテーブルがプロシージャにより作成された後、参照されるとします。ここで、プロシージャ内からベーステーブルを参照する必要があるとします。テーブルの間接識別子は、目的のものではないローカルテーブルに解決されます。代わりにベーステーブルを正確に識別するには、テーブル参照変数を使用します。次に例を示します。

```
CREATE OR REPLACE PROCEDURE PROC1()
BEGIN
    DECLARE LOCAL TEMPORARY TABLE myTab (x INTEGER, y INTEGER);
    DECLARE tab_ref TABLE REF = TABLE REF (myTab);
    INSERT INTO myTab VALUES (1,100), (2,200), (3,300);
```

```
    SELECT * FROM PROC2( tab_ref );
END;
CREATE OR REPLACE PROCEDURE PROC2( IN @tab_ref TABLE REF )
RESULT (v1 LONG VARCHAR, v2 INTEGER)
BEGIN
    DECLARE LOCAL TEMPORARY TABLE myTab ( pk INTEGER, val LONG VARCHAR );
    INSERT INTO myTab VALUES( 1, 'apple'), (3, 'pear'), (10, 'banana');
    SELECT val,T2.y FROM myTab T1 JOIN TABLE REF( @tab_ref ) AS T2 ON T2.x = T1.pk;
END;
CALL PROC1;
```

表 2: 結果:

v1	v2
apple	100
pear	300

PROC2 内の myTab テーブルは PROC1 で作成された myTab をシャドウ (非表示) にするため、PROC2 内で名前 myTab を使用してアクセス可能なテーブルのみローカルで myTab と宣言されます。テーブル参照 (`TABLE REF( @tab_ref )`) を使用すると、ジョイン先のオブジェクト (この例では、PROC1 で作成されたテーブル) がより正確に識別されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、テーブル参照変数 @ref を宣言して GROUPO.Employees テーブル参照に設定し、テーブル参照テーブルを使用していテーブルを照会します。

```
DECLARE @ref TABLE REF = TABLE REF ( GROUPO.Employees )
SELECT * FROM TABLE REF ( @ref ) AS T;
```

次の例は、@tableDefinition というテーブル参照変数を作成して GROUPO.Employees テーブル参照に設定し、テーブル参照テーブルを使用してテーブルから選択します。

```
CREATE VARIABLE @tableDefinition TABLE REF;
SET @tableDefinition = TABLE REF ( GROUPO.Employees );
SELECT * FROM TABLE REF ( @tableDefinition ) AS T;
```

次のコード巣にペットは、データ型 TABLE REF を持つ @myTableRefVariable1 という変数を宣言し、GROUPO.Employees テーブル参照に設定します。

```
DECLARE @myTableRefVariable1 TABLE REF;
SET @myTableRefVariable1 = TABLE REF ( GROUPO.Employees );
```

次の例は、TABLE REF 変数を作成して GROUPO.Employees テーブルに設定し、誕生日が 2 月の従業員のテーブル参照変数を照会します。

```
CREATE VARIABLE @myTableRefVariable2 TABLE REF;
```

```
SET @myTableRefVariable2 = TABLE REF ( GROUPO.Employees );
SELECT T.surname, T.givenname, T.birthdate FROM TABLE REF
( @myTableRefVariable2 ) AS T
    WHERE MONTH( T.birthdate ) = 2;
```

表 3: 結果

surname	givenname	birthdate
Davidson	Jo Ann	1957/02/17
Samuels	Peter	1968/02/28
Barker	Joseph	1969/02/14
Sterling	Paul	1950/02/27

次の例は、GROUPO.Employees テーブルを更新するために複数の文で使用されているテーブル参照変数 (@myTableRefVariable3) を示しています。DML 文でテーブル参照変数を使用してテーブルを指定する場合、相関名 (この例では T) が必要です。

```
CREATE VARIABLE @myTableRefVariable3 TABLE REF;
SET @myTableRefVariable3 = TABLE REF ( GROUPO.Employees );
UPDATE TABLE REF ( @myTableRefVariable3 ) AS T SET T.GivenName =
REPLACE( GivenName, 'Fran', 'Francis' );
DELETE FROM TABLE REF ( @myTableRefVariable3 ) AS T WHERE Surname = 'Holmes';
SELECT * FROM TABLE REF ( @myTableRefVariable3 ) AS T;
```

次の例は、プロシージャでテーブル参照変数を使用する方法を示しています。

```
CREATE OR REPLACE PROCEDURE leapday_births( IN @tab TABLE REF )
RESULT ( Surname person_name_t, GivenName person_name_t, Birthdate TIMESTAMP)
BEGIN
    SELECT Surname, GivenName, Birthdate FROM TABLE REF ( @tab ) AS T
        WHERE MONTH( Birthdate ) = 02 AND DAY( Birthdate ) = 29;
END;
CREATE VARIABLE @myTableRefVariable4 TABLE REF;
SET @myTableRefVariable4 = TABLE REF ( GROUPO.Employees );
CALL leapday_births(@myTableRefVariable4);
```

関連情報

SQL 変数 [118 ページ]
識別子 [6 ページ]
間接識別子 [8 ページ]

1.2.9 空間データ型

多くの空間データ型がサポートされます。これらのデータ型のマニュアルは、空間 SQL API のマニュアルと同じ場所にあります。

1.2.10 ドメイン

ドメインは、適切な位置に精度や小数点以下の桁数を含み、さらにオプションとしてデフォルト値や CHECK 条件などを含んでいる、組み込みデータ型のエイリアスです。ドメインには、通貨データ型のように事前に定義されたものもありますが、ユーザが独自のドメインを追加することもできます。

ドメインはユーザ定義データ型とも呼ばれ、データベース全体を通して、カラムを同じ NULL または NOT NULL 条件、同じ DEFAULT 設定、同じ CHECK 条件を持つ同じデータ型に自動的に定義できます。ドメインはデータベース全体の一貫性を高め、特定の種類のエラーを防止するのに役立ちます。

簡単なドメイン

ドメインを作成するには、CREATE DOMAIN 文を使用します。

次の文は street_address という名前の、35 文字の文字列のデータ型を作成します。

```
CREATE DOMAIN street_address CHAR( 35 );
```

CREATE DATATYPE は CREATE DOMAIN の代わりに使用できますが、お奨めしません。

ドメインを作成するには、CREATE DATATYPE または CREATE ANY OBJECT システム権限が必要です。データ型を一度作成すると、CREATE DOMAIN 文を実行したユーザ ID がそのデータ型の所有者となります。このデータ型はすべてのユーザが使用できます。他のデータベースオブジェクトとは異なり、所有者名をデータ型名の前に置きません。

カラムを定義する場合は、street_address データ型を他のデータ型とまったく同じように使用します。たとえば、2 つのカラムがある次のテーブルでは、2 番目のカラムが street_address カラムです。

```
CREATE TABLE twocol (
    ID INT,
    street street_address
);
```

次のようにしてドメインを削除することもできます。

```
DROP DOMAIN street_address;
```

この文を実行できるのは、このデータ型をデータベースのどのテーブルでも使用していないときのみです。使用されているドメインを削除しようとすると、エラーメッセージが表示されいます。

ドメインの制約とデフォルト

NULL 入力可や DEFAULT 値の設定など、カラムに関連する属性の多くをドメインに組み込むことができます。データ型上で定義されたカラムは、NULL 設定、CHECK 条件、DEFAULT 値を自動的に継承します。このため、データベース全体を通してカラムに同じような意味を持たせ、統一性をとることができます。

たとえば、SQL Anywhere サンプルデータベース内の多くのプライマリキーカラムは、ID 番号を保持する整数カラムです。次の文は、このようなカラムに有効なデータ型を作成します。

```
CREATE DOMAIN ID INT
NOT NULL
DEFAULT AUTOINCREMENT
CHECK( @col > 0 );
```

デフォルトでは、id データ型を使用して作成されたカラムには NULL 値を格納できません。このため、これらのカラムにはデフォルト値として自動的に増分された値が設定され、必ず整数を保持しています。`@col` 変数では、`col` の代わりに任意の ID を使用できます。

カラムに対して明示的に属性を指定すると、データ型の属性を上書きできます。id データ型を使用して作成されたカラムに明示的に NULL が許可されると、id データ型の設定にかかわらず、NULL を使用できます。

互換性

名前を付けた制約とデフォルト

ドメインをベースデータ型で作成し、オプションとして NULL または NOT NULL 条件、デフォルト値、CHECK 条件を含めます。名前付き制約と名前付きデフォルトはサポートしません。

データ型を作成する

sp_addtype システムプロシージャを使って、ドメインを追加したり、CREATE DOMAIN 文を使用したりできます。

関連情報

CREATE DOMAIN 文 [794 ページ]
DROP DOMAIN 文 [1035 ページ]

1.2.11 データ型の比較

データ型の異なる引数の間で比較 (= など) を行う場合、1 つのデータ型で比較操作ができるように、1 つ以上の引数を変換します。

規則によっては、変換に失敗したり、比較が予期しない結果になることがあります。そのような場合は、CAST または CONVERT を使用して引数の 1 つを明示的に変換してください。

引数を明示的に別のデータ型にキャストして、これらの変換規則を上書きすることができます。たとえば、DATE と CHAR を CHAR として比較するには、DATE を CHAR に明示的にキャストします。

このセクションの内容:

関連情報

CAST 関数 [データ型変換] [264 ページ]

1.2.11.1 損失を伴う変換および置換文字

変換後の文字セットで文字を表現できない場合は、代わりに置換文字が使用されます。この種の変換は、損失を伴うと考えられます。つまり、変換後の文字セットで元の文字を表現できない場合は、その文字が失われます。

また、文字セットによって置換文字が異なる場合があるだけでなく、ある文字セットの置換文字が別の文字セットでは非置換文字であることもあります。1 文字に対して複数の変換が実行されるときには、このことを理解することが重要になります。最終的な文字が、変換後の文字セットで期待される置換文字として表示されない可能性があるためです。

たとえば、クライアントの文字セットが Windows-1252 で、データベースの文字セットが ISO_8859-1:1987 (一部のバージョンの UNIX における米国のデフォルト) であるとします。次に、非 Unicode のクライアントアプリケーション (たとえば Embedded SQL) がユーロ記号を CHAR、VARCHAR、または LONG VARCHAR カラムに挿入しようとします。この文字は CHAR 文字セットに存在しないため、ISO_8859-1:1987 の置換文字である 0x1A が挿入されます。

同じ ISO_8859-1:1987 置換文字が Unicode としてフェッチされる (たとえば ODBC で `SELECT * FROM t` を SQL_C_WCHAR にバインドされたカラムにフェッチする) 場合、この文字は Unicode のコードポイント U+001A になります (Unicode では、コードポイント U+001A はレコードセパレータ制御文字です)。ただし、Unicode の置換文字はコードポイント U+FFFD です。この例は、データに置換文字が含まれる場合であっても、複数回の変換が原因で、これらの文字が変換後の文字セットの置換文字に変換されない可能性があることを示しています。

このため、複数の文字セット間で変換を行う場合は、置換文字がどのように使用されるかを理解してテストすることが重要です。

on_charset_conversion_failure オプションを使用すると、文字を変換後の文字セットで表現できない場合の変換時の動作を指定しやすくなることがあります。

関連情報

データ型変換 [192 ページ]
CHAR と NCHAR の比較 [187 ページ]

1.2.11.2 CHAR と NCHAR の比較

CHAR 型 (CHAR、VARCHAR、LONG VARCHAR) の値と NCHAR 型 (NCHAR、NVARCHAR、LONG NVARCHAR) の値との間で比較を実行する場合は、比較を実行する型を判断するために推定規則が使用されます。

一般的に、一方の値のみがカラム参照に基づいている場合は、カラム参照が含まれている値の型を使用して比較が実行されます。

推定規則では、値がカラム参照に基づいているかどうかが重要です。一方の値が変数、ホスト変数、またはリテラル定数であるか、またはカラム参照に基づいていない複雑な式であり、かつもう一方の値がカラム参照に基づいている場合は、暗黙的に、定数に基づく値がカラムに基づく値にキャストされます。

次に、推定規則を適用順に示します。

- NCHAR 値がカラム参照に基づいている場合は、暗黙的に CHAR 値が NCHAR にキャストされ、NCHAR として比較されます。NCHAR 値と CHAR 値の両方がカラム参照に基づいている場合も同様です。
- NCHAR 値がカラム参照に基づいておらず、かつ CHAR 値がカラム参照に基づいている場合は、暗黙的に NCHAR 値が CHAR にキャストされ、CHAR として比較されます。
  NCHAR から CHAR への変換が予想される場合は、on_charset_conversion_failure オプションの設定を検討することが重要です。このオプションでは、NCHAR 文字を CHAR 文字セットで表せないときの動作を制御するからです。
- どちらの値もカラム参照に基づいていない場合、暗黙的に CHAR 値が NCHAR にキャストされ、NCHAR として比較されます。

例

条件 `Employees.GivenName = N'Susan'` では、CHAR カラム (Employees.GivenName) をリテラル N'Susan' と比較します。値 N'Susan' は CHAR にキャストされ、この比較は次のように記述されたものとして実行されます。

```
Employees.GivenName  = CAST( N'Susan' AS CHAR );
```

または、条件 `Employees.GivenName = T.nchar_column` では、値 T.nchar_column を CHAR にキャストできないことが検出されます。この比較は、次のように記述された場合と同等に実行され、Employees.GivenName のインデックスは使用できません。

```
CAST( Employees.GivenName AS NCHAR ) = T.nchar_column;
```

関連情報

NCHAR から CHAR への変換 [193 ページ]
NCHAR から CHAR への変換 [193 ページ]
損失を伴う変換および置換文字 [186 ページ]
CAST 関数 [データ型変換] [264 ページ]
CONVERT 関数 [データ型変換] [283 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.2.11.3 数値データ型間の比較

数値データ型を比較するときは規則が使用されます。規則は以下の表示順で検証され、一致する最初の規則が使用されます。

1. 一方の引数が TINYINT 型で、もう一方の引数が INTEGER 型の場合は、両方を INTEGER に変換してから比較を行います。
2. 一方の引数が TINYINT 型で、もう一方の引数が SMALLINT 型の場合は、両方を SMALLINT に変換してから比較を行います。
3. 一方の引数が UNSIGNED SMALLINT 型で、もう一方の引数が INTEGER 型の場合は、両方を INTEGER に変換してから比較を行います。
4. 引数のデータ型に共通のスーパータイプがある場合、その共通のスーパータイプデータ型に変換してから比較を行います。スーパータイプは、次の各リストの最終的なデータ型です。
   - BIT > TINYINT > UNSIGNED SMALLINT > UNSIGNED INTEGER > UNSIGNED BIGINT > NUMERIC
   - SMALLINT > INTEGER > BIGINT > NUMERIC
   - REAL > DOUBLE
   - CHAR > LONG VARCHAR
   - BINARY > LONG BINARY

   たとえば、2 つの引数が BIT 型と TINYINT 型である場合、これらの引数は NUMERIC 型に変換されます。

1.2.11.4 日付と時刻の比較

次の表に、特定のデータ型を日付、時刻、または日付／時刻のデータ型と比較したときの暗黙の変換をまとめます。

データ型	データ型	変換
CHAR	DATE	CHAR は TIMESTAMP にキャストされ、DATE は TIMESTAMP にキャストされます
CHAR	TIME	CHAR は TIME にキャストされます
CHAR	TIMESTAMP	CHAR は TIMESTAMP にキャストされます
CHAR	TIMESTAMP WITH TIME ZONE	CHAR は TIMESTAMP WITH TIME ZONE にキャストされます
DATE	TIME	無効
DATE	TIMESTAMP	DATE は TIMESTAMP にキャストされます
DATE	TIMESTAMP WITH TIME ZONE	DATE は TIMESTAMP WITH TIME ZONE にキャストされます
DATE	SMALLINT、INTEGER、BIGINT、NUMERIC	SMALLINT、INTEGER、BIGINT、NUMERIC の値は日付文字列として扱われ、TIMESTAMP にキャストされます。DATE は TIMESTAMP にキャストされます
DATE	REAL、FLOAT、DOUBLE	REAL、FLOAT、DOUBLE は 0000-02-29 から経過した日数として扱われ、TIMESTAMP にキャストされます。DATE は TIMESTAMP にキャストされます
TIME	TIMESTAMP	TIMESTAMP は TIME にキャストされます
TIME	TIMESTAMP WITH TIME ZONE	無効
TIMESTAMP	TIMESTAMP WITH TIME ZONE	TIMESTAMP は TIMESTAMP WITH TIME ZONE にキャストされます
TIMESTAMP	SMALLINT、INTEGER、BIGINT、NUMERIC	SMALLINT、INTEGER、BIGINT、NUMERIC の値は日付文字列として扱われ、TIMESTAMP にキャストされます
TIMESTAMP	REAL、FLOAT、DOUBLE	REAL、FLOAT、DOUBLE は 0000-02-29 から経過した日数として扱われ、TIMESTAMP にキャストされます

これ以降では、前述の表で示された情報をさらに詳しく説明しています。

1. データ型 TIME の値に比較できるのは、データ型 TIME、TIMESTAMP、CHAR の値だけです。それ以外のデータ型の値との比較は、変換エラーになります。時間値と別のデータ型の値を比較するときには、比較データ型は TIME になります。
2. TIMESTAMP、SMALLINT、INTEGER、BIGINT、NUMERIC、REAL、FLOAT、DOUBLE の値を DATE の値と比較するとき、比較データ型は常に TIMESTAMP になります。
3. TIMESTAMP WITH TIME ZONE 値を DATE 値と比較するとき、比較データ型は TIMESTAMP WITH TIME ZONE になります。
4. 時間値が TIMESTAMP にキャストされると、結果は、現在の日付と時間値を組み合わせたかたちになります。

5. データ型 SMALLINT、INTEGER、BIGINT、NUMERIC の正確な数値は、日付値に変換できます。変換は、数字を文字列として扱うことで実行されます。たとえば、整数値 20100401 は 2010 年 4 月 1 日を表します。
6. 符号なしの正確な数値データ型である BIT、TINYINT、UNSIGNED SMALLINT、UNSIGNED INTEGER、UNSIGNED BIGINT は、日付値には変換できません。
7. 概数値データ型である REAL、FLOAT、DOUBLE は、架空の日付 0000-02-29 から経過した日数として数値を扱うことで日付に変換できます。たとえば、307 は 0001-01-01 を表し、734169 は 2010-04-01 を表します。

関連情報

日付と時刻データ型 [153 ページ]

1.2.11.5 複合型の比較

配列要素は最初の要素から比較されます。

相違が見つかった場合、比較処理は停止し、その直前に比較した要素間の比較結果が返されます。すべての要素の比較が等しい場合、両配列は等しいものとなります。実行される比較は、配列に保持されていない式に対して実行される比較と同じです。ある配列が別の配列より短く、短い配列のすべての要素が長い配列の同じ要素と等しい場合、短い配列は長い配列より小さいと見なされます。

配列を比較する場合、それぞれの配列は和両立のデータ型を使用した値を保持している必要があります。重複排除と GROUP BY も、配列の式でサポートされています。たとえば、次の配列の比較では、クエリは 1 を返します。

```
SELECT IF ARRAY(3,4,5) > ARRAY(2,3,4) THEN 1 ELSE 0 ENDIF;
```

Row 型を比較して、ジョイン、重複排除、グループ化で使用することができます。ここで比較する 2 つのローの型は、上の例のローの式と同じだとします。

```
BEGIN
DECLARE test1 ROW(x INT, w ROW(y INT, z INT));
DECLARE test2 ROW(a INT, b ROW(c INT, d CHAR(3)));
SET test1 = ROW(3, ROW(6,7));
SET test2 = ROW(3, ROW(8,'7'));
SELECT (IF (test1 > test2) THEN 1 ELSE 0 ENDIF) AS RESULT FROM SYS.DUMMY;
END
```

2 つのローの式は、それぞれの構造が一致する場合にのみ比較できます。ただし、ローの式は同じ構造である必要がありますが、ローの型の属性名は同じである必要はなく、個々のリーフ値のデータ型も同じである必要はありません。和両立であることのみ必要です。

等号演算と不等号演算以外では、すべての ROW の比較は UNKNOWN となります。

1.2.11.6 Transact-SQL の文字列から日付/時刻への変換

時刻値のみ (日付なし) を含む文字を日付／時刻データ型に変換すると、データベースサーバでは現在の日付が使用されます。

時刻の部分が 3 桁未満の場合、ピリオドまたはコロンが後ろに付いているかどうかにかかわらず、データベースサーバでは同様に解釈されます。つまり、1 桁は 10 単位、2 桁は 100 単位、3 桁は 1000 単位です。

例

データベースサーバでは、セパレータの有無にかかわらず、同じ方法でミリ秒値が変換されます。

```
12:34:56.7 は 12:34:56.700 に
12:34:56:7 は 12:34:56.700 に
12.34.56.78 は 12:34:56.780 に
12.34.56:78 は 12:34:56.780 に
12:34:56.789 は 12:34:56.789 に
12:34:56:789 は 12:34:56.789 に
```

1.2.11.7 その他の比較

他のデータ型の比較も行われます。

1. CHAR (CHAR、VARCHAR、LONG VARCHAR など、ただし NCHAR 型以外) が混在しているデータ型の場合は、LONG VARCHAR に変換してから比較を行います。
2. いずれかの引数のデータ型が UNIQUEIDENTIFIER の場合は、UNIQUEIDENTIFIER に変換してから比較を行います。
3. いずれかの引数のデータ型がビット配列 (VARBIT または LONG VARBIT) の場合は、LONG VARBIT に変換してから比較を行います。
4. 一方の引数が CHARACTER データ型で、もう一方の引数が BINARY データ型の場合は、BINARY に変換してから比較を行います。
5. 一方の引数が CHAR 型で、もう一方の引数が NCHAR 型の場合は、事前に定義された推定規則を使用します。
6. 規則が存在しない場合は、NUMERIC 型に変換してから比較を行います。
   たとえば、2 つの引数が REAL 型と CHAR 型の場合、これらの引数はいずれも NUMERIC 型に変換されます。

格納される値の特性のため、ARRAY や TABLE REF などの一部のデータ型は、比較で使用されると、同じ型の値と比較される場合でも期待される結果を返さないことがあります。変数のみサポートされる TABLE REF データ型の場合、TABLE REF 型の 2 つの値は、同一の値が含まれていて所有者が同じ場合は等しいと見なされます。

関連情報

CHAR と NCHAR の比較 [187 ページ]

1.2.12 データ型変換

型の変換は自動的に発生することもあれば、CAST または CONVERT 関数を使って明示的に型変換が要求されることもあります。次の関数を使うことによっても、強制的に型変換を実行できます。

DATE 関数

式を日付に変換し、時間、分、秒を削除します。変換エラーがあればレポートされます。

DATETIME 関数

式を TIMESTAMP に変換し、タイムゾーンがあれば削除します。変換エラーがあればレポートされます。

STRING 関数

この関数は CAST(value AS LONG VARCHAR) と同じです。

VALUE+0.0

CAST (value AS DECIMAL) と同じです。

自動データ型変換の概要を次に示します。

- 数値式の中、あるいは引数として数値が期待される関数への引数として文字列を使う場合は、文字列は数字に変換されます。
- 文字列式の中、あるいは文字列関数の引数として数字を使う場合は、数字は文字列に変換されてから使用されます。
- すべての日付定数は文字列として指定されます。文字列は、自動的に日付に変換されてから使用されます。

TABLE REF データ型などの一部のデータ型 (必ず変数と同時に使用するデータ型) は、別のデータ型との間で変換を行うことはできません。

自動的なデータベース変換が適切ではない場合があります。たとえば、次の例では自動データ型変換が失敗します。

```
'12/31/90' + 5
'a' > 0
```

このセクションの内容:

関連情報

データ型変換関数 [203 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
STRING 関数 [文字列] [547 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.2.12.1 NCHAR から CHAR への変換

NCHAR から CHAR への変換は、CHAR データと NCHAR データの比較の一環として、または具体的に要求されたときに起こることがあります。

CHAR 文字セットによって CHAR 型に変換できない NCHAR 文字がある可能性があるため、このタイプの変換は損失を伴います。NCHAR 文字が CHAR に変換できない場合は、代わりに CHAR 文字セットの置換文字が使用されます。シングルバイト文字セットの場合、一般的に 16 進の 1A です。

on_charset_conversion_failure オプションの設定に応じて、文字が変換できないときに次のいずれかが起こります。

- 置換文字が使用され、警告が発行されない
- 置換文字が使用され、警告が発行される
- エラーが返される

そのため、NCHAR から CHAR への変換時はこのオプションを検討することが重要です。

関連情報

CHAR と NCHAR の比較 [187 ページ]

1.2.12.2 NUMERIC 型と文字列型への NULL 定数の変換

NULL 定数を NUMERIC または文字列型 (CHAR、VARCHAR、LONG VARCHAR、BINARY、VARBINARY、LONG BINARY) に変換すると、サイズは 0 に設定されます。次に例を示します。

`SELECT CAST( NULL AS CHAR )` は CHAR(0) を返します。

`SELECT CAST( NULL AS NUMERIC )` は NUMERIC(1,0) を返します。

1.2.12.3 ビット配列の変換

ビット配列との間で変換を行うことができます。

整数をビット配列に変換する

整数をビット配列に変換すると、ビット配列の長さは整数型のビット数になり、ビット配列の値はバイナリ表現になります。整数の最上位ビットは配列の最初のビットになります。

例

`SELECT CAST( CAST( 1 AS BIT ) AS VARBIT )` は 1 を含む VARBIT(1) を返します。

`SELECT CAST( CAST( 8 AS TINYINT ) AS VARBIT )` は 8 を含む VARBIT(00001000) を返します。

`SELECT CAST( CAST( 194 AS INTEGER ) AS VARBIT )` は 32 を含む VARBIT(00000000000000000000000011000010) を返します。

バイナリをビット配列に変換する

長さ `n` のバイナリ型をビット配列に変換すると、配列の長さは `n` * 8 ビットになります。ビット配列の最初の 8 ビットはバイナリ値の最初のバイトになります。バイナリ値の最上位ビットは配列の最初のビットになります。ビット配列の次の 8 ビットはバイナリ値の 2 番目のバイトになります。以降も同様です。

例

`SELECT CAST( 0x8181 AS VARBIT )` は 16 を含む VARBIT(1000000110000001) を返します。

文字をビット配列に変換する

長さ `n` の文字データ型をビット配列に変換すると、配列の長さは `n` ビットになります。各文字は '0' または '1' で、配列の対応するビットには値 0 または 1 が割り当てられます。

例

`SELECT CAST( '001100' AS VARBIT )` は 6 を含む VARBIT(001100) を返します。

ビット配列を整数に変換する

ビット配列を整数データ型に変換すると、ビット配列のバイナリ値は、最初に最上位ビットを使用し、整数型の格納形式に従って解釈されます。

例

`SELECT CAST( CAST( '11000010' AS VARBIT ) AS INTEGER )` は 194 (11000010_2 = 0xC2 = 194) を返します。

ビット配列をバイナリに変換する

ビット配列をバイナリに変換すると、配列の最初の 8 ビットがバイナリ値の最初のバイトになります。配列の最初のビットは、バイナリ値の最上位ビットになります。続く 8 ビットは 2 番目のバイトとして使用されます。以降も同様です。ビット配列の長さが 8 の倍数ではない場合、バイナリ値の最終バイトの最下位ビットには追加のゼロが入力されます。

例

`SELECT CAST( CAST( '1111' AS VARBIT ) AS BINARY )` は 0xF0 (1111_2 は 11110000_2 = 0xF0 になります) を返します。

`SELECT CAST( CAST( '0011000000110001' AS VARBIT ) AS BINARY )` は 0x3031 (0011000000110001_2 = 0x3031) を返します。

ビット配列を文字に変換する

長さ `n` ビットのビット配列を文字データ型に変換すると、結果の長さは `n` 文字になります。結果の各文字には、配列のビットに応じて '0' または '1' が指定されます。

例

`SELECT CAST( CAST( '01110' AS VARBIT ) AS VARCHAR )` は文字列 '01110' を返します。

1.2.12.4 数値セットの変換

DOUBLE 型を NUMERIC 型に変換すると、上位 15 桁の精度が維持されます。

関連情報

CAST 関数 [データ型変換] [264 ページ]
CONVERT 関数 [データ型変換] [283 ページ]

1.2.12.5 Java と SQL のデータ型変換

Java 型と SQL 型の間のデータ型変換は、Java ストアドプロシージャと JDBC アプリケーションの両方で必要です。Java から SQL、SQL から Java へのデータ型変換は、JDBC 標準に準拠して行われます。次の表で変換について説明します。

このセクションの内容:

1.2.12.5.1 Java から SQL のデータ型変換

Java 型と SQL 型の間のデータ型変換は、Java ストアドプロシージャと JDBC アプリケーションの両方で必要です。

Java 型	SQL 型
String	CHAR
String	VARCHAR
String	TEXT
java.math.BigDecimal	NUMERIC
java.math.BigDecimal	MONEY
java.math.BigDecimal	SMALLMONEY
boolean	BIT
byte	TINYINT
short	SMALLINT
int	INTEGER

Java 型	SQL 型
long	BIGINT
float	REAL
double	DOUBLE
byte[ ]	VARBINARY
byte[ ]	IMAGE
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	TIMESTAMP
java.lang.Double	DOUBLE
java.lang.Float	REAL
java.lang.Integer	INTEGER
java.lang.Long	BIGINT

1.2.12.5.2 SQL から Java のデータ型変換

SQL 型と Java 型の間のデータ型変換は、特定のマッピングに従います。

SQL 型	Java 型
CHAR	String
VARCHAR	String
TEXT	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
MONEY	java.math.BigDecimal
SMALLMONEY	java.math.BigDecimal
UNSIGNED BIGINT	java.math.BigDecimal (precision=20, scale=0)
BIT	boolean
TINYINT	byte
SMALLINT	short
UNSIGNED SMALLINT	int
INTEGER	int
UNSIGNED INTEGER	long
BIGINT	long
REAL	float

SQL 型	Java 型
FLOAT	double
DOUBLE	double
BINARY	byte[ ]
VARBINARY	byte[ ]
LONG BINARY	byte[ ]
IMAGE	byte[ ]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

1.3 SQL 関数

関数は、データベースの情報を返すために使用します。関数は、式を使用できるところではどこからでも呼び出すことができます。

マニュアルに特に指定がないかぎり、関数の引数のいずれかが NULL である場合は、NULL が返されます。

関数には、SQL 文と同じ構文の表記規則を使用します。

SQL Anywhere では、引数を省略できる場合、引数として DEFAULT を指定できます。

このセクションの内容:

関数のタイプ [199 ページ]
関数は、実行されるデータのタイプ、または使用されるコンテキストに従ってグループ化できます。

関数 [217 ページ]
関数を 1 つずつリストし、その右側に関数のタイプ (数値、文字など) を示します。

関連情報

SQL 構文の表記規則 [610 ページ]

1.3.1 関数のタイプ

関数は、実行されるデータのタイプ、または使用されるコンテキストに従ってグループ化できます。

> **注記**
>
> 特に明記しないかぎり、NULL をパラメータとして受け取る SQL Anywhere 関数は、NULL を返します。

Ultra Light: Ultra Light では、SQL Anywhere 用に記載されている関数のサブセットをサポートしていますが、一部に違いがあります。

このセクションの内容:

テキスト関数とイメージ関数は、text データ型と image データ型に対して作用します。TEXTPTR テキストおよびイメージ関数のみがサポートされます。

1.3.1.1 集合関数

集合関数は、データベースから選択されたローグループのデータを要約します。グループは、SELECT 文の GROUP BY 句を使用して形成されます。集合関数は、SELECT リストと、SELECT 文の HAVING 句と ORDER BY 句でのみ使用できます。

SQL Anywhere 関数のリスト

次の集合関数を使用できます。

- ARRAY_AGG 関数 [集合]
- AVG 関数 [集合]
- BIT_AND 関数 [集合]
- BIT_OR 関数 [集合]
- BIT_XOR 関数 [集合]
- COVAR_POP 関数 [集合]
- COVAR_SAMP 関数 [集合]
- COUNT 関数 [集合]
- COUNT_BIG 関数 [集合]
- CORR 関数 [集合]
- FIRST_VALUE 関数 [集合]
- GROUPING 関数 [集合]
- LAST_VALUE 関数 [集合]
- LIST 関数 [集合]
- MAX 関数 [集合]
- MEDIAN 関数 [集合]
- MIN 関数 [集合]
- REGR_AVGX 関数 [集合]
- REGR_AVGY 関数 [集合]
- REGR_COUNT 関数 [集合]
- REGR_INTERCEPT 関数 [集合]
- REGR_R2 関数 [集合]
- REGR_SLOPE 関数 [集合]
- REGR_SXX 関数 [集合]
- REGR_SXY 関数 [集合]
- REGR_SYY 関数 [集合]
- SET_BITS 関数 [集合]
- STDDEV 関数 [集合]

STDDEV_POP 関数 [集合]
STDDEV_SAMP 関数 [集合]
SUM 関数 [集合]
VAR_POP 関数 [集合]
VAR_SAMP 関数 [集合]
VARIANCE 関数 [集合]
XMLAGG 関数 [集合]

Ultra Light 関数のリスト

次の集合関数を使用できます。

AVG 関数 [集合] - Ultra Light
COUNT 関数 [集合] - Ultra Light
COUNT_UPLOAD_ROWS 関数 [集合] - Ultra Light
LIST 関数 [集合] - Ultra Light
MAX 関数 [集合] - Ultra Light
MIN 関数 [集合] - Ultra Light
SUM 関数 [集合] - Ultra Light

1.3.1.2 複合関数

複合関数を使用して配列のタスクを実行できます。

関数のリスト

次の複合関数を使用できます。

ARRAY コンストラクタ [複合]
ROW コンストラクタ [複合]
ARRAY_MAX_CARDINALITY 関数 [複合]
CARDINALITY 関数 [複合]
TRIM_ARRAY 関数 [複合]

関連情報

ARRAY_AGG 関数 [集合] [236 ページ]
UNNEST 配列演算子 [26 ページ]

1.3.1.3 ビット配列関数

ビット配列関数を使用すると、ビット配列でタスクを実行できます。

関数のリスト

次のビット配列関数を使用できます。

- BIT_AND 関数 [集合]
- BIT_OR 関数 [集合]
- BIT_XOR 関数 [集合]
- BIT_LENGTH 関数 [ビット配列]
- BIT_SUBSTR 関数 [ビット配列]
- COUNT_SET_BITS 関数 [ビット配列]
- GET_BIT 関数 [ビット配列]
- SET_BIT 関数 [ビット配列]
- SET_BITS 関数 [集合]

関連情報

ビット処理演算子 [31 ページ]
sa_get_bits システムプロシージャ [1510 ページ]
BIT_AND 関数 [集合] [249 ページ]
BIT_OR 関数 [集合] [251 ページ]
BIT_XOR 関数 [集合] [254 ページ]
BIT_LENGTH 関数 [ビット配列] [250 ページ]
BIT_SUBSTR 関数 [ビット配列] [252 ページ]
COUNT_SET_BITS 関数 [ビット配列] [293 ページ]
GET_BIT 関数 [ビット配列] [372 ページ]
SET_BIT 関数 [ビット配列] [522 ページ]
SET_BITS 関数 [集合] [524 ページ]

1.3.1.4 ランキング関数

ランキング関数により、クエリで指定した順番に基づいて、結果セットにある各ローのランク値を計算できます。

関数のリスト

次のランキング関数を使用できます。

- CUME_DIST 関数 [ランキング]
- DENSE_RANK 関数 [ランキング]
- PERCENT_RANK 関数 [ランキング]
- RANK 関数 [ランキング]

関連情報

CUME_DIST 関数 [ランキング] [299 ページ]
DENSE_RANK 関数 [ランキング] [332 ページ]
PERCENT_RANK 関数 [ランキング] [465 ページ]
RANK 関数 [ランキング] [479 ページ]

1.3.1.5 データ型変換関数

データ型変換関数を使用して、引数のデータ型を他のデータ型に変換したり、変換が可能かどうかをテストしたりします。

SQL Anywhere 関数のリスト

次のデータ型変換関数を使用できます。

- BINTOHEX 関数 [データ型変換]
- CAST 関数 [データ型変換]
- CONVERT 関数 [データ型変換]
- HEXTOBIN 関数 [データ型変換]
- HEXTOINT 関数 [データ型変換]
- INTTOHEX 関数 [データ型変換]
- ISDATE 関数 [データ型変換]
- ISNUMERIC 関数 [その他]
- TREAT 関数 [データ型変換]

Ultra Light 関数のリスト

次のデータ型変換関数を使用できます。

- CAST 関数 [データ型変換] - Ultra Light
- CONVERT 関数 [データ型変換] - Ultra Light
- HEXTOINT 関数 [データ型変換] - Ultra Light
- INTTOHEX 関数 [データ型変換] - Ultra Light
- ISDATE 関数 [データ型変換] - Ultra Light

関連情報

BINTOHEX 関数 [データ型変換] [248 ページ]
CAST 関数 [データ型変換] [264 ページ]
CONVERT 関数 [データ型変換] [283 ページ]
HEXTOBIN 関数 [データ型変換] [382 ページ]
HEXTOINT 関数 [データ型変換] [383 ページ]
INTTOHEX 関数 [データ型変換] [406 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
ISNUMERIC 関数 [その他] [411 ページ]
TREAT 関数 [データ型変換] [570 ページ]

1.3.1.6 日付および時刻関数

日付と時刻の関数は、DATE、TIME、TIMESTAMP、および TIMESTAMP WITH TIME ZONE の各データ型に対する操作を実行します。

SQL Anywhere には、Transact-SQL の日付型と時間型に対する互換性サポート (DATETIME と SMALLDATETIME を含む) があります。これらの Transact-SQL データ型は、ネイティブな TIMESTAMP データ型を介したドメインとして実装されます。

SQL Anywhere では、次の日付と時刻関数を使用できます。

- DATE 関数 [日付と時刻]
- DATEADD 関数 [日付と時刻]
- DATEDIFF 関数 [日付と時刻]
- DATEFORMAT 関数 [日付と時刻]
- DATENAME 関数 [日付と時刻]
- DATEPART 関数 [日付と時刻]
- DATETIME 関数 [日付と時刻]
- DAY 関数 [日付と時刻]
- DAYNAME 関数 [日付と時刻]

DAYS 関数 [日付と時刻]
DOW 関数 [日付と時刻]
GETDATE 関数 [日付と時刻]
HOUR 関数 [日付と時刻]
HOURS 関数 [日付と時刻]
MINUTE 関数 [日付と時刻]
MINUTES 関数 [日付と時刻]
MONTH 関数 [日付と時刻]
MONTHNAME 関数 [日付と時刻]
MONTHS 関数 [日付と時刻]
NOW 関数 [日付と時刻]
QUARTER 関数 [日付と時刻]
SECOND 関数 [日付と時刻]
SECONDS 関数 [日付と時刻]
SWITCHOFFSET 関数 [日付と時刻]
SYSDATETIMEOFFSET 関数 [日付と時刻]
TODAY 関数 [日付と時刻]
TODATETIMEOFFSET 関数 [日付と時刻]
WEEKS 関数 [日付と時刻]
YEAR 関数 [日付と時刻]
YEARS 関数 [日付と時刻]
YMD 関数 [日付と時刻]

Ultra Light では、次の日付と時刻関数を使用できます。

DATE 関数 [日付と時刻] - Ultra Light
DATEADD 関数 [日付と時刻] - Ultra Light
DATEDIFF 関数 [日付と時刻] - Ultra Light
DATEFORMAT 関数 [日付と時刻] - Ultra Light
DATENAME 関数 [日付と時刻] - Ultra Light
DATEPART 関数 [日付と時刻] - Ultra Light
DATETIME 関数 [日付と時刻] - Ultra Light
DAY 関数 [日付と時刻] - Ultra Light
DAYNAME 関数 [日付と時刻] - Ultra Light
DAYS 関数 [日付と時刻] - Ultra Light
DOW 関数 [日付と時刻] - Ultra Light
GETDATE 関数 [日付と時刻] - Ultra Light
HOUR 関数 [日付と時刻] - Ultra Light
HOURS 関数 [日付と時刻] - Ultra Light
MINUTE 関数 [日付と時刻] - Ultra Light
MINUTES 関数 [日付と時刻] - Ultra Light
MONTH 関数 [日付と時刻] - Ultra Light
MONTHNAME 関数 [日付と時刻] - Ultra Light
MONTHS 関数 [日付と時刻] - Ultra Light

NOW 関数 [日付と時刻] - Ultra Light
QUARTER 関数 [日付と時刻] - Ultra Light
SECOND 関数 [日付と時刻] - Ultra Light
SECONDS 関数 [日付と時刻] - Ultra Light
SWITCHOFFSET 関数 [日付と時刻] - Ultra Light
TODAY 関数 [日付と時刻] - Ultra Light
TODATETIMEOFFSET 関数 [日付と時刻] - Ultra Light
WEEKS 関数 [日付と時刻] - Ultra Light
YEAR 関数 [日付と時刻] - Ultra Light
YEARS 関数 [日付と時刻] - Ultra Light
YMD 関数 [日付と時刻] - Ultra Light

このセクションの内容:

日付の単位の指定 [206 ページ]
日付関数の多くは、日付の単位で構成される日付を使用します。次の表は、使用できる日付の単位の値を示します。

1.3.1.6.1 日付の単位の指定

日付関数の多くは、日付の単位で構成される日付を使用します。次の表は、使用できる日付の単位の値を示します。

日付と時刻の関数を使用する場合、日付または時刻の減算にマイナス記号を指定できます。たとえば、31 日前のタイムスタンプを取得するには、次の文を実行します。

```
SELECT DATEADD(day, -31, NOW());
```

日付の単位	省略形	値
Year	YY	1 ～ 9999
Quarter	QQ	1 ～ 4
Month	MM	1 ～ 12
Week	WK	1 ～ 54 日曜日を週の最初の日とします。54 週ある年は、土曜日から始まる閏年で発生します。
Day	DD	1 ～ 31
Dayofyear	DY	1 ～ 366
Weekday	DW	1 ～ 7 (日曜日 = 1、...、土曜日 = 7)
Hour	HH	0 ～ 23
Minute	MI	0 ～ 59
Second	SS	0 ～ 59
Millisecond	MS	0 ～ 999

日付の単位	省略形	値
Microsecond	MCS または US	0 ～ 999999
Calyearofweek	CYR	1 ～ 9999。その週が何年に開始したかを示します。週に年の最初の数日が含まれている場合は、その年の最初の曜日に応じて、週の最初の日が前年になる場合があります。年の最初の曜日が月曜日～木曜日の場合は、前年に属する日がその年に含まれることはありませんが、年の最初の曜日が金曜日～日曜日の場合、年の最初の週はその年の月曜日から開始します。
Calweekofyear	CWK	1 ～ 53。指定した日付がその年の第何週であるかを示します。 ISO の週のシステムと ISO 8601 の日付と時刻の標準の詳細については、ISO week date を参照してください。
Caldayofweek	CDW	1 ～ 7(月曜日 = 1、...、日曜日 = 7)
TZOffset	TZ	-840 ～ 840

関連情報

日付と時刻データ型 [153 ページ]
関数 [217 ページ]

1.3.1.7 ユーザ定義関数のサポート

ユーザ定義関数 (UDF) は、プログラムまたは環境のユーザによって作成される関数です。ユーザ定義関数には、プログラムまたは環境に組み込まれている関数との相違点があります。

SQL のユーザ定義関数

CREATE FUNCTION 文を使用して、独自の SQL 関数を実装できます。

CREATE FUNCTION 文中の RETURN 文によって、関数のデータ型が決まります。

一度 SQL ユーザ定義関数を作成すると、同じデータ型の組み込み関数が使用される任意の場所でその関数を使用できます。

Java および CLR でのユーザ定義関数

Java クラスを使用すると、必要に応じて関数をデータベースサーバからクライアントアプリケーションに移動できるという追加の利点があるため、より強力で柔軟性に富んだユーザ定義関数を実装できます。インストールされた Java クラスの "クラスメソッド" は、同じデータ型の組み込み関数が使用されるところでは必ずユーザ定義関数として使用できます。インスタンスメソッドは、クラスの特定のインスタンスと関連しているため、標準のユーザ定義関数とは動作が異なります。

CLR ストアドプロシージャおよび関数のサポートが含まれています。CLR ストアドプロシージャまたは関数の動作は、SQL ストアドプロシージャまたは関数と同じです。ただし、プロシージャまたは関数のコードは Microsoft C# または Visual Basic などの Microsoft .NET 言語で記述され、その実行はデータベースサーバの外側 (つまり別の Microsoft .NET 実行ファイル内) で行われます。Microsoft.NET バージョン 2.0 のみがサポートされています。

ユーザ定義関数とユーザ定義プロシージャのどちらを作成するかの判断

関数はプロシージャと似ています。関数とプロシージャのどちらを作成するかは、必要な戻り値と呼び出すオブジェクトに基づいて判断します。判断する際には、以下に示すユニークな特性について検討します。

関数:

- 任意の型の単一値を返すことができ、RETURNS 句を使用して戻り値の型を宣言できる
- 式の使用が可能なほとんどの場所で使用できる
- 定義できるのは IN パラメータのみ

プロシージャ:

- INOUT または OUT パラメータを使用して複数の値を返すことができる
- 結果セットを返すことができる
- クエリの FROM 句で参照したり、CALL 文や Transact-SQL の EXECUTE 文を使用して参照できる
- 名前付きパラメータを使用して呼び出すことができる

関連情報

CREATE FUNCTION 文 [837 ページ]
名前付きパラメータ [123 ページ]

1.3.1.8 その他の関数

その他の関数は、算術式、文字列式、日付/時刻式、他の関数の戻り値に対して操作を実行します。

SQL Anywhere 関数のリスト

次の各種関数を使用できます。

- ARGN 関数 [その他]
- COALESCE 関数 [その他]
- CONFLICT 関数 [その他]
- ERRORMSG 関数 [その他]
- ESTIMATE 関数 [その他]
- ESTIMATE_SOURCE 関数 [その他]
- EXPERIENCE_ESTIMATE 関数 [その他]
- EXPLANATION 関数 [その他]
- EXPRTYPE 関数 [その他]
- GET_IDENTITY 関数 [その他]
- GRAPHICAL_PLAN 関数 [その他]
- GREATER 関数 [その他]
- IDENTITY 関数 [その他]
- IFNULL 関数 [その他]
- INDEX_ESTIMATE 関数 [その他]
- ISNULL 関数 [その他]
- LESSER 関数 [その他]
- NEWID 関数 [その他]
- NULLIF 関数 [その他]
- NUMBER 関数 [その他]
- PLAN 関数 [その他]
- REWRITE 関数 [その他]
- ROW_NUMBER 関数 [その他]
- SQLDIALECT 関数 [その他]
- SQLFLAGGER 関数 [その他]
- ERROR_LINE 関数 [その他]
- TRACEBACK 関数 [その他]
- TRANSACTSQL 関数 [その他]
- VAREXISTS 関数 [その他]
- WATCOMSQL 関数 [その他]

Ultra Light 関数のリスト

次の各種関数を使用できます。

ARGN 関数 [その他]
COALESCE 関数 [その他]
EXPLANATION 関数 [その他]
GREATER 関数 [その他]
IFNULL 関数 [その他]
ISNULL 関数 [その他]
LESSER 関数 [その他]
NEWID 関数 [その他]
NULLIF 関数 [その他]

1.3.1.9 数値関数

数値関数は、数値データ型の算術演算を実行したり、数値情報を返したりします。

SQL Anywhere 関数のリスト

次の数値関数を使用できます。

ABS 関数 [数値]
ACOS 関数 [数値]
ASIN 関数 [数値]
ATAN 関数 [数値]
ATAN2 関数 [数値]
CEILING 関数 [数値]
COS 関数 [数値]
COT 関数 [数値]
DEGREES 関数 [数値]
EXP 関数 [数値]
FLOOR 関数 [数値]
LOG 関数 [数値]
LOG10 関数 [数値]
MOD 関数 [数値]
PI 関数 [数値]
POWER 関数 [数値]
RADIANS 関数 [数値]
RAND 関数 [数値]

REMAINDER 関数 [数値]
ROUND 関数 [数値]
SIGN 関数 [数値]
SIN 関数 [数値]
SQRT 関数 [数値]
TAN 関数 [数値]
TRUNCNUM 関数 [数値]

Ultra Light 関数のリスト

次の数値関数を使用できます。

ABS 関数 [数値]
ACOS 関数 [数値]
ASIN 関数 [数値]
ATAN 関数 [数値]
ATAN2 関数 [数値]
CEILING 関数 [数値]
COS 関数 [数値]
COT 関数 [数値]
DEGREES 関数 [数値]
EXP 関数 [数値]
FLOOR 関数 [数値]
LOG 関数 [数値]
LOG10 関数 [数値]
MOD 関数 [数値]
PI 関数 [数値]
POWER 関数 [数値]
RADIANS 関数 [数値]
REMAINDER 関数 [数値]
ROUND 関数 [数値]
SIGN 関数 [数値]
SIN 関数 [数値]
SQRT 関数 [数値]
TAN 関数 [数値]
TRUNCNUM 関数 [数値]

1.3.1.10 Web サービス関数

HTTP 関数は、Web サービス内の HTTP 要求の処理を支援します。同様に、SOAP 関数は、Web サービス内の SOAP 要求の処理を支援します。

次の関数を使用できます。

- HTML_DECODE 関数 [その他]
- HTML_ENCODE 関数 [その他]
- HTTP_BODY 関数 [Web サービス]
- HTTP_DECODE 関数 [Web サービス]
- HTTP_ENCODE 関数 [Web サービス]
- HTTP_HEADER 関数 [Web サービス]
- HTTP_RESPONSE_HEADER 関数 [Web サービス]
- HTTP_VARIABLE 関数 [Web サービス]
- NEXT_HTTP_HEADER 関数 [Web サービス]
- NEXT_HTTP_RESPONSE_HEADER 関数 [Web サービス]
- NEXT_HTTP_VARIABLE 関数 [Web サービス]
- NEXT_SOAP_HEADER 関数 [SOAP]
- SOAP_HEADER 関数 [SOAP]

また、Web サービスで使用できるシステムプロシージャもあります。

関連情報

Web サービスシステムプロシージャ [1431 ページ]
HTML_DECODE 関数 [その他] [388 ページ]
HTML_ENCODE 関数 [その他] [389 ページ]
HTTP_BODY 関数 [Web サービス] [391 ページ]
HTTP_DECODE 関数 [Web サービス] [392 ページ]
HTTP_ENCODE 関数 [Web サービス] [393 ページ]
HTTP_HEADER 関数 [Web サービス] [395 ページ]
HTTP_RESPONSE_HEADER 関数 [Web サービス] [397 ページ]
HTTP_VARIABLE 関数 [Web サービス] [399 ページ]
NEXT_HTTP_HEADER 関数 [Web サービス] [452 ページ]
NEXT_HTTP_RESPONSE_HEADER 関数 [Web サービス] [454 ページ]
NEXT_HTTP_VARIABLE 関数 [Web サービス] [455 ページ]
NEXT_SOAP_HEADER 関数 [SOAP] [457 ページ]
SOAP_HEADER 関数 [SOAP] [529 ページ]

1.3.1.11 文字列関数

文字列関数は、文字列に対して変換、抽出、操作の演算を実行したり、文字列に関する情報を返したりします。

マルチバイト文字セットを操作する場合は、使用する関数が文字とバイトのどちらの情報を返すかを十分に確認してください。

SQL Anywhere 関数のリスト

次の文字列関数を使用できます。

- ASCII 関数 [文字列]
- BASE64_DECODE 関数 [文字列]
- BASE64_ENCODE 関数 [文字列]
- BYTE_LENGTH 関数 [文字列]
- BYTE_SUBSTR 関数 [文字列]
- CHAR 関数 [文字列]
- CHARINDEX 関数 [文字列]
- CHAR_LENGTH 関数 [文字列]
- COMPARE 関数 [文字列]
- COMPRESS 関数 [文字列]
- CSCONVERT 関数 [文字列]
- DECOMPRESS 関数 [文字列]
- DECRYPT 関数 [文字列]
- DIFFERENCE 関数 [文字列]
- ENCRYPT 関数 [文字列]
- HASH 関数 [文字列]
- INSERTSTR 関数 [文字列]
- LCASE 関数 [文字列]
- LEFT 関数 [文字列]
- LENGTH 関数 [文字列]
- LOCATE 関数 [文字列]
- LOWER 関数 [文字列]
- LTRIM 関数 [文字列]
- NCHAR 関数 [文字列]
- PATINDEX 関数 [文字列]
- READ_CLIENT_FILE 関数 [文字列]
- READ_SERVER_FILE 関数 [文字列]
- REGEXP_SUBSTR 関数 [文字列]
- REPEAT 関数 [文字列]
- REPLACE 関数 [文字列]
- REPLICATE 関数 [文字列]
- REVERSE 関数 [文字列]

RIGHT 関数 [文字列]
RTRIM 関数 [文字列]
SIMILAR 関数 [文字列]
SORTKEY 関数 [文字列]
SOUNDEX 関数 [文字列]
SPACE 関数 [文字列]
STR 関数 [文字列]
STRING 関数 [文字列]
STRTOUUID 関数 [文字列]
STUFF 関数 [文字列]
SUBSTRING 関数 [文字列]
TO_CHAR 関数 [文字列]
TO_NCHAR 関数 [文字列]
TRIM 関数 [文字列]
UCASE 関数 [文字列]
UNICODE 関数 [文字列]
UNISTR 関数 [文字列]
UPPER 関数 [文字列]
UUIDTOSTR 関数 [文字列]
XMLCONCAT 関数 [文字列]
XMLELEMENT 関数 [文字列]
XMLFOREST 関数 [文字列]
XMLGEN 関数 [文字列]

Ultra Light 関数のリスト

次の文字列関数を使用できます。

ASCII 関数 [文字列]
BYTE_LENGTH 関数 [文字列]
BYTE_SUBSTR 関数 [文字列]
CHAR 関数 [文字列]
CHARINDEX 関数 [文字列]
CHAR_LENGTH 関数 [文字列]
DIFFERENCE 関数 [文字列]
INSERTSTR 関数 [文字列]
LCASE 関数 [文字列]
LEFT 関数 [文字列]
LENGTH 関数 [文字列]
LOCATE 関数 [文字列]
LOWER 関数 [文字列]

LTRIM 関数 [文字列]
PATINDEX 関数 [文字列]
REPEAT 関数 [文字列]
REPLACE 関数 [文字列]
REPLICATE 関数 [文字列]
RIGHT 関数 [文字列]
RTRIM 関数 [文字列]
SIMILAR 関数 [文字列]
SOUNDEX 関数 [文字列]
SPACE 関数 [文字列]
STR 関数 [文字列]
STRING 関数 [文字列]
STRTOUUID 関数 [文字列]
STUFF 関数 [文字列]
SUBSTRING 関数 [文字列]
TRIM 関数 [文字列]
UCASE 関数 [文字列]
UPPER 関数 [文字列]
UUIDTOSTR 関数 [文字列]

1.3.1.12 システム関数

システム関数は、システム情報を返します。

関数のリスト

次のシステム関数を使用できます。

CONNECTION_EXTENDED_PROPERTY 関数 [文字列]
CONNECTION_PROPERTY 関数 [システム]
DATALENGTH 関数 [システム]
DB_ID 関数 [システム]
DB_NAME 関数 [システム]
DB_EXTENDED_PROPERTY 関数 [システム]
DB_PROPERTY 関数 [システム]
EVENT_CONDITION 関数 [システム]
EVENT_CONDITION_NAME 関数 [システム]
EVENT_PARAMETER 関数 [システム]
NEXT_CONNECTION 関数 [システム]
NEXT_DATABASE 関数 [システム]

PROPERTY 関数 [システム]
PROPERTY_DESCRIPTION 関数 [システム]
PROPERTY_NAME 関数 [システム]
PROPERTY_NUMBER 関数 [システム]
SUSER_ID 関数 [システム]
SUSER_NAME 関数 [システム]
TSEQUAL 関数 [システム] (旧式)
USER_ID 関数 [システム]
USER_NAME 関数 [システム]
DB_PROPERTY 関数 [システム]

Ultra Light 関数

DB_PROPERTY 関数 [システム]
ML_GET_SERVER_NOTIFICATION 関数 [システム]
SYNC_PROFILE_OPTION_VALUE 関数 [システム]

SQL Anywhere の注意

- db_id、db_name、datalength は、組み込み関数として実装されます。
- 一部のシステム関数は、ストアドプロシージャとして実装されます。

他の場所で説明されていないシステム関数は次の表に注記してあります。これらの関数は、ストアドプロシージャとして実装されます。

構文: COL_LENGTH

```
COL_LENGTH( @object_name, @column_name )
```

指定されたカラムについて INTEGER によって定義された長さを返します。`@object_name` には、所有者を含めることができます ('GROUPO.Customers' など)。

構文: COL_TERM

```
COL_NAME( @object_id, @column_id [, @database_id ] )
```

CHAR(128) カラムの名前を返します。

構文: INDEX_COL

```
INDEX_COL ( @table_name, @index_id, @key_# [, @user_id ] )
```

インデックスカラムの CHAR(128) の名前を返します。`@table_name` には、所有者を含めることができます ('GROUPO.Customers' など)。

構文: OBJECT_ID

```
OBJECT_ID( @object_name )
```

INTEGER オブジェクト ID を返します。`@object_name` には、所有者を含めることができます ('GROUPO.Customers' など)。

構文: OBJECT_NAME

```
OBJECT_NAME ( @object_id [, @database_id ] )
```

CHAR(128) オブジェクトの名前を返します。

1.3.1.13 テキスト関数とイメージ関数

テキスト関数とイメージ関数は、text データ型と image データ型に対して作用します。TEXTPTR テキストおよびイメージ関数のみがサポートされます。

1.3.2 関数

関数を 1 つずつリストし、その右側に関数のタイプ (数値、文字など) を示します。

このセクションの内容:

CHAR_LENGTH 関数 [文字列] [269 ページ]
文字列の文字数を返します。

CHARINDEX 関数 [文字列] [270 ページ]
ある文字列内で 1 つの文字列の位置を返します。

COALESCE 関数 [その他] [272 ページ]
リストの中から NULL でない最初の式を返します。この関数は ISNULL 関数と同じです。

COMPARE 関数 [文字列] [273 ページ]
代替照合規則に基づいて 2 つの文字列を比較できます。

COMPRESS 関数 [文字列] [275 ページ]
文字列を圧縮し、LONG BINARY 型の値を返します。

CONFLICT 関数 [その他] [277 ページ]
カラムが、SQL Remote 環境で統合データベースに対して実行される UPDATE の競合の原因であるかどうかを示します。

CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]
指定したプロパティの値を返します。オプションでプロパティ固有の文字列パラメータを指定できます。

CONNECTION_PROPERTY 関数 [システム] [281 ページ]
特定の接続プロパティの値を文字列で返します。

CONVERT 関数 [データ型変換] [283 ページ]
指定されたデータ型に変換した式を返します。

CORR 関数 [集合] [286 ページ]
数値のペアのセットの相関係数を返します。

COS 関数 [数値] [287 ページ]
引数で与えられた角度のコサインをラジアン単位で返します。

COT 関数 [数値] [288 ページ]
引数で与えられた角度のコタンジェントをラジアン単位で返します。

COUNT 関数 [集合] [289 ページ]
指定されたパラメータに従って、グループのロー数をカウントします。

COUNT_BIG 関数 [集合] [291 ページ]
指定されたパラメータに従って、グループのロー数をカウントします。

COUNT_SET_BITS 関数 [ビット配列] [293 ページ]
配列でビットが 1 (TRUE) に設定された数値のカウントを返します。

COVAR_POP 関数 [集合] [294 ページ]
数値のペアのセットの母共分散を返します。

COVAR_SAMP 関数 [集合] [295 ページ]
数値のペアのセットの標本共分散を返します。

CSCONVERT 関数 [文字列] [297 ページ]
文字列の文字セットを変換します。

CUME_DIST 関数 [ランキング] [299 ページ]
ローのグループ内で 1 つの値の相対位置を計算します。

DATALENGTH 関数 [システム] [301 ページ]

式の結果に必要な基本となる記憶領域の長さ (バイト) を返します。

DATE 関数 [日付と時刻] [303 ページ]
式を日付に変換し、時間、分、秒を削除します。

DATEADD 関数 [日付と時刻] [304 ページ]
日付の単位をその引数に追加することによって求められる TIMESTAMP または TIMESTAMP WITH TIME ZONE の値を返します。

DATEDIFF 関数 [日付と時刻] [305 ページ]
2 つの日付間の期間を返します。

DATEFORMAT 関数 [日付と時刻] [307 ページ]
日付式を表す文字列を、指定した形式で返します。

DATENAME 関数 [日付と時刻] [308 ページ]
TIMESTAMP または TIMESTAMP WITH TIME ZONE の値の特定部分の名前 (月の名前 "June" など) を文字列で返します。

DATEPART 関数 [日付と時刻] [310 ページ]
TIMESTAMP または TIMESTAMP WITH TIME ZONE の値の一部を返します。

DATETIME 関数 [日付と時刻] [311 ページ]
式を TIMESTAMP 値に変換します。

DAY 関数 [日付と時刻] [313 ページ]
引数の日付を 1 ～ 31 の間の整数で返します。

DAYNAME 関数 [日付と時刻] [314 ページ]
日付から曜日の名前を返します。

DAYS 関数 [日付と時刻] [315 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の日数を返します。

DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]
指定したプロパティの値を返します。オプションでプロパティ固有の文字列パラメータを指定できます。

DB_ID 関数 [システム] [321 ページ]
データベース ID 番号を返します。

DB_NAME 関数 [システム] [322 ページ]
指定した ID 番号を持つデータベースの名前を返します。

DB_PROPERTY 関数 [システム] [324 ページ]
指定されたデータベースのプロパティの値を返します。

DECOMPRESS 関数 [文字列] [325 ページ]
文字列を解凍し、LONG BINARY 値を返します。

DECRYPT 関数 [文字列] [327 ページ]
指定されたキーを使用して文字列を復号化し、LONG BINARY 値を返します。

DEGREES 関数 [数値] [331 ページ]
数値をラジアンから度数に変換します。

DENSE_RANK 関数 [ランキング] [332 ページ]
パーティション内の値のランクを計算します。同位の値の場合、DENSE_RANK はランキングシーケンス内にギャップを残しません。

DIFFERENCE 関数 [文字列] [334 ページ]
2 つの文字列式の SOUNDEX 値の差を返します。

DOW 関数 [日付と時刻] [335 ページ]
指定した日付の曜日を表す 1 ～ 7 の数を返します (日曜日 = 1、月曜日 = 2、以下同様)。

ENCRYPT 関数 [文字列] [336 ページ]
指定された暗号化キーを使用して指定された値を暗号化し、LONG BINARY 値を返します。

ERROR_LINE 関数 [その他] [340 ページ]
TRY...CATCH 文の CATCH ブロックを呼び出したエラーが発生したプロシージャまたはバッチの行番号を返します。

ERROR_MESSAGE 関数 [その他] [342 ページ]
TRY...CATCH 文の CATCH ブロックを呼び出したエラーのメッセージテキストを返します。

ERROR_PROCEDURE 関数 [その他] [343 ページ]
例外ハンドラが実行される契機となったエラー内のプロシージャ名を返します。

ERROR_SQLCODE 関数 [その他] [344 ページ]
エラーハンドラを呼び出したエラーの SQLCODE を返します。

ERROR_SQLSTATE 関数 [その他] [346 ページ]
エラーハンドラを呼び出したエラーの SQLSTATE を返します。

ERROR_STACK_TRACE 関数 [その他] [347 ページ]
エラーハンドラを呼び出したエラーの呼び出しシーケンススタックトレースを返します。

ERRORMSG 関数 [その他] [349 ページ]
現在のエラー、または指定した SQLSTATE 値または SQLCODE 値のエラーメッセージを返します。

ESTIMATE 関数 [その他] [350 ページ]
指定したパラメータに基づいてクエリオプティマイザによって計算されたパーセンテージとして、選択性推定を返します。

ESTIMATE_SOURCE 関数 [その他] [352 ページ]
クエリオプティマイザで使用される選択性推定のソースを提供します。

EVENT_CONDITION 関数 [システム] [353 ページ]
イベントハンドラがトリガされる条件を指定します。

EVENT_CONDITION_NAME 関数 [システム] [355 ページ]
EVENT_CONDITION に指定可能なパラメータをリストします。

EVENT_PARAMETER 関数 [システム] [356 ページ]
イベントハンドラのためのコンテキスト情報を提供します。

EXP 関数 [数値] [359 ページ]
自然対数の底 e を、与えられた引数でべき乗した結果を返します。

EXPERIENCE_ESTIMATE 関数 [その他] [360 ページ]
指定したパラメータに基づいてクエリオプティマイザによって計算されたパーセンテージとして、選択性推定を返します。

EXPLANATION 関数 [その他] [362 ページ]
SQL 文の最適化方法をプレーンテキスト文字列で返します。

EXPRTYPE 関数 [その他] [363 ページ]
式のデータ型を識別する文字列を返します。

EXTENDED_PROPERTY function [システム] [365 ページ]
指定されたデータベースのプロパティの値を返します。オプションでプロパティ固有の文字列パラメータを指定できます。

EXTRACT 関数 [日付と時刻] [367 ページ]
TIMESTAMP 式から日付要素を返します。

FIRST_VALUE 関数 [集合] [369 ページ]
ウィンドウの最初のローの値を返します。

FLOOR 関数 [数値] [371 ページ]
指定された値以下の、最大の整数値を返します。

GET_BIT 関数 [ビット配列] [372 ページ]
ビット配列の指定したビットの値 (1 または 0) を返します。

GET_IDENTITY 関数 [その他] [373 ページ]
AUTOINCREMENT カラムに値を割り当てます。AUTOINCREMENT を使用して数を生成する代わりに、この関数を使用できます。

GETDATE 関数 [日付と時刻] [375 ページ]
現在の年、月、日、時、分、秒、秒以下を返します。

GRAPHICAL_PLAN 関数 [その他] [376 ページ]
SQL 文のプランの最適化方法を、XML フォーマットの文字列で返します。

GREATER 関数 [その他] [378 ページ]
2 つのパラメータ値のうち、より大きい値を返します。

GROUPING 関数 [集合] [379 ページ]
GROUP BY 演算の結果セット内のカラムが NULL である場合、その理由が小計ローの一部であるためか、または基本データによるためかを識別します。

HASH 関数 [文字列] [380 ページ]
指定された値をハッシュ形式で返します。

HEXTOBIN 関数 [データ型変換] [382 ページ]
16 進文字列と同等の LONG BINARY を返します。

HEXTOINT 関数 [データ型変換] [383 ページ]
16 進文字列と同等の 10 進整数を返します。

HOUR 関数 [日付と時刻] [385 ページ]
TIMESTAMP 値の時間部分を返します。

HOURS 関数 [日付と時刻] [386 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の時間数を返します。

HTML_DECODE 関数 [その他] [388 ページ]
HTML リテラル文字列で表示される特殊文字エンティティを復号化します。

HTML_ENCODE 関数 [その他] [389 ページ]
HTML ドキュメントに挿入する文字列内の特殊文字をエンコードします。

HTTP_BODY 関数 [Web サービス] [391 ページ]
HTTP 要求の本文をバイナリ形式で返します。たとえば、POST 要求では、これは未加工の POST データになります。

グループ内のローごとに値のデリミタ付きリストを返します。

LOCATE 関数 [文字列] [423 ページ]
異なる文字列内のある文字列の位置を返します。

LOG 関数 [数値] [425 ページ]
数の自然対数を返します。

LOG10 関数 [数値] [426 ページ]
数値の対数 (基数 10) を返します。

LOWER 関数 [文字列] [427 ページ]
文字列中のすべての文字を小文字に変換します。

LTRIM 関数 [文字列] [428 ページ]
文字列の先行ブランクを削除します。

MAX 関数 [集合] [430 ページ]
各ローグループで見つかった式の最大値を返します。

MEDIAN 関数 [集合] [431 ページ]
ローのセットの数値式の中央値を計算します。

MICROSECOND 関数 [日付と時刻] [433 ページ]
TIMESTAMP 式のマイクロ秒の部分を返します。

MILLISECOND 関数 [日付と時刻] [435 ページ]
TIMESTAMP 式のミリ秒の部分を返します。

MIN 関数 [集合] [436 ページ]
各ローグループで見つかった式の最小値を返します。

MINUTE 関数 [日付と時刻] [438 ページ]
TIMESTAMP 値の分部分を返します。

MINUTES 関数 [日付と時刻] [439 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の分境界の数を返します。

MOD 関数 [数値] [441 ページ]
整数を整数で割ったときの余りを返します。

MONTH 関数 [日付と時刻] [442 ページ]
指定した日付の月を返します。

MONTHNAME 関数 [日付と時刻] [443 ページ]
日付から月の名前を返します。

MONTHS 関数 [日付と時刻] [444 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の月境界の数を返します。

NCHAR 関数 [文字列] [446 ページ]
Unicode のコードポイントがパラメータに指定された 1 文字を含む NCHAR 文字列を返します。値が有効なコードポイント値ではない場合は、NULL を返します。

NEWID 関数 [その他] [447 ページ]
UUID (ユニバーサルユニーク識別子) 値を生成します。UUID は、GUID (グローバルユニーク識別子) と同じです。

NEXT_CONNECTION 関数 [システム] [449 ページ]
次の接続の識別番号を返します。

NEXT_DATABASE 関数 [システム] [451 ページ]
データベースの識別番号を返します。

NEXT_HTTP_HEADER 関数 [Web サービス] [452 ページ]
次の HTTP ヘッダの名前を返します。

NEXT_HTTP_RESPONSE_HEADER 関数 [Web サービス] [454 ページ]
次の HTTP 応答ヘッダの名前を取得します。

NEXT_HTTP_VARIABLE 関数 [Web サービス] [455 ページ]
次の HTTP 変数の名前を返します。

NEXT_SOAP_HEADER 関数 [SOAP] [457 ページ]
SOAP 要求ヘッダの次のヘッダキーを返します。

NOW 関数 [日付と時刻] [458 ページ]
現在の日付と時刻を TIMESTAMP 値で返します。精度はシステムクロックの精度によって制限されます。

NULLIF 関数 [その他] [459 ページ]
式を比較して、CASE 式の省略形を提供します。

NUMBER 関数 [その他] [461 ページ]
クエリの結果の連続した各ローに対して、1 から開始する番号を生成します。NUMBER 関数は主に、SELECT リストで使用するために提供されています。

PATINDEX 関数 [文字列] [462 ページ]
文字列のパターンが最初に出現した開始位置を表す整数を返します。

PERCENT_RANK 関数 [ランキング] [465 ページ]
ロー X が関数の引数と ORDER BY 指定で定義される場合、PERCENT_RANK 関数は、グループのロー数で割ったロー X - 1 のランクを計算します。

PI 関数 [数値] [466 ページ]
PI の数値を返します。

PLAN 関数 [その他] [467 ページ]
SQL 文の長いプランの最適化方法を文字列で返します。

POWER 関数 [数値] [469 ページ]
数のべき乗を表す数を計算します。

PROPERTY 関数 [システム] [470 ページ]
指定したデータベースサーバのプロパティの値を文字列で返します。

PROPERTY_DESCRIPTION 関数 [システム] [471 ページ]
プロパティの説明を返します。

PROPERTY_IS_TRACKABLE 関数 [システム] [472 ページ]
追跡された値を格納することによって指定したデータベースサーバプロパティの履歴データを更新できるかどうかを返します。

PROPERTY_NAME 関数 [システム] [473 ページ]
指定した接続レベルで、指定したプロパティ ID を持つプロパティの名前を返します。

PROPERTY_NUMBER 関数 [システム] [474 ページ]
指定したプロパティ名を持つプロパティのプロパティ番号を返します。

QUARTER 関数 [日付と時刻] [475 ページ]

指定した TIMESTAMP 式から、四半期を示す数を返します。

RADIANS 関数 [数値] [476 ページ]
度数をラジアンに変換します。

RAND 関数 [数値] [477 ページ]
オプションのシードから、間隔 0 ～ 1 の乱数を返します。

RANK 関数 [ランキング] [479 ページ]
値のグループ内でのランクの値を計算します。同位の場合、RANK 関数はランキングシーケンス内にギャップを残します。

READ_CLIENT_FILE 関数 [文字列] [481 ページ]
クライアントコンピュータ上で指定したファイルからデータを読み込みます。

READ_SERVER_FILE 関数 [文字列] [482 ページ]
サーバ上の指定したファイルからデータを読み込み、ファイルの完全な内容または一部の内容を LONG BINARY 値として返します。

REGEXP_SUBSTR 関数 [文字列] [484 ページ]
正規表現を使用して文字列から部分文字列を抽出します。

REGR_AVGX 関数 [集合] [486 ページ]
回帰直線の独立変数の平均を計算します。

REGR_AVGY 関数 [集合] [487 ページ]
回帰直線の従属変数の平均を計算します。

REGR_COUNT 関数 [集合] [489 ページ]
回帰直線の調整に使用される NULL 以外の数値のペアの数を表す整数を返します。

REGR_INTERCEPT 関数 [集合] [490 ページ]
従属変数と独立変数に最適な線形回帰直線の y 切片を計算します。

REGR_R2 関数 [集合] [492 ページ]
回帰直線の決定係数 (*R-squared* または適合度の統計情報とも呼びます) を計算します。

REGR_SLOPE 関数 [集合] [493 ページ]
NULL 以外のペアに調整された線形回帰直線の傾きを計算します。

REGR_SXX 関数 [集合] [495 ページ]
線形回帰モデルに使用される独立した式の平方値の合計を返します。REGR_SXX 関数は、回帰モデルの統計的な有効性を評価するときに使用できます。

REGR_SXY 関数 [集合] [496 ページ]
従属変数と独立変数の積和を返します。REGR_SXY 関数は、回帰モデルの統計的な有効性を評価するときに使用できます。

REGR_SYY 関数 [集合] [498 ページ]
回帰モデルの統計的有効性を評価できる値を返します。

REMAINDER 関数 [数値] [499 ページ]
整数を整数で割ったときの余りを返します。

REPEAT 関数 [文字列] [501 ページ]
文字列を指定された回数だけ連結します。

REPLACE 関数 [文字列] [502 ページ]

文字列を別の文字列で置換し、新しい結果を返します。

REPLICATE 関数 [文字列] [504 ページ]
文字列を指定された回数だけ連結します。

REVERSE 関数 [文字列] [505 ページ]
文字式の逆の値を返します。

REWRITE 関数 [その他] [506 ページ]
書き換えられた SELECT 文、UPDATE 文または DELETE 文を返します。

RIGHT 関数 [文字列] [508 ページ]
文字列の一番右側にある文字を返します。

ROUND 関数 [数値] [509 ページ]
`numeric-expression` を `integer-expression` で指定した小数点以下の桁数に丸めます。

ROW コンストラクタ [複合] [510 ページ]
ペア (`field namedata type`, ...) (名前は **fields**) のシーケンスを返します。

ROW_NUMBER 関数 [その他] [512 ページ]
各ローにユニークな番号を割り当てます。この関数は NUMBER 関数の代わりに使用できます。

ROWID 関数 [その他] [514 ページ]
テーブル内のローを一意に識別する UNSIGNED BIGINT ビット値を返します。

RTRIM 関数 [文字列] [515 ページ]
文字列の後続ブランクを削除します。

SECOND 関数 [日付と時刻] [517 ページ]
TIMESTAMP 引数の秒の値を返します。

SECONDS 関数 [日付と時刻] [518 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の秒境界の数を返します。

SECURE_SIGN_MESSAGE 関数 [文字列] [520 ページ]
メッセージをデジタル署名します。

SECURE_VERIFY_MESSAGE 関数 [文字列] [521 ページ]
メッセージをデジタル検証します。

SET_BIT 関数 [ビット配列] [522 ページ]
ビット配列の特定ビットの値を設定します。

SET_BITS 関数 [集合] [524 ページ]
ローのセットに含まれる値に対応する特定ビットを 1 (TRUE) に設定するときに、ビット配列を作成します。

SIGN 関数 [数値] [525 ページ]
指定された数値の符号 (正または負) を返します。

SIMILAR 関数 [文字列] [526 ページ]
2 つの文字列の類似性を示す数を返します。

SIN 関数 [数値] [527 ページ]
数のサインを返します。

SOAP_HEADER 関数 [SOAP] [529 ページ]
SOAP ヘッダエントリまたは SOAP 要求のヘッダエントリの属性値を返します。

SORTKEY 関数 [文字列] [531 ページ]
ソートキー値を生成します。つまり、代替照合規則に基づいて文字列をソートする場合に使用できる値を生成します。

SOUNDEX 関数 [文字列] [533 ページ]
文字列の発音を表す数を返します。

SPACE 関数 [文字列] [535 ページ]
指定した数のスペースを返します。

SQLDIALECT 関数 [その他] [536 ページ]
文の SQL ダイアレクトを示す Watcom SQL または Transact-SQL のどちらかを返します。

SQLFLAGGER 関数 [その他] [537 ページ]
ANSI/ISO SQL 標準など、指定した規格に対する、指定した SQL 文の準拠性を返します。

SQRT 関数 [数値] [538 ページ]
数の平方根を返します。

STACK_TRACE 関数 [その他] [539 ページ]
現在の文のスタックトレースに関する情報を返します。

STDDEV 関数 [集合] [542 ページ]
STDDEV_SAMP のエイリアスです。

STDDEV_POP 関数 [集合] [542 ページ]
数値式からなる母集団の標準偏差を DOUBLE として計算します。

STDDEV_SAMP 関数 [集合] [544 ページ]
数値式からなるサンプルの標準偏差を DOUBLE として計算します。

STR 関数 [文字列] [546 ページ]
指定した数に相当する文字列を返します。

STRING 関数 [文字列] [547 ページ]
1 つ以上の文字列を連結して 1 つの長い文字列にします。

STRTOUUID 関数 [文字列] [548 ページ]
文字列の値をユニークな識別子 (UUID または GUID) の値に変換します。

STUFF 関数 [文字列] [550 ページ]
1 つの文字列から複数の文字を削除して、別の文字列に置き換えます。

SUBSTRING 関数 [文字列] [551 ページ]
文字列の部分文字列を返します。

SUM 関数 [集合] [554 ページ]
ローグループごとに、指定された式の合計を返します。

SUSER_ID 関数 [システム] [555 ページ]
指定したユーザ名の数値のユーザ ID を返します。

SUSER_NAME 関数 [システム] [557 ページ]
指定したユーザ ID のユーザ名を返します。

SWITCHOFFSET 関数 [日付と時刻] [558 ページ]
元のタイムゾーンオフセットから指定のタイムゾーンオフセットに変換される TIMESTAMP WITH TIME ZONE 値を返します。

SYSDATETIMEOFFSET 関数 [日付と時刻] [559 ページ]

システムクロックを使用するデータベースサーバの現在の日付、時刻、およびタイムゾーンオフセットを返します。

TAN 関数 [数値] [560 ページ]
数のタンジェントを返します。

TEXTPTR 関数 [テキストとイメージ] [561 ページ]
指定したカラムへの 16 バイトのバイナリポインタを返します。この機能は Transact-SQL との互換性のためにのみ提供されているものであり、使用しないことをお奨めします。

TO_CHAR 関数 [文字列] [562 ページ]
任意のサポートされている文字セットの文字データを、データベースの CHAR 文字セットに変換します。

TO_NCHAR 関数 [文字列] [564 ページ]
任意のサポートされている文字セットの文字データを、NCHAR 文字セットに変換します。

TODATETIMEOFFSET 関数 [日付と時刻] [565 ページ]
指定のタイムゾーンオフセットを使用して、TIMESTAMP 値を TIME STAMP WITH TIME ZONE 値に変換します。

TODAY 関数 [日付と時刻] [566 ページ]
現在の日付を DATE 値で返します。

TRACEBACK 関数 [その他] [567 ページ]
ストアドプロシージャ、トリガ、またはカスタム関数の実行中に発生した最後の例外 (エラー) のスタックで、文を返します。

TRACED_PLAN 関数 [その他] (廃止予定) [568 ページ]
診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。この関数は、*SQL Central* でトレーシングデータを使用してクエリのグラフィカルなプランを生成するときに使用します。

TRANSACTSQL 関数 [その他] [569 ページ]
Transact-SQL に Watcom SQL 文を書き直します。

TREAT 関数 [データ型変換] [570 ページ]
ジオメトリ式の宣言されたタイプをサブタイプに変更します。この関数は、空間データで使用します。

TRIM 関数 [文字列] [572 ページ]
先行ブランクと後続ブランクを文字列から削除します。

TRIM_ARRAY 関数 [複合] [573 ページ]
配列で指定した数の要素で構成される、暗黙的にバインドされた配列を返します。

TRUNCNUM 関数 [数値] [574 ページ]
指定した桁数で小数点以下を切り捨てます。

TSEQUAL 関数 [システム] (旧式) [576 ページ]
2 つの TIMESTAMP 値を比較し、これらが同じかどうかを返します。

UCASE 関数 [文字列] [577 ページ]
文字列中のすべての文字を大文字に変換します。

UNICODE 関数 [文字列] [578 ページ]
文字列の最初の文字に Unicode のコードポイントを含む整数を返します。最初の文字が有効なエンコーディングではない場合は NULL を返します。

UNISTR 関数 [文字列] [579 ページ]
文字と Unicode エスケープシーケンスで構成される文字列を NCHAR 文字列に変換します。

UPPER 関数 [文字列] [581 ページ]

文字列中のすべての文字を大文字に変換します。

USER_ID 関数 [システム] [582 ページ]
指定したユーザ名の数値のユーザ ID を返します。

USER_NAME 関数 [システム] [583 ページ]
指定したユーザ ID のユーザ名を返します。

UUIDTOSTR 関数 [文字列] [584 ページ]
ユニークな識別子の値 (UUID または GUID) を文字列の値に変換します。

VAR_POP 関数 [集合] [586 ページ]
数値式からなる母集団の統計上の平方偏差を DOUBLE として計算します。

VAR_SAMP 関数 [集合] [588 ページ]
数値式からなるサンプルの統計上の平方偏差を DOUBLE として計算します。

VAREXISTS 関数 [その他] [590 ページ]
指定された名前のユーザ定義変数が存在する場合は、1 を返します。該当する変数が存在しない場合は、0 を返します。

VARIANCE 関数 [集合] [591 ページ]
VAR_SAMP のエイリアス。

WATCOMSQL 関数 [その他] [591 ページ]
Watcom SQL に Transact-SQL 文を書き直します。この関数は、既存の Adaptive Server Enterprise ストアドプロシージャを Watcom SQL 構文に変換するときに役立ちます。

WEEKS 関数 [日付と時刻] [592 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の週の数を返します。

WRITE_CLIENT_FILE 関数 [文字列] [594 ページ]
クライアントコンピュータ上にファイルを作成して書き込みます。

XMLAGG 関数 [集合] [596 ページ]
XML 値のコレクションから XML 要素のフォレストを生成します。

XMLCONCAT 関数 [文字列] [597 ページ]
XML 要素のフォレストを生成します。

XMLELEMENT 関数 [文字列] [599 ページ]
クエリ内の XML 要素を生成します。

XMLFOREST 関数 [文字列] [601 ページ]
XML 要素のフォレストを生成します。

XMLGEN 関数 [文字列] [602 ページ]
XQuery コンストラクタに基づいて XML 値を生成します。

YEAR 関数 [日付と時刻] [604 ページ]
TIMESTAMP 引数の年部分を返します。

YEARS 関数 [日付と時刻] [605 ページ]
TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の年数を返します。

YMD 関数 [日付と時刻] [607 ページ]
指定した年、月、日に相当する日付値を返します。引数は -32768 ～ 32767 の INTEGER です。

関連情報

関数のタイプ [199 ページ]

1.3.2.1 ABS 関数 [数値]

数値式の絶対値を返します。

構文

```
ABS( numeric-expression )
```

パラメータ

numeric-expression

絶対値が返される数値。

戻り値

数値式の絶対値。

数値式データ型	戻り値
INT	INT
FLOAT	FLOAT
DOUBLE	DOUBLE
NUMERIC	NUMERIC

標準

ANSI/ISO SQL 標準

オプションの言語機能 T441 の一部です。

例

次の文は、値 66 を返します。

```
SELECT ABS( -66 );
```

1.3.2.2 ACOS 関数 [数値]

数値式のアークコサインをラジアン単位で返します。

構文

```
ACOS( numeric-expression )
```

パラメータ

numeric-expression

角度のコサイン。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、0.52 のアークコサイン値を返します。

```
SELECT ACOS( 0.52 );
```

関連情報

ASIN 関数 [数値] [240 ページ]
ATAN 関数 [数値] [242 ページ]
ATAN2 関数 [数値] [243 ページ]
COS 関数 [数値] [287 ページ]

1.3.2.3 ARGN 関数 [その他]

引数リストから選択された引数を返します。

構文

```
ARGN( integer-expression, expression [ ,...] )
```

パラメータ

integer-expression

式リスト内での引数の位置。

expression

関数に渡される任意のデータ型の式。すべて同じデータ型の式を指定してください。

戻り値

`integer-expression` の値を n とした場合、引数リストの n 番目の引数 (1 から開始) を返します。

備考

式のデータ型は任意ですが、すべて同じデータ型にしてください。integer-expression は、1 からリスト内の式の数までの範囲内で指定してください。範囲外の値を指定すると、NULL が返されます。複数の式は、カンマで区切って指定します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 6 を返します。

```
SELECT ARGN( 6, 1,2,3,4,5,6 );
```

1.3.2.4 ARRAY コンストラクタ [複合]

特定のデータ型の要素を返します。

構文

```
ARRAY(
expression [, expression ... ]
| single-column-query-expression
)
```

パラメータ

expression

ROW 型の要素 expression。

single-column-query-expression

単一のカラムを返すクエリ文。

戻り値

配列値

備考

すべての式が和両立である必要があります。

すべての式が NULL に初期化され、特定の配列要素内に値が明示的または暗黙的に配置されるまで NULL のままとなります。

ARRAY 型には他の ARRAY 値または ROW 値を含めることができます。または ROW 型の一部とすることができます。

FETCH 文は、配列への値の転送をサポートしています。個別の式の配列、配列全体、または配列の一部に値をフェッチすることができます。

特定の値または値のベクトルは、二重角括弧を使用して区別することができます。

標準

ANSI/ISO SQL 標準

標準になし。

Oracle

VARRAY は、ARRAY の同義語として使用することができます。

例

次の例は、配列の構成方法を示します。

```
SELECT FIRST f[[2]] FROM ( SELECT ARRAY( ID,Quantity ) FROM GROUPO.Products )
  AS dt( f ) ORDER BY f[[1]] ASC;
```

この例では、Products テーブルの各行について、どちらも整数であるカラム ID と Quantity カラムの値を使用して ARRAY コントラクタが ARRAY 型を構築します。結果は各行の配列の最初の要素別に順序付けされ、返される結果は、最小の最初の要素 (製品 ID 300) を持つ配列からの 2 番目の要素となります。single-column query expression から直接得られる配列を構成することもできます。

次の例は、配列を構成する代替方法を示します。

```
SELECT * FROM GROUPO.SalesOrders S WHERE ARRAY( SELECT P.ID FROM
GROUPO.Products P  JOIN GROUPO.SalesOrderItems SI ON( P.ID = SI.ProductID )AND
SI.ID = S.ID
ORDER  BY P.ID )< ARRAY ( SELECT ID FROM GROUPO.Products ORDER BY ID );
```

次の例では、クエリの SELECT リストが 3 つの配列を使用します。配列の 1 つにより GROUP BY 式が生成され、MAX 関数が他の配列を使用します。各 ARRAY 型は、結果がクライアントに返される前に特定の要素に対する参照を解除します。

```
SELECT FIRST ARRAY( Quantity )[[1]], MAX( ARRAY( ID,Quantity ) )[[1]],
      MAX( ARRAY( name,name ))[[2]] FROM Products GROUP BY ARRAY( Quantity )
      ORDER BY 1;
```

次の例は、FETCH 文を使用して次の配列に転送する方法を示します。

```
BEGIN
```

```
DECLARE product_orders ARRAY(10) OF ARRAY OF INTEGER;
DECLARE products ARRAY(10) OF INTEGER;
DECLARE greatest_orders INTEGER = 0;
DECLARE i INTEGER = 1;
DECLARE curs CURSOR FOR
  SELECT ProductID,
  ARRAY_AGG( Quantity ) AS Quantities
    FROM GROUPO.SalesOrderItems
    GROUP BY ProductID
    ORDER BY ProductID;
OPEN curs;
lp: LOOP
  FETCH NEXT curs INTO products[[i]], product_orders[[i]];
  IF SQLCODE <> 0 THEN LEAVE lp; END IF;
  IF i = 1 THEN
    SET greatest_orders = 1;
  ELSE
    IF CARDINALITY( product_orders[[greatest_orders]] )
        < CARDINALITY( product_orders[[i]] ) THEN
      SET greatest_orders = i;
    END IF;
  END IF;
  SET i = i + 1;
END LOOP;
IF greatest_orders >= 1 THEN
  SELECT * FROM GROUPO.Products WHERE ID = products[[greatest_orders]];
END IF;
END;
```

次の例は、ARRAY コンストラクタを使用して、配列を定数のリストで初期化しています。曜日の配列の 2 番目の要素が選択されています。

```
BEGIN
   DECLARE @dow ARRAY( 7 ) OF CHAR(3) = ARRAY( 'Sun', 'Mon', 'Tue', 'Wed', 'Thu',
'Fri', 'Sat' );
   SELECT @dow[[2]];
END
```

関連情報

複合データ型 ROW および ARRAY [178 ページ]
複合関数 [201 ページ]
複合型の比較 [190 ページ]
FETCH 文 [ESQL] [SP] [1101 ページ]

1.3.2.5 ARRAY_AGG 関数 [集合]

指定された expression と配列要素の型が一致するグループごとに、指定された expression から未バインドの 1 次元配列を作成します。

構文

```
ARRAY_AGG( expression
```

```
[ ORDER BY order-by-expression [ ASC | DESC ], ... ] )
```

パラメータ

expression

配列のベースとなる expression。配列は、expression からの最初のグループの値を持つ最初の要素で作成され、さらに 2 番目のグループの値を持つ 2 番目の要素で作成され、以下同様となります。

order-by-expression

expression により返されるローの順序を決定します。order-by-expression が指定されていない場合、返されるローの順序は決定できません。

order-by-expression

関数によって返された項目を並べ替えます。この引数の前にカンマは必要ありません。このため、delimiter-string を指定しない場合、使用が簡単になります。

order-by-expression には整数リテラルを指定できません。ただし、整数リテラルを含む変数を指定できます。

ORDER BY 句に定数が含まれている場合、それらの定数はオプティマイザによって解釈され、同義の ORDER BY 句に置き換えられます。たとえば、オプティマイザは ORDER BY 'a' を ORDER BY 式として解釈します。

クエリブロックに、有効な ORDER BY 句が指定された複数の集合関数が含まれているとき、それらの ORDER BY 句を単一の ORDER BY 句に論理的に結合できる場合は、そのクエリブロックを実行できます。たとえば、次の ORDER BY 句の場合は、

```
ORDER BY expression1, 'a', expression2
```

```
ORDER BY expression1, 'b', expression2, 'c', expression3
```

次の ORDER BY 句として結合されます。

```
ORDER BY expression1, expression2, expression3
```

戻り値

ARRAY

備考

配列要素は、最初の要素で始まる入力によって設定されます。

ARRAY_AGG は、その入力内の NULL 値を無視しません。NULL 値は、他の値のような個別の要素として配列に格納されます。グループが空白の場合、ARRAY_AGG 関数の結果には、そのグループの NULL 要素が含まれます。

ARRAY_AGG は Window 関数として使用できませんが、Window 関数の入力には使用できます。

UNNEST 配列演算子は配列から一連の行を作成し、他の関連する式を使用して各配列要素を処理するのに使用できます。

標準

ANSI/ISO SQL 標準

機能 S098。

例

次の文は、製品のすべての色のリストが含まれる配列を生成する方法を示しています。この配列をプロシージャにパラメータとして渡して、テーブル内に標準以外の色があったかどうかを確認することができます。

```
CREATE VARIABLE color_list ARRAY OF LONG VARCHAR;
SELECT ARRAY_AGG(DISTINCT Color) INTO color_list FROM Products;
```

関連情報

UNNEST 配列演算子 [26 ページ]
LIST 関数 [集合] [420 ページ]

1.3.2.6 ARRAY_MAX_CARDINALITY 関数 [複合]

配列の要素の最大数を返します。

構文

```
ARRAY_MAX_CARDINALITY( array-expression )
```

パラメータ

array-expression

評価する配列式。

`array-expression` が NULL の場合は、ARRAY_MAX_CARDINALITY が NULL を返します。

戻り値

INTEGER

備考

コレクションのカーディナリティは、コレクションの要素数です。

未バインドの配列の場合は、ARRAY_MAX_CARDINALITY がサポートされている配列の最大サイズの制限を返します。バインドされている配列またはコンストラクタを使用して構成される配列の場合、ARRAY_MAX_CARDINALITY は明示的または暗黙的に宣言される配列の最大サイズを返します。

標準

ANSI/ISO SQL 標準

機能 S403。

関連情報

ARRAY_AGG 関数 [集合] [236 ページ]
CARDINALITY 関数 [複合] [262 ページ]
TRIM_ARRAY 関数 [複合] [573 ページ]

1.3.2.7 ASCII 関数 [文字列]

文字列式の最初のバイトの ASCII 値を整数で返します。

構文

```
ASCII( string-expression )
```

パラメータ

string-expression

文字列。

戻り値

SMALLINT

備考

文字列が空の場合は、0 を返します。リテラル文字列は、引用符で囲んで指定します。データベース文字セットがマルチバイトであり、パラメータ文字列の最初の文字が複数バイトから構成される場合、結果は NULL です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 90 を返します。

```
SELECT ASCII( 'Z' );
```

関連情報

文字列関数 [213 ページ]
CHAR 関数 [文字列] [268 ページ]

1.3.2.8 ASIN 関数 [数値]

数値のアークサインをラジアン単位で返します。

構文

```
ASIN( numeric-expression )
```

パラメータ

numeric-expression

角度のサイン。

戻り値

DOUBLE

備考

SIN 関数と ASIN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、0.52 のアークサイン値を返します。

```
SELECT ASIN( 0.52 );
```

関連情報

ACOS 関数 [数値] [232 ページ]
ATAN 関数 [数値] [242 ページ]
ATAN2 関数 [数値] [243 ページ]
SIN 関数 [数値] [527 ページ]

1.3.2.9 ATAN 関数 [数値]

数値のアークタンジェントをラジアン単位で返します。

構文

```
ATAN( numeric-expression )
```

パラメータ

numeric-expression

角度のタンジェント。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

ATAN 関数と TAN 関数は逆変換の演算です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、0.52 のアークタンジェント値を返します。

```
SELECT ATAN( 0.52 );
```

関連情報

ACOS 関数 [数値] [232 ページ]
ASIN 関数 [数値] [240 ページ]
ATAN2 関数 [数値] [243 ページ]
TAN 関数 [数値] [560 ページ]

1.3.2.10 ATAN2 関数 [数値]

2 つの数の比率のアークタンジェントをラジアン単位で返します。

構文

```
{ ATN2 | ATAN2 }( numeric-expression-1, numeric-expression-2 )
```

パラメータ

numeric-expression-1

アークタンジェントが計算される比率の分子。

numeric-expression-2

アークタンジェントが計算される比率の分母。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、比率 0.52 ～ 0.60 のアークタンジェント値を返します。

```
SELECT ATAN2( 0.52, 0.60 );
```

関連情報

ACOS 関数 [数値] [232 ページ]
ASIN 関数 [数値] [240 ページ]
ATAN 関数 [数値] [242 ページ]
TAN 関数 [数値] [560 ページ]

1.3.2.11 AVG 関数 [集合]

対象となるローセットの、数値式の平均値またはユニークな値からなるセットの平均値を計算します。

構文

数値式

```
AVG( [ ALL | DISTINCT ] numeric-expression )
```

Window 関数

```
AVG( [ ALL ] numeric-expression) OVER ( window-spec )
```

```
window-spec: 以下の備考部分を参照してください。
```

Ultra Light 数値式

```
AVG( [ DISTINCT ] numeric-expression )
```

パラメータ

[ALL] numeric-expression

各グループのローで平均値が計算される式。

DISTINCT 句

グループごとにユニークな数値の平均を計算します。

戻り値

グループにローが含まれていない場合は、NULL 値を返します。

引数が DOUBLE の場合は DOUBLE を、それ以外の場合は NUMERIC を返します。

備考

この平均には、numeric-expression が NULL 値であるローは含まれません。

この関数ではオーバーフローエラーが発生することがあり、これにより、エラーが返される場合があります。オーバーフローエラーを回避するために、numeric-expression で CAST 関数を使用できます。

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

コア機能。numeric-expression 構文は標準のコア機能ですが、window-spec 構文はオプションの言語機能 T611、"Basic OLAP operations" の一部で構成されます。カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの言語機能 F561、"Full value expressions" の一部です。このソフトウェアでは、言語機能 F441、"Extended set function support" もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。このソフトウェアでは、オプションの言語機能 F442、"Mixed column references in set function" がサポートされていません。また、AVG 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、SQL Anywhere 17 Demo に接続されると値 49988.623200 を返します。

```
SELECT AVG( Salary ) FROM Employees;
```

次の文は、SQL Anywhere 17 Demo データベースに接続されると Products テーブルの製品価格の平均を返します。

```
SELECT AVG( DISTINCT UnitPrice ) FROM Products;
```

次の文は、サブクエリからの限定式と外部 SELECT ブロックからの外部参照 (p.Quantity) の両方が AVG の引数に含まれているため、SQL Anywhere 17 Demo に接続されると、SQLSTATE 42W68 エラーを返します。

```
SELECT * from GROUPO.Products as p
WHERE p.Quantity > ( SELECT AVG( 0.5 * p.Quantity + 0.5 * s.Quantity )
                     from GROUPO.SalesOrderItems as s
                     WHERE s.ProductID = p.ProductID )
```

関連情報

CAST 関数 [データ型変換] [264 ページ]
SUM 関数 [集合] [554 ページ]
COUNT 関数 [集合] [289 ページ]

1.3.2.12 BASE64_DECODE 関数 [文字列]

MIME base64 フォーマットを使用してデータを復号化し、文字列を LONG VARCHAR として返します。

構文

```
BASE64_DECODE( string-expression )
```

パラメータ

string-expression

復号化される文字列。文字列は base64 エンコードである必要があります。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Embedded SQL プログラムからイメージテーブルにイメージを挿入します。入力データ (ホスト変数) は base64 エンコードである必要があります。

```
EXEC SQL INSERT INTO images ( image_data ) VALUES ( BASE64_DECODE ( :img ) );
```

関連情報

文字列関数 [213 ページ]
BASE64_ENCODE 関数 [文字列] [247 ページ]

1.3.2.13 BASE64_ENCODE 関数 [文字列]

MIME base64 フォーマットを使用してデータをエンコードし、そのデータを 7 ビット ASCII 文字列として返します。

構文

```
BASE64_ENCODE( string-expression )
```

パラメータ

string-expression

エンコードされる文字列。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、イメージを含む架空のテーブルからデータを取り出し、ASCII フォーマットで返します。結果として返される文字列は電子メールメッセージに埋め込まれ、元のイメージを取り出すために受信者によって復号化されます。

```
SELECT BASE64_ENCODE( image_data ) FROM IMAGES;
```

関連情報

文字列関数 [213 ページ]
BASE64_DECODE 関数 [文字列] [246 ページ]

1.3.2.14 BINTOHEX 関数 [データ型変換]

バイナリ文字列に相当する 16 進数を返します。

構文

```
BINTOHEX( binary-expression )
```

パラメータ

binary-expression

16 進数文字列に変換されるバイナリ文字列です。

戻り値

BINTOHEX 関数は、LONG VARCHAR 文字列を返します。結果の長さは、入力された文字列の長さの 2 倍になります。

備考

CAST、CONVERT、BINTOHEX、HEXTOBIN、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、313233 を含む文字列を返します。

```
SELECT BINTOHEX(0x313233);
```

関連情報

16 進値との変換 [12 ページ]
HEXTOBIN 関数 [データ型変換] [382 ページ]

1.3.2.15 BIT_AND 関数 [集合]

ローグループごとに、指定された式のビット処理 AND を返します。

構文

```
BIT_AND( bit-expression )
```

パラメータ

bit-expression

集約されるオブジェクト。この式は、VARBIT 配列、BINARY 値、または INTEGER (BIT や TINYINT などのすべての整数の変形を含む) にできます。

戻り値

引数と同じデータ型。比較される各ビット位置について、すべてのローでそのビット位置に 1 がある場合は 1 を返し、それ以外の場合は 0 を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、CHAR カラムを含む 4 つのローを生成し、値を VARBIT に変換します。

```
SELECT BIT_AND( CAST( row_value AS VARBIT ) )
FROM dbo.sa_split_list( '0001,0111,0100,0011' );
```

結果 0000 は次のように決定されます。

1. ロー 1 (0001) とロー 2 (0111) の間でビット処理 AND が実行され、結果は 0001 になります (両方の値の 4 番目のビットに 1 があります)。
2. 前の比較結果 (0001) とロー 3 (0100) の間でビット処理 AND が実行され、結果は 0000 になります (これらの値の同じビット位置に 1 がありませんでした)。
3. 前の比較結果 (0000) とロー 4 (0011) の間でビット処理 AND が実行され、結果は 0000 になります (これらの値の同じビット位置に 1 がありませんでした)。

関連情報

BIT_OR 関数 [集合] [251 ページ]
BIT_XOR 関数 [集合] [254 ページ]
ビット処理演算子 [31 ページ]

1.3.2.16 BIT_LENGTH 関数 [ビット配列]

配列に格納されているビット数を返します。

構文

```
BIT_LENGTH( bit-expression )
```

パラメータ

bit-expression

長さを決定するビット式。

戻り値

INT

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 8 を返します。

```
SELECT BIT_LENGTH( '01101011' );
```

関連情報

CHAR_LENGTH 関数 [文字列] [269 ページ]

1.3.2.17 BIT_OR 関数 [集合]

ローグループごとに、指定された式のビット処理 OR を返します。

構文

```
BIT_OR( bit-expression )
```

パラメータ

bit-expression

集約されるオブジェクト。この式は、VARBIT 配列、BINARY 値、または INTEGER (BIT や TINYINT などのすべての整数の変形を含む) にできます。

戻り値

引数と同じデータ型。この関数では、比較される各ビット位置について、いずれかのローでそのビット位置に 1 がある場合は 1 を返し、それ以外の場合は 0 を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、CHAR カラムを含む 4 つのローを生成し、値を VARBIT に変換します。

```
SELECT BIT_OR( CAST( row_value AS VARBIT ) )
FROM dbo.sa_split_list( '0001,0111,0100,0011' );
```

結果 0111 は次のように決定されます。

1. ロー 1 (0001) とロー 2 (0111) の間でビット処理 OR が実行され、結果は 0111 になります。
2. 前の比較結果 (0111) とロー 3 (0100) の間でビット処理 OR が実行され、結果は 0111 になります。
3. 前の比較結果 (0111) とロー 4 (0011) の間でビット処理 OR が実行され、結果は 0111 になります。

関連情報

BIT_AND 関数 [集合] [249 ページ]
BIT_XOR 関数 [集合] [254 ページ]
ビット処理演算子 [31 ページ]

1.3.2.18 BIT_SUBSTR 関数 [ビット配列]

ビット配列の部分配列を返します。

構文

```
BIT_SUBSTR( bit-expression [, start [, length ] ])
```

パラメータ

bit-expression

部分配列を抽出するビット配列。

start

返す部分配列の開始位置。負の開始位置は、配列の先頭からではなく、末尾からのビット数を指定します。配列の先頭ビットの位置を 1 とします。

length

返す部分配列の長さ。正の length は、開始位置から `length` ビット右側の位置で部分配列が終わることを指定し、負の length は、開始位置から最大 `length` ビット左側のビットを返します。

戻り値

LONG VARBIT

備考

`start` と `length` には、正または負の値を指定できます。負の数と正の数を適切に組み合わせて使用すると、文字列の先頭または末尾のどちらからでも部分配列を取得できます。`length` に負の値を使用しても、部分配列で返されるビットの順序には影響がありません。

`length` を指定すると、部分配列は指定した長さに限定されます。`start` が 0 で `length` が負でない場合は、1 の start 値が使用されます。`start` が 0 で `length` が負の場合は、-1 の start 値が使用されます。

`length` を指定しない場合、配列の末尾までが選択範囲になります。

BIT_SUBSTR 関数も次と同様ですが、より高速です。

```
CAST( SUBSTR( CAST( bit-expression AS VARCHAR ),
start [, length ] )
AS VARBIT );
```

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、1101 を返します。

```
SELECT BIT_SUBSTR( '001101', 3 );
```

次の文は、10110 を返します。

```
SELECT BIT_SUBSTR( '01011011101111011111', 2, 5 );
```

次の文は、11111 を返します。

```
SELECT BIT_SUBSTR( '01011011101111011111', -5, 5 );
```

関連情報

SUBSTRING 関数 [文字列] [551 ページ]

1.3.2.19 BIT_XOR 関数 [集合]

ローグループごとに、指定された式のビット処理 XOR を返します。

構文

```
BIT_XOR( bit-expression )
```

パラメータ

bit-expression

集約されるオブジェクト。この式は、VARBIT 配列、BINARY 値、または INTEGER (BIT や TINYINT などのすべての整数の変形を含む) にできます。

戻り値

引数と同じデータ型。比較される各ビット位置について、奇数個のローでそのビット位置に 1 がある場合は 1 を返し、それ以外の場合は 0 を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、CHAR カラムを含む 4 つのローを生成し、値を VARBIT に変換します。

```
SELECT BIT_XOR( CAST( row_value AS VARBIT ) )
FROM dbo.sa_split_list( '0001,0111,0100,0011' );
```

結果 `0001` は次のように決定されます。

1. ロー 1 (0001) とロー 2 (0111) の間でビット処理排他的 OR (XOR) が実行され、結果は 0110 になります。

2. 前の比較結果 (0110) とロー 3 (0100) の間でビット処理 XOR が実行され、結果は 0010 になります。
3. 前の比較結果 (0010) とロー 4 (0011) の間でビット処理 XOR が実行され、結果は 0001 になります。

関連情報

BIT_AND 関数 [集合] [249 ページ]
BIT_OR 関数 [集合] [251 ページ]
ビット処理演算子 [31 ページ]

1.3.2.20 BYTE_INSERTSTR 関数 [文字列]

別の文字列のバイト単位で指定された位置に文字列を挿入します。

構文

```
BYTE_INSERTSTR( insert-position, source-string, insert-string )
```

パラメータ

insert-position

このバイト位置の後に、`insert-string` が挿入されます。文字列の先頭バイトの位置を 0 とします。

source-string

`insert-string` が挿入される文字列。`source-string` の長さに制限はありません。

insert-string

挿入する文字列。

戻り値

LONG BINARY

備考

引数 `source-string` および `insert-string` は、バイナリ文字列として扱われます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、5 番目のバイト位置から 123456 を挿入し、値 `0xfedcba9876`**123456**`543210` を返します。

```
SELECT BYTE_INSERTSTR(5,0xfedcba9876543210,0x123456);
```

1.3.2.21 BYTE_LENGTH 関数 [文字列]

文字列のバイト数を返します。

構文

```
BYTE_LENGTH( string-expression )
```

パラメータ

string-expression

長さが計算される文字列。

戻り値

INT

備考

`string-expression` の後続空白スペースは、返される長さに含まれます。

NULL 文字列の戻り値は NULL です。

マルチバイト文字セットの文字列の場合、BYTE_LENGTH の値は、CHAR_LENGTH で返される文字数と異なることがあります。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light: Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。同等の関数は、OCTET_LENGTH 関数です。

例

次の文は、値 12 を返します。

```
SELECT BYTE_LENGTH( 'Test Message' );
```

関連情報

文字列関数 [213 ページ]
CHAR_LENGTH 関数 [文字列] [269 ページ]
DATALENGTH 関数 [システム] [301 ページ]
LENGTH 関数 [文字列] [417 ページ]

1.3.2.22 BYTE_LOCATE 関数

異なる文字列内のある BYTE 文字列の位置を返します。

構文

```
BYTE_LOCATE( source-string, search-string [, start-position ] )
```

パラメータ

source-string 検索される文字列。
search-string

検索する文字列。
start-position

文字列内の、検索を開始するバイト位置。最初のバイトの位置は 1 です。開始オフセットが負の場合、BYTE_LOCATE 関数は、文字列の末尾からカウントして、最初ではなく最後の一致文字列オフセットを返します。負のオフセットは、文字

列の末尾のどれだけの部分が検索から除外されるかを示します。除外されるバイト数は、(–1 * offset) –1 で計算されます。

start-position は検索が開始されたオフセットとして機能しますが、検索が開始された場所に関係なく、戻り値には一致する文字列の実際の開始位置が反映されます。

戻り値

INTEGER

備考

start-position を指定すると、文字列のオフセットから検索が開始します。

source-string の長さに制限はありませんが、search-string は 255 バイトに制限されます。search-string が 255 バイトより長い場合、この関数は NULL 値を返します。search-string が見つからない場合は、0 を返します。長さが 0 の search-string を検索すると、1 を返します。いずれかの引数が NULL の場合、結果も NULL になります。

source-string と search-string は、バイナリデータ型に変換可能などのデータ型でもかまいません。バイナリ比較が使用されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

start-position が 0 ～ 8 (これらの値を含む) の任意の正の数値の場合、次の文は 8 を返します。これは、文字列の最初の一致バイトの位置が 8 であることを示しています。start-position が 8 より大きい場合、位置 8 より後に文字列 'party' が見つからないため、この例は 0 を返します。

```
SELECT BYTE_LOCATE(
   'office party this week - rsvp as soon as possible',
   'party',
   2 );
```

start-position が 0 ～ -38 (これらの値を含む) の任意の数値の場合、次の文は 8 を返します。-38 より小さい数値の場合、0 を返します。これは、文字列の最初の一致バイトの位置が 8 であることと、最後の一致バイト ('party' の 'y') の位置が、文字列の末尾から後方にカウントして 38 であることを示しています。

```
SELECT BYTE_LOCATE(
   'office party this week - rsvp as soon as possible',
   'party',
   -38 );
```

1.3.2.23 BYTE_REPLACE 関数

文字列を別の文字列で置換し、新しい結果を返します。

構文

```
BYTE_REPLACE( source-string, search-string, replace-string )
```

パラメータ

source-string

検索される文字列。

search-string

検索して replace-string に置換する文字列。search-string は 255 バイトに制限されています。search-string が空の文字列の場合は、source-string を未変更のまま返します。

replace-string

search-string のすべてのインスタンスを置換する文字列。replacement-string が空の文字列の場合は、すべての search-string が削除されます。

戻り値

LONG BINARY

備考

search-string のすべてのインスタンスが replace-string で置換されます。

source-string、search-string、replace-string は、バイナリデータ型に変換可能などのデータ型でもかまいません。バイナリ比較が使用されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 xx.def.xx.ghi を返します。

```
SELECT BYTE_REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' );
```

1.3.2.24 BYTE_STUFF 関数 [文字列]

1 つの文字列から複数のバイトを削除して、別のバイトに置き換えます。

構文

```
BYTE_STUFF( source-string, start-position, length, insert-string )
```

パラメータ

source-string

BYTE_STUFF 関数によって変更されるバイト文字列。source-string の長さに制限はありません。

start-position

削除を開始する文字のバイト位置。文字列の先頭バイトの位置を 1 とします。

length

削除するバイト数。

insert-string

挿入する文字列。BYTE_STUFF 関数を使用して文字列の一部を削除するには、NULL の置換文字列を使用します。

戻り値

LONG BINARY

備考

引数 source-string および insert-string は、バイナリデータ型に変換可能などのデータ型でもかまいません。バイナリ文字列として扱われます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、5 番目のバイト位置から 123456 を挿入し、値 `0xfedcba9876`**123456**`543210` を返します。

```
SELECT BYTE_STUFF(0xfedcba9876543210, 6, 0, 0x123456);
```

1.3.2.25 BYTE_SUBSTR 関数 [文字列]

文字列の部分文字列を返します。部分文字列は、文字ではなくバイトを使用して計算されます。

構文

```
BYTE_SUBSTR( source-string, start-position [, length ] )
```

パラメータ

source-string

部分文字列が取得される文字列。`source-string` の長さに制限はありません。

start-position

部分文字列の開始位置を示す整数式。正の整数は、文字列の先頭から開始し、最初の文字は位置 1 です。負の整数は、文字列の末尾から始まる部分文字列を指定し、最後の文字の位置は -1 です。

length

部分文字列の長さを示す整数式。正の `length` は、取得するバイト数を開始位置から開始することを指定します。負の `length` は、開始位置から最大 `length` バイト左側の文字を返します。

戻り値

入力に応じて BINARY または LONG BINARY。

備考

length を指定すると、部分文字列は指定したバイト数に制限されます。start-position と length には、正または負の値を指定できます。負の数と正の数を適切に組み合わせて使用して、文字列の先頭または末尾のどちらかから部分文字列を取得します。

start-position が 0 で length が負でない場合は、1 の start-position 値が使用されます。start-position が 0 で length が負の場合は、-1 の start 値が使用されます。

引数 source-string は、バイナリデータ型に変換可能などのデータ型でもかまいません。バイナリ文字列として扱われます。部分文字列は、文字ではなくバイトを使用して計算されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 Test を返します。

```
SELECT BYTE_SUBSTR( 'Test Message', 1, 4 );
```

関連情報

文字列関数 [213 ページ]
SUBSTRING 関数 [文字列] [551 ページ]

1.3.2.26 CARDINALITY 関数 [複合]

NULL を含む、割り当てられた値を持つ配列の最大数を返します。

構文

```
CARDINALITY( array-expression )
```


パラメータ

array-expression

カーディナリティのある配列式が計算されます。

`array-expression` が NULL の場合は、CARDINALITY が NULL を返します。

戻り値

INTEGER

備考

コレクションのカーディナリティは、コレクションの要素数です。

その結果は、ゼロと配列の最大サイズの間となります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、値 4 を返します。

```
SELECT CARDINALITY( ARRAY( 3,4 ) || ARRAY( 5,6 ) );
```

関連情報

ARRAY_AGG 関数 [集合] [236 ページ]
ARRAY_MAX_CARDINALITY 関数 [複合] [238 ページ]
TRIM_ARRAY 関数 [複合] [573 ページ]

1.3.2.27 CAST 関数 [データ型変換]

指定されたデータ型に変換した式の値を返します。

構文

```
CAST( expression AS datatype )
```

パラメータ

expression

変換される式。

datatype

キャスト後の式のデータ型。データ型を明示的にセットするか、%TYPE 属性を指定して、データ型をテーブルまたはビュー内のカラムのデータ型、または変数のデータ型に設定します。

戻り値

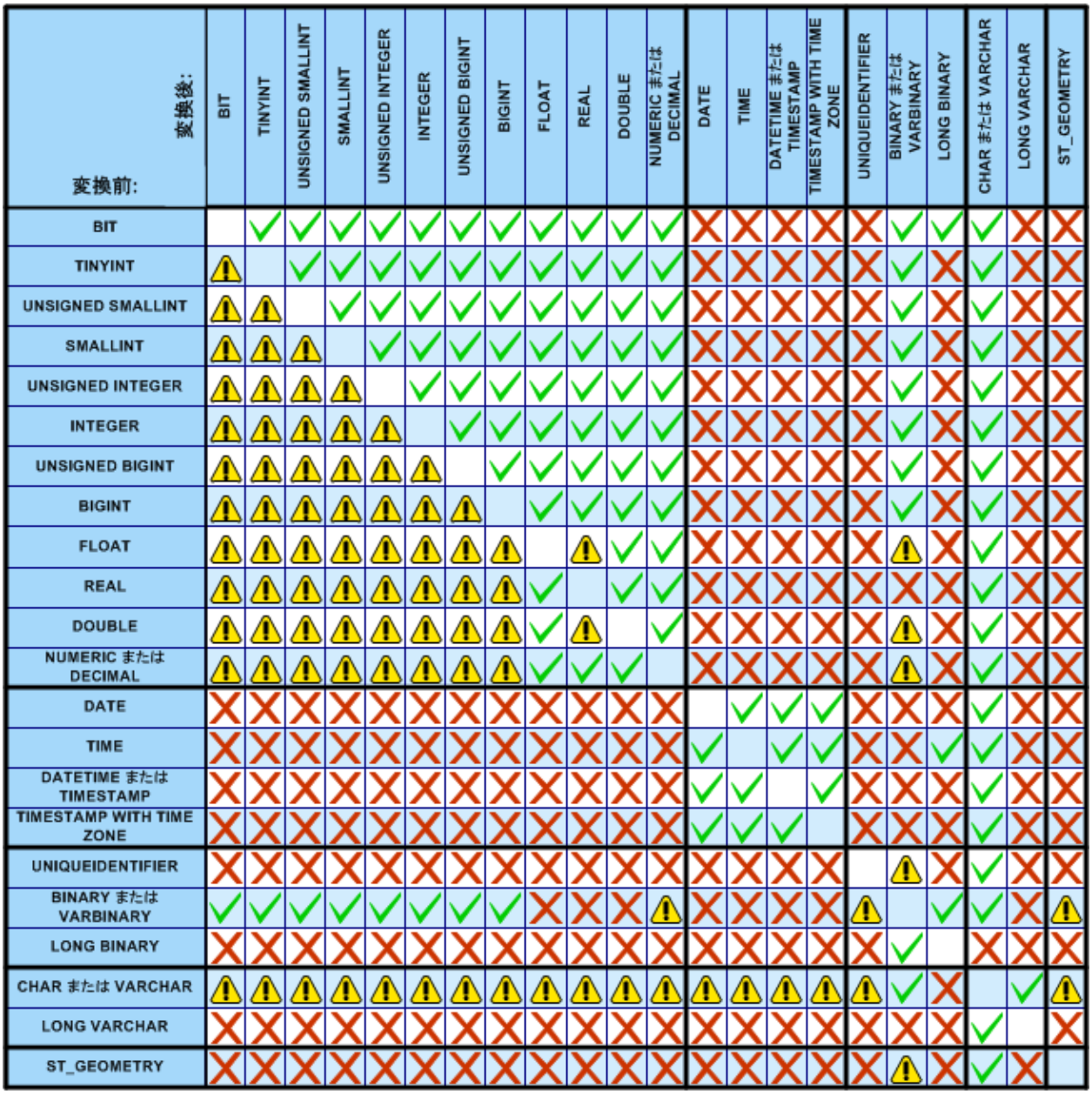
要求されたデータ型によって異なります。

備考

CAST 関数を使用して文字列をトランケートする場合、string_rtruncation データベースオプションは OFF に設定する必要があります。OFF に設定されていない場合はエラーになります。文字列をトランケートするには、LEFT 関数を使用します。

文字列型の長さを指定しない場合は、適切な長さが選択されます。DECIMAL 変換で精度も位取りも指定しない場合は、データベースサーバによって適切な値が選択されます。

Ultra Light: CAST 関数で精度と位取りを明示的に示すことをおすすめします。変換できるかどうかは、変換で使用する値によります。変換エラーが発生しないように、元のデータ型の値は新しいデータ型と互換性がある必要があります。次のチャートを使用して、変換がサポートされているかどうかを判断します。



シンボル	互換性
✓	常に変換
✗	変換しない

シンボル	互換性
⚠	値依存

注記

Ultra Light の場合:

- VARBINARY と UNIQUEIDENTIFIER 間で変換を行うには、VARBINARY 値が 16 バイト長である必要があります。
- NUMERIC と VARBINARY 間で変換を行うには、NUMERIC ソースが BIGINT としてもキャスト可能な値を持っている必要があります。
- VARCHAR から ST_GEOMETRY に変換するには、VARCHAR ソースが WKT フォーマットまたは EWKT フォーマットのいずれかで有効なジオメトリを表している必要があります。
- VARBINARY から ST_GEOMETRY に変換するには、VARBINARY ソースが WKB フォーマットで有効なジオメトリを表している必要があります。
- WKB または WKT でフォーマットされているソースから ST_GEOMETRY にキャストするには、0 の SRID が ST_GEOMETRY 値に割り当てられている必要があります。ST_GEOMETRY からキャストする場合、VARCHAR 値は EWKT でフォーマットされ、VARBINARY 値は WKB でフォーマットされます。

HEXTOINT および INTTOHEX 関数を使用すると、16 進値変換を行うことができます。

標準

ANSI/ISO SQL 標準

コア機能。ただし、このソフトウェアの CAST では、ANSI/ISO SQL 標準で許可されていない数多くのデータ型変換がサポートされています。たとえば、整数値を DATE データ型にキャストできますが、ANSI/ISO SQL 標準ではこのような変換はできません。

例

次の関数は、文字列を日付として使用することを保証します。

```
SELECT CAST( '2000-10-31' AS DATE );
```

式 1 + 2 の値を計算し、その結果を 1 文字の文字列にキャストします。

```
SELECT CAST( 1 + 2 AS CHAR );
```

VARCHAR と ST_GEOMETRY 間のキャストは通常暗黙的に行われます。たとえば、次の文は、ST_POINT 関数と VARCHAR を使用して ST_GEOMETRY カラムに値を追加します。それぞれの値はテーブルカラム全体で ST_GEOMETRY データ型に暗黙的にキャストされますが、その結果は依然として VARCHAR として示されます。

```
INSERT INTO T1 VALUES (2, ST_POINT(1,2,0), 'SRID=2163;Point(1 2)');
```

次の文は、Employees テーブルの BirthDate カラム (DATE データ型) に定義されているデータ型に値をキャストします。

```
SELECT CAST ( '1966-10-30' AS Employees.BirthDate%TYPE );
```

Ultra Light: 次のように CAST 関数を使用して、文字列を短縮できます。

```
SELECT CAST ( 'Surname' AS CHAR(5) );
```

関連情報

データ型変換 [192 ページ]
16 進値との変換 [12 ページ]
CONVERT 関数 [データ型変換] [283 ページ]
LEFT 関数 [文字列] [416 ページ]

1.3.2.28 CEILING 関数 [数値]

指定した値以上になる最初の整数を返します。正の数値の場合、この処理は丸めとも呼ばれています。

構文

```
{ CEILING | CEIL } ( numeric-expression )
```

パラメータ

numeric-expression

切り上げ値を計算する対象の数値。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準

ANSI/ISO SQL 標準

CEILING 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 60 を返します。

```
SELECT CEILING( 59.84567 );
```

関連情報

FLOOR 関数 [数値] [371 ページ]

1.3.2.29 CHAR 関数 [文字列]

数値の ASCII 値を持つ文字を返します。

構文

```
CHAR( integer-expression )
```

パラメータ

integer-expression

ASCII 文字に変換される数値。0 ～ 255 の範囲内の数を指定します。

戻り値

VARCHAR

備考

返される文字は、バイナリソート順に従って、現在のデータベース側文字セットの指定した数値式に対応しています。

整数式の値が 255 より大きいか、0 より小さい場合、CHAR は NULL を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文を実行すると、値 Y が返ります。

```
SELECT CHAR( 89 );
```

関連情報

文字列関数 [213 ページ]

1.3.2.30 CHAR_LENGTH 関数 [文字列]

文字列の文字数を返します。

構文

```
CHAR_LENGTH ( string-expression )
```

パラメータ

string-expression

長さが計算される文字列。

戻り値

INT

備考

後続空白スペースは、返される長さに含まれます。

NULL 文字列の戻り値は NULL です。

マルチバイト文字セットの文字列の場合、CHAR_LENGTH 関数で返される値は、BYTE_LENGTH 関数で返されるバイト数と異なることがあります。

データ型が CHAR、VARCHAR、LONG VARCHAR、NCHAR の場合、CHAR_LENGTH 関数と LENGTH 関数を使用すると同じ結果が得られます。ただし、BINARY データ型とビット配列データ型には LENGTH 関数を使用します。この関数は NCHAR の入力または出力をサポートしています。

Ultra Light: データ型が CHAR、VARCHAR、LONG VARCHAR の場合、CHAR_LENGTH 関数と LENGTH 関数を使用すると同じ結果が得られます。ただし、BINARY データ型とビット配列データ型には LENGTH 関数を使用します。

標準

ANSI/ISO SQL 標準

CHAR_LENGTH はコア機能です。NCHAR データ型の式での CHAR_LENGTH の使用は、オプションの ANSI/ISO SQL 言語機能 F421 の一部です。

例

次の文は、値 8 を返します。

```
SELECT CHAR_LENGTH( 'Chemical' );
```

関連情報

文字列関数 [213 ページ]
BYTE_LENGTH 関数 [文字列] [256 ページ]

1.3.2.31 CHARINDEX 関数 [文字列]

ある文字列内で 1 つの文字列の位置を返します。

構文

```
CHARINDEX( string-expression-1, string-expression-2 )
```

パラメータ

string-expression-1

検索する文字列。256 バイト未満の値にしてください。

string-expression-2

検索される文字列。

戻り値

INT

備考

`string-expression-1` の先頭文字の位置を 1 とします。検索される文字列内に、検索する文字列のインスタンスが複数存在する場合、CHARINDEX 関数は最初のインスタンスの位置を返します。

検索される文字列内に、検索する文字列が存在しない場合、CHARINDEX 関数は 0 を返します。

いずれかの引数が NULL の場合、結果も NULL になります。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、姓が K という文字で始まる場合にのみ、Employees テーブルの Surname カラムと GivenName カラムから姓名を返します。

```
SELECT Surname, GivenName
FROM GROUPO.Employees
WHERE CHARINDEX( 'K', Surname ) = 1;
```

次の結果が返されます。

Surname	GivenName
Klobucher	James

Surname	GivenName
Kuo	Felicia
Kelly	Moira

関連情報

文字列関数 [213 ページ]
SUBSTRING 関数 [文字列] [551 ページ]
REPLACE 関数 [文字列] [502 ページ]
LOCATE 関数 [文字列] [423 ページ]

1.3.2.32 COALESCE 関数 [その他]

リストの中から NULL でない最初の式を返します。この関数は ISNULL 関数と同じです。

構文

```
COALESCE( expression, expression [ ,...] )
```

パラメータ

expression

任意の式。

2 つ以上の式を関数に渡します。すべての式は比較可能であることが必要です。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、リストから NULL でない最初の式を返します。データベースサーバは、比較が可能な共通のデータ型を見つけることができないと、エラーを返します。

備考

結果が NULL になるのはすべての引数が NULL の場合のみです。

このパラメータにはスカラ型を指定できますが、同じ型を指定する必要はありません。

標準

ANSI/ISO SQL 標準

コア機能。

例

次の文は、値 34 を返します。

```
SELECT COALESCE( NULL, 34, 13, 0 );
```

関連情報

ISNULL 関数 [その他] [410 ページ]

1.3.2.33 COMPARE 関数 [文字列]

代替照合規則に基づいて 2 つの文字列を比較できます。

構文

```
COMPARE(
string-expression-1,
string-expression-2
[, { collation-id | collation-name[(collation-tailoring-string)] } ]
)
```

パラメータ

string-expression-1

最初の文字列式。

string-expression-2

2 番目の文字列式。

文字列式には、データベース側文字セットでエンコードされた文字のみを指定できます。

collation-id

使用するソート順を指定する変数または定数 (整数)。`collation-id` は、組み込みの照合にのみ使用できます。

照合名または照合 ID を指定しない場合のデフォルトは、デフォルト Unicode マルチ言語です。

collation-name

使用する照合順の名前を指定する文字列または文字変数。また、char_collation または db_collation (たとえば、`COMPARE( 'abc', 'ABC', 'char_collation' );`) を指定して、データベースが使用している CHAR の照合順を使用できます。同様に、nchar_collation を指定して、データベースで使用している NCHAR の照合順を使用できます。

collation-tailoring-string

オプションで、文字列の比較を詳細に制御することを目的に、照合の適合化オプション (`collation-tailoring-string`) を指定できます。これらのオプションは、キーワード=値 の形式で、カッコで囲んで指定して、その後ろに照合名を記述します。たとえば、`'UCA(locale=es;case=LowerFirst;accent=respect)'` のように記述します。これらのオプションの組み合わせを指定する構文は、CREATE DATABASE 文の COLLATION 句を定義する構文と同じです。

> **注記**
>
> UCA 照合を指定すると、照合の適合化のすべてのオプションがサポートされます。その他の照合の場合、大文字小文字の区別の適合化オプションのみがサポートされます。

戻り値

選択した照合規則に基づく INTEGER。

値	意味
1	`string-expression-1` は `string-expression-2` よりも大きい
0	`string-expression-1` は `string-expression-2` と等しい
-1	`string-expression-1` は `string-expression-2` より小さい

備考

COMPARE 関数は、データベースのブランク埋め込みが有効になっている場合でも、空の文字列とスペースだけの文字列を同等と見なしません。COMPARE 関数は SORTKEY 関数を使用して、比較に使用する照合キーを生成します。したがって、空の文字列、1 つのスペースを持った文字列、2 つのスペースを持った文字列は、等しいと見なされません。

`string-expression-1` または `string-expression-2` が NULL 値の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、COMPARE 関数を使用して 3 つの比較を行います。

```
SELECT COMPARE( 'abc','ABC','UCA(case=LowerFirst)' ),
       COMPARE( 'abc','ABC','UCA(case=Ignore)' ),
       COMPARE( 'abc','ABC','UCA(case=UpperFirst)' );
```

戻り値は -1、0、1 で、それぞれ比較の結果を示します。最初の比較の結果は -1 です。これは、*string-expression-2* ('ABC') が *string-expression-1* ('abc') よりも小さいことを示します。最初の COMPARE 文で、大文字と小文字の区別が LowerFirst に設定されているため、このような結果となります。

関連情報

文字列関数 [213 ページ]
SORTKEY 関数 [文字列] [531 ページ]

1.3.2.34 COMPRESS 関数 [文字列]

文字列を圧縮し、LONG BINARY 型の値を返します。

構文

```
COMPRESS( string-expression [ , compression-algorithm-alias ] )
```

パラメータ

string-expression

圧縮される文字列。この関数にはバイナリ値を渡すことができます。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

compression-algorithm-alias

圧縮に使用するアルゴリズムのエイリアス。サポートされている値は zip と gzip です (いずれも同じアルゴリズムに基づいていますが、ヘッダと後書きが異なります)。zip は幅広くサポートされている圧縮アルゴリズムです。gzip は UNIX の gzip ユーティリティと互換性がありますが、zip アルゴリズムには互換性がありません。

解凍は同じアルゴリズムで実行します。

戻り値

LONG BINARY

備考

COMPRESS によって返される値は判読できません。返された値が元の文字列より長い場合、その最大サイズは元の文字列 + 12 バイトよりも 0.1% を超えて大きくなることはありません。DECOMPRESS 関数を使用して、圧縮された `string-expression` を解凍できます。

圧縮された値をテーブルに格納する場合は、文字セット変換がデータに対して実行されないように、カラムを BINARY または LONG BINARY にしてください。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、文字列 'Hello World' を gzip アルゴリズムを使用して圧縮して作成したバイナリ文字列の長さを返します。この例は、圧縮すると値の長さが短くなるかどうかを判断する場合に便利です。

```
SELECT LENGTH( COMPRESS( 'Hello world', 'gzip' ) );
```

関連情報

文字列関数 [213 ページ]
DECOMPRESS 関数 [文字列] [325 ページ]

1.3.2.35 CONFLICT 関数 [その他]

カラムが、SQL Remote 環境で統合データベースに対して実行される UPDATE の競合の原因であるかどうかを示します。

構文

```
CONFLICT( column-name )
```

パラメータ

column-name

競合をテストされるカラムの名前。

戻り値

カラムが SQL Remote Message Agent によって実行される UPDATE 文の VERIFY リストにあり、またその文の VALUES リストで提供されている値が更新されるローのカラムの元の値と一致しない場合に TRUE を返します。それ以外の場合は、FALSE を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

CONFLICT 関数は、エラーメッセージを回避するために、SQL Remote RESOLVE UPDATE トリガで使用します。
CONFLICT 関数の使用法を示すために、次のテーブルについて考えます。

```
CREATE TABLE Admin (
  PKey bigint NOT NULL DEFAULT GLOBAL AUTOINCREMENT,
  TextCol CHAR(20) NULL, PRIMARY KEY ( PKey ) );
```

統合データベースとリモートデータベースの両方で、Admin テーブルに次のローがあると想定します。

```
1, 'Initial'
```

ここで、統合データベースでローを次のように更新します。

```
UPDATE Admin SET TextCol = 'Consolidated Update' WHERE PKey = 1;
```

リモートデータベースで、ローを次のように別の値に更新します。

```
UPDATE Admin SET TextCol = 'Remote Update' WHERE PKey = 1;
```

次に、リモートデータベースで dbremote を実行します。これによって、統合データベースで実行される次の文を含むメッセージファイルが生成されます。

```
UPDATE Admin SET TextCol='Remote Update'
VERIFY ( TextCol )
VALUES ( 'Initial' )
WHERE PKey=1;
```

SQL Remote Message Agent を統合データベースで実行し、この UPDATE 文を適用すると、SQL Anywhere は、VERIFY 句と VALUES 句を使用して、RESOLVE UPDATE トリガが起動するかどうかを決定します。RESOLVE UPDATE トリガは、統合データベースに対して SQL Remote Message Agent から更新が実行された場合にのみ起動します。次に、RESOLVE UPDATE トリガを示します。

```
CREATE TRIGGER ResolveUpdateAdmin
RESOLVE UPDATE ON Admin
REFERENCING OLD AS OldConsolidated
    NEW AS NewRemote
    REMOTE as OldRemote
FOR EACH ROW BEGIN
  MESSAGE 'OLD';
  MESSAGE OldConsolidated.PKey || ',' || OldConsolidated.TextCol;
  MESSAGE 'NEW';
  MESSAGE NewRemote.PKey || ',' || NewRemote.TextCol;
  MESSAGE 'REMOTE';
  MESSAGE OldRemote.PKey || ',' || OldRemote.TextCol;
END;
```

統合データベースの TextCol カラムの現在の値 (`'Consolidated Update'`) は、関連付けられているカラムの VALUES 句の値 (`'Initial'`) と一致しないため、RESOLVE UPDATE トリガが起動します。

PKey カラムはリモートで実行された UPDATE 文で修正されず、このトリガからアクセスできる OldRemote.PKey 値がないため、このトリガは失敗します。

CONFLICT 関数は、次の値を返すことによってこのエラーの回避に役立ちます。

- OldRemote.PKey 値がない場合は、FALSE を返します。
- OldRemote.PKey 値があっても OldConsolidated.PKey と一致する場合は、FALSE を返します。
- OldRemote.PKey 値があり、OldConsolidated.PKey と異なる場合は、TRUE を返します。

CONFLICT 関数を使用して、トリガを次のように書き直し、エラーを回避できます。

```
CREATE TRIGGER ResolveUpdateAdmin
RESOLVE UPDATE ON Admin
REFERENCING OLD AS OldConsolidated
    NEW AS NewRemote
    REMOTE as OldRemote
FOR EACH ROW BEGIN
  message 'OLD';
  message OldConsolidated.PKey || ',' || OldConsolidated.TextCol;
  message 'NEW';
  message NewRemote.PKey || ',' || NewRemote.TextCol;
  message 'REMOTE';
  if CONFLICT( PKey ) then
    message OldRemote.PKey;
  end if;
  if CONFLICT( TextCol ) then
```

```
    message OldRemote.TextCol;
  end if;
END;
```

関連情報

CREATE TRIGGER 文 [982 ページ]

1.3.2.36 CONNECTION_EXTENDED_PROPERTY 関数 [文字列]

指定したプロパティの値を返します。オプションでプロパティ固有の文字列パラメータを指定できます。

構文

```
CONNECTION_EXTENDED_PROPERTY(
{ property-id | property-name }
[, property-specific-argument [, connection-id ] ]
)
```

パラメータ

property-id

接続プロパティの ID。

property-name

接続プロパティの名前。サポートされているプロパティ名は次のとおりです。

CharSet

指定した規格によって決まる接続の CHAR 文字セットのラベルを返します。使用できる値は、次のとおりです。ASE、IANA、MIME、JAVA、WINDOWS、UTR22、IBM、ICU。デフォルトは IANA です。ただし、データベース接続が TDS で作成された場合のデフォルトは ASE です。

NcharCharSet

指定した規格によって決まる接続の NCHAR 文字セットのラベルを返します。可能な値は CharSet で前述した一覧と同じです。

HasSecuredFeature 接続で `property-specific-argument` 引数の 1 つ以上の機能が保護されている場合、Yes を返します。`property-specific-argument` が NULL の場合、NULL を返します。

Progress

文が実行されている時間に関する情報を返します。`property-specific-argument` を指定し、続けて `connection-id` を指定すると、文の進行状況に固有の情報が返されます。

property-specific-argument

次の接続プロパティと関連付けられたオプションのプロパティ固有の文字列パラメータ。

HasSecuredFeature `feature-list`

これらの機能の 1 つ以上が保護されているかどうかを調べるための機能のリストを指定します。

Progress

PercentComplete

文の処理が完了した割合を取得するには、PercentComplete を指定します。

Completed

完了したユニット数を取得するには、Completed を指定します。

Total

処理予定のユニットの合計数を取得するには、Total を指定します。

Units

処理予定のページ数、ロー数、またはバイト数。

Elapsed

現在の経過時間 (ミリ秒で表示)。

Remaining

推定残り時間 (ミリ秒) を取得するには、Remaining を指定します。

Raw

上記の値すべての組み合わせを上に挙げた順にセミコロンで区切って取得するには、Raw を指定します。たとえば、`43;9728;22230;pages;5025;6138` のように記述します。

Formatted

ユーザが読むことができる形式で表示するには、を参照してください。例:

```
43% ( 9728 of 22230 pages ) complete after 00:00:05; estimated 00:00:06
remaining
```

残り時間がまだ推定されていないか、完了したユニット数が元の推定よりも大きい場合、Remaining 値は空になります。

Formatted を除くすべてのプロパティ固有の引数では、大きいバイト値がキロバイトやメガバイトに変換されません。

connection-id

データベース接続の接続 ID 番号。この値が指定されていない場合は、現在の接続の ID 番号が使用されます。

戻り値

拡張接続プロパティを返します。戻り値は VARCHAR です。

備考

プロパティ ID またはプロパティ名のどちらかが指定されている必要があります。

CONNECTION_EXTENDED_PROPERTY 関数は CONNECTION_PROPERTY に似ています。異なる点は、CONNECTION_EXTENDED_PROPERTY 関数がプロパティ固有の文字列パラメータをオプションで指定できることです。プロパティ固有の引数の解釈は、最初の引数で指定されたプロパティ ID またはプロパティ名によって異なります。

CONNECTION_EXTENDED_PROPERTY 関数を使用して、任意の接続プロパティ用に値を返すことができます。ただし、拡張の情報は、拡張のプロパティにのみ使用できます。

権限

現在の接続 ID に対してこの関数を実行する場合、権限は必要ありません。他の接続に対してこの関数を実行する場合は、SERVER OPERATOR または DROP MONITOR のシステム権限が必要です。

無効なパラメータ値を指定した場合、または必要とされるシステム権限のどれかがない場合は、NULL が戻ります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Java 規格によって決まる、現在の接続の CHAR 文字セットを返します。

```
SELECT CONNECTION_EXTENDED_PROPERTY( 'charset', 'Java' );
```

関連情報

CONNECTION_PROPERTY 関数 [システム] [281 ページ]
DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]
DB_PROPERTY 関数 [システム] [324 ページ]

1.3.2.37 CONNECTION_PROPERTY 関数 [システム]

特定の接続プロパティの値を文字列で返します。

構文

```
CONNECTION_PROPERTY(
{ property-id | property-name }
```

```
[ , connection-id ] )
```

パラメータ

property-id

接続プロパティの ID。

property-name

接続プロパティの名前。

connection-id

データベース接続の接続 ID 番号。この値が指定されていない場合は、現在の接続の ID 番号が使用されます。

戻り値

VARCHAR、LONG VARCHAR

備考

プロパティ ID またはプロパティ名のどちらかが指定されている必要があります。

権限

現在の接続 ID に対してこの関数を実行する場合、権限は必要ありません。他の接続に対してこの関数を実行する場合は、SERVER OPERATOR または DROP MONITOR のシステム権限が必要です。

無効なパラメータ値を指定した場合、または必要とされるシステム権限のどれかがない場合は、NULL が戻ります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、管理されている準備文の数を返します。

```
SELECT CONNECTION_PROPERTY( 'PrepStmt' );
```

関連情報

PROPERTY_NUMBER 関数 [システム] [474 ページ]

1.3.2.38 CONVERT 関数 [データ型変換]

指定されたデータ型に変換した式を返します。

構文

```
CONVERT( datatype, expression [ , format-style ] )
```

パラメータ

datatype

変換後の式のデータ型です。データ型を明示的にセットするか、%TYPE 属性を指定して、データ型をテーブルまたはビュー内のカラムのデータ型、または変数のデータ型に設定します。

expression

変換される式。

format-style

出力値に適用されるスタイルコードです。このパラメータを使用するのは、文字列を日付または時刻のデータ型に変換するとき、またはその反対方向に変換するときです。次の表は、サポートされるスタイルコードと、そのスタイルコードで作成される出力形式の表現です。スタイルコードは世紀を出力形式に含めるかどうかによって 2 つのカラムに分けられます (たとえば、06 と 2006 など)。

スタイルコード 0 は、引数が指定されない場合に使用されます。

100 以上の位なし (yy) のスタイルコード	100 以上の位あり (yyyy) のスタイルコード	出力形式
-	0 または 100	Mmm dd yyyy hh:nnAA
1	101	mm/dd/yy[yy]

100 以上の位なし (yy) のスタイルコード	100 以上の位あり (yyyy) のスタイルコード	出力形式
2	102	[yy]yy.mm.dd
3	103	dd/mm/yy[yy]
4	104	dd.mm.yy[yy]
5	105	dd-mm-yy[yy]
6	106	dd Mmm yy[yy]
7	107	Mmm dd, yy[yy]
8	108	hh:nn:ss
-	9 または 109	Mmm dd yyyy hh:nn:ss:sssAA
10	110	mm-dd-yy[yy]
11	111	[yy]yy/mm/dd
12	112	[yy]yymmdd
-	13 または 113	dd Mmm yyyy hh:nn:ss:sss (24 時間表記、ヨーロッパのデフォルト、ミリ秒、4 桁の年)
-	14 または 114	hh:nn:ss:sss (24 時間表記)
-	20 または 120	yyyy-mm-dd hh:nn:ss (24 時間表記、ODBC 標準、4 桁の年)
-	21 または 121	yyyy-mm-dd hh:nn:ss.sss (24 時間表記、ODBC 標準、ミリ秒、4 桁の年)

戻り値

指定されたデータ型によって異なります。

備考

CONVERT 関数を使用して、文字列の解析時にあいまいさがない場合、文字列を DATE、TIME、または TIMESTAMP データ型に変換できます。`format-style` が指定されている場合、データベースサーバはこれを文字列の解析方法のヒントとして使用します。データベースサーバは、文字列を明確に解析できない場合にエラーを返します。

Ultra Light: この関数は CAST 関数と同様ですが、形式スタイルを指定して、日付と時刻のデータ型を変換するのに使用できます。

標準

ANSI/ISO SQL 標準

CONVERT 関数は、ANSI/ISO SQL 標準で定義されています。ただし、標準の CONVERT は、入力文字列式を異なる文字セットにコード変換することを目的としており、この機能はこのソフトウェアで CSCONVERT 機能として実装されています。

例

次の文は、形式スタイルの使用方法を示します。

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 104 ) FROM GROUPO.SalesOrders;
```

OrderDate
16.03.2000
20.03.2000
23.03.2000
25.03.2000
...

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 7 ) FROM GROUPO.SalesOrders;
```

OrderDate
Mar 16, 00
Mar 20, 00
Mar 23, 00
Mar 25, 00
...

次の文は、整数への変換を示し、値 5 を返します。

```
SELECT CONVERT( integer, 5.2 );
```

次の文は、Employees テーブルの BirthDate カラム (DATE データ型) に定義されているデータ型に値を変換します。

```
SELECT CONVERT ( Employees.BirthDate%TYPE, '1966-10-30' );
```

関連情報

CAST 関数 [データ型変換] [264 ページ]
CSCONVERT 関数 [文字列] [297 ページ]

1.3.2.39 CORR 関数 [集合]

数値のペアのセットの相関係数を返します。

構文

```
CORR( dependent-expression, independent-expression)
```

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

`dependent-expression` と `independent-expression` はどちらも数値です。関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` と `independent-expression`) のセットに適用されます。次の計算が行われます。

COVAR_POP (`y`, `x`) / STDDEV_POP (`y`) * STDDEV_POP (`x`)

`y` は `dependent-expression` を表し、`x` は `independent-expression` を表します。

標準

ANSI/ISO SQL 標準

CORR 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の例は、年齢が所得レベルに関連付けられているかどうかを検出するために相関を実行して、値 0.44022675645996 を返します。

```
SELECT CORR( Salary, ( YEAR( NOW( ) ) - YEAR( BirthDate ) ) ) FROM
GROUPO.Employees;
```

関連情報

集合関数 [200 ページ]
COVAR_POP 関数 [集合] [294 ページ]
STDDEV_POP 関数 [集合] [542 ページ]

1.3.2.40 COS 関数 [数値]

引数で与えられた角度のコサインをラジアン単位で返します。

構文

```
COS( numeric-expression )
```

パラメータ

numeric-expression

角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、角度 0.52 ラジアンのコサイン値を返します。

```
SELECT COS( 0.52 );
```

関連情報

ACOS 関数 [数値] [232 ページ]
COT 関数 [数値] [288 ページ]
SIN 関数 [数値] [527 ページ]
TAN 関数 [数値] [560 ページ]

1.3.2.41 COT 関数 [数値]

引数で与えられた角度のコタンジェントをラジアン単位で返します。

構文

```
COT( numeric-expression )
```

パラメータ

numeric-expression

角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、0.52 のコタンジェント値を返します。

```
SELECT COT( 0.52 );
```

関連情報

COS 関数 [数値] [287 ページ]
SIN 関数 [数値] [527 ページ]
TAN 関数 [数値] [560 ページ]

1.3.2.42 COUNT 関数 [集合]

指定されたパラメータに従って、グループのロー数をカウントします。

構文

式

```
COUNT( [ * | [ ALL | DISTINCT ] expression ] )
```

Window 関数

```
COUNT( [ * | [ ALL ]expression ]) OVER ( window-spec )
```

```
window-spec: 以下の備考部分を参照してください。
```

Ultra Light 式

```
COUNT( [ * | [ DISTINCT ] expression ] )
```

パラメータ

*****

各グループの中のロー数を返します。セマンティック上、COUNT(*) と COUNT() は同義です。

[ALL] expression

`expression` の値が NULL でないロー数をグループごとに返します。

Ultra Light expression

`expression` の値が NULL でないロー数をグループごとに返します。

p

DISTINCT expression

`expression` の値が NULL でないすべてのローについて、重複しない `expression` 値の数をグループごとに返します。

戻り値

COUNT 関数は、INT データ型の値を返します。

COUNT は、NULL 値を返しません。グループにローがないか、またはグループにおいて `expression` の値がすべて NULL である場合、COUNT は 0 を返します。

備考

SQL Anywhere では、COUNT 関数は、最大値 2147483647 を返します。大規模な結果セットをカウントする場合、結果に含まれるローの数が多くなる可能性がある場合、またはオーバーフローの可能性がある場合には、COUNT_BIG 関数を使用してください。`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

コア機能。Window 関数として使用する場合、COUNT はオプションの ANSI/ISO SQL 言語機能 T611、"Basic OLAP operations" の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、"Full value expressions" の一部です。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、"Extended set function support" もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、"Mixed column references in set function" がサポートされていません。このソフトウェアでは、COUNT 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の文は、ユニークな各都市と、その都市で働く従業員数を返します。

```
SELECT City, COUNT( * ) FROM GROUPO.Employees GROUP BY City;
```

関連情報

WINDOW 句 [1409 ページ]
AVG 関数 [集合] [244 ページ]
SUM 関数 [集合] [554 ページ]
COUNT_BIG 関数 [集合] [291 ページ]

1.3.2.43 COUNT_BIG 関数 [集合]

指定されたパラメータに従って、グループのロー数をカウントします。

構文

```
COUNT_BIG(
[ * | [ ALL | DISTINCT ] expression ]
| [ * | [ ALL ] expression ]) OVER ( window-spec
)
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

*****

各グループの中のロー数を返します。セマンティック上、COUNT_BIG(*) と COUNT_BIG() は同義です。

[ALL] expression

`expression` の値が NULL でないロー数をグループごとに返します。

DISTINCT expression

`expression` の値が NULL でないすべてのローについて、重複しない `expression` 値の数をグループごとに返します。

戻り値

COUNT_BIG は、BIGINT データ型の値を返します。

COUNT_BIG は、NULL 値を返しません。グループにローがないか、またはグループにおいて `expression` の値がすべて NULL である場合、COUNT_BIG は 0 を返します。

備考

大規模な結果セットをカウントする場合、結果に含まれるローの数が多くなる可能性がある場合、またはオーバーフローの可能性がある場合には、COUNT_BIG 関数を使用することをお奨めします。それ以外の場合は、COUNT 関数を使用します。この関数の最大値は 2147483647 です。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

標準になし。

このソフトウェアでは、オプションの機能 F442、「Mixed column references in set function」がサポートされていません。このソフトウェアでは、COUNT_BIG 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、ユニークな各都市と、その都市で働く従業員数を返します。

```
SELECT City, COUNT_BIG( * ) FROM GROUPO.Employees GROUP BY City;
```

関連情報

AVG 関数 [集合] [244 ページ]
SUM 関数 [集合] [554 ページ]
COUNT 関数 [集合] [289 ページ]

1.3.2.44 COUNT_SET_BITS 関数 [ビット配列]

配列でビットが 1 (TRUE) に設定された数値のカウントを返します。

構文

```
COUNT_SET_BITS( bit-expression )
```

パラメータ

bit-expression

設定されたビットを決定するビット配列。

戻り値

UNSIGNED INT

備考

`bit-expression` が NULL の場合、NULL を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 4 を返します。

```
SELECT COUNT_SET_BITS( '00110011' );
```

次の文は、値 12 を返します。

```
SELECT COUNT_SET_BITS( '0011001111111111' );
```

1.3.2.45 COVAR_POP 関数 [集合]

数値のペアのセットの母共分散を返します。

構文

式

```
COVAR_POP( dependent-expression, independent-expression)
```

Window 関数

```
COVAR_POP( dependent-expression, independent-expression)
OVER( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

`dependent-expression` と `independent-expression` はどちらも数値です。関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` または `independent-expression`) ペアのセットに適用されます。その後、次の計算が行われます。

```
( SUM( y * x ) - SUM( x ) * SUM( y ) / n ) / n
```

`y` は `dependent-expression` を表し、`x` は `independent-expression` を表します。

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

COVAR_POP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の例は、従業員の年齢と給与間の関連の強さを測定します。この関数は、値 73785.84005866687 を返します。

```
SELECT COVAR_POP( Salary, ( YEAR( NOW( ) ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
COVAR_SAMP 関数 [集合] [295 ページ]
SUM 関数 [集合] [554 ページ]

1.3.2.46 COVAR_SAMP 関数 [集合]

数値のペアのセットの標本共分散を返します。

構文

式

```
COVAR_SAMP( dependent-expression, independent-expression)
```

Window 関数

```
COVAR_SAMP( dependent-expression, independent-expression)
OVER ( window-spec )
```

window-spec: 次の「備考」の項を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

`dependent-expression` と `independent-expression` はどちらも数値です。関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` または `independent-expression`) ペアのセットに適用されます。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

COVAR_SAMP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の例は、値 74782.9460054052 を返します。

```
SELECT COVAR_SAMP( Salary, ( 2008 - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
COVAR_POP 関数 [集合] [294 ページ]
SUM 関数 [集合] [554 ページ]

1.3.2.47 CSCONVERT 関数 [文字列]

文字列の文字セットを変換します。

構文

```
CSCONVERT(
string-expression,
target-charset-string [, source-charset-string [, options ] ] )
```

パラメータ

string-expression

変換される文字列。

target-charset-string

変換後の文字セット。target-charset-string は、サポートされている任意の文字セットラベルです。または以下を指定します。

os_charset

データベースサーバをホストしているオペレーティングシステムが使用する文字セットのエイリアス。

char_charset

データベースで使用される CHAR 文字セットのエイリアス。

nchar_charset

データベースで使用される NCHAR 文字セットのエイリアス。

options

次のいずれかのオプションを指定できます。

バイトオーダーマーク (BOM) の新規サポート

read_bom=on または **read_bom=off** を指定して、読み込みバイトオーダーマークをオンまたはオフにします。**write_bom=on** または **write_bom=off** を指定して、書き込みバイトオーダーマークをオンまたはオフにします。デフォルトでは、動作は read_bom=on および write_bom=off です。

source-charset-string

string-expression に使用される文字セット。デフォルトは db_charset (データベース側文字セット) です。
source-charset-string は、サポートされている任意の文字セットラベルです。または以下を指定します。

os_charset

データベースサーバをホストしているオペレーティングシステムが使用する文字セットのエイリアス。

char_charset

データベースで使用される CHAR 文字セットのエイリアス。

nchar_charset

データベースで使用される NCHAR 文字セットのエイリアス。

戻り値

LONG BINARY

備考

サポートされている文字セットのリストを表示するには、次のコマンドを実行します。

```
dbinit -le
```

標準

ANSI/ISO SQL 標準

標準になし。ANSI/ISO SQL 標準では、1 つの文字セットから別の文字セットへの文字列データの変換に CONVERT 関数 (ソフトウェアに用意された CONVERT 関数と混同しないこと) が使用され、この関数の引数は CSCONVERT のものとは異なります。

例

このフラグメントは、mytext カラムを繁体文字中国語文字セットから簡体文字中国語文字セットに変換します。

```
SELECT CSCONVERT( mytext, 'cp936', 'cp950' )
FROM mytable;
```

このフラグメントは、mytext カラムをデータベース側文字セットから簡体文字中国語文字セットに変換します。

```
SELECT CSCONVERT( mytext, 'cp936' )
FROM mytable;
```

ファイル名がデータベースに格納される場合は、そのデータベース側文字セットで格納されます。サーバが読み込みまたは書き込みを行うファイルの名前がデータベースに格納されている場合 (外部ストアドプロシージャなど)、ファイル名をオペレーティングシステムの文字セットに明示的に変換してから、ファイルにアクセスしてください。データベースに格納されているファイル名をクライアントが取得する場合は、クライアントの文字セットに自動的に変換されるため、明示的な変換は必要ありません。

このフラグメントは、ファイル名カラムの値をデータベース側文字セットからオペレーティングシステムの文字セットに変換します。

```
SELECT CSCONVERT( filename, 'os_charset' )
FROM mytable;
```

ファイル名のリストが入っているテーブルがあります。外部ストアドプロシージャは、このテーブルのファイル名をパラメータとして取り出し、そのファイルから直接情報を読み込みます。次の文は、文字セットの変換が必要ない場合に正しく動作します。

```
SELECT MYFUNC( filename )
FROM mytable;
```

`mytable` 句は、ファイル名カラムが含まれているテーブルを示します。ただし、ファイル名をオペレーティングシステムの文字セットに変換する必要がある場合は、次の文を使用します。

```
SELECT MYFUNC( csconvert( filename, 'os_charset' ) )
FROM mytable;
```

関連情報

文字列関数 [213 ページ]

1.3.2.48 CUME_DIST 関数 [ランキング]

ローのグループ内で 1 つの値の相対位置を計算します。

構文

```
CUME_DIST( ) OVER ( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

戻り値

0 ～ 1 の DOUBLE 値

備考

復号ソートキーは、現在 CUME_DIST 関数では使用できません。他の任意の RANK 関数では復号ソートキーを使用できます。

`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。Window 関数として使用する場合、ORDER BY 句を指定する必要があります。また PARTITION BY 句は指定できますが、ROWS 句や RANGE 句はいずれも指定できません。

標準

ANSI/ISO SQL 標準

CUME_DIST 関数は、オプションの SQL 言語機能 T612、「Advanced OLAP operations」の一部です。

例

次の例は、カリフォルニアに住む従業員の給与に関する累積分布を示す結果セットを返します。

```
SELECT DepartmentID, Surname, Salary,
CUME_DIST() OVER (PARTITION BY DepartmentID
ORDER BY Salary DESC) "Rank"
FROM GROUPO.Employees
WHERE State IN ('CA');
```

結果セットは次のとおりです。

DepartmentID	Surname	Salary	Rank
200	Savarino	72300.000	0.333333333333333
200	Clark	45000.000	0.666666666666667
200	Overbey	39300.000	1

関連情報

WINDOW 句 [1409 ページ]
DENSE_RANK 関数 [ランキング] [332 ページ]
PERCENT_RANK 関数 [ランキング] [465 ページ]
RANK 関数 [ランキング] [479 ページ]

1.3.2.49 DATALENGTH 関数 [システム]

式の結果に必要な基本となる記憶領域の長さ (バイト) を返します。

構文

```
DATALENGTH( expression )
```

パラメータ

expression

通常は、カラム名です。`expression` が文字列定数の場合は、引用符で囲みます。

戻り値

UNSIGNED INT

備考

DATALENGTH 関数の戻り値は次のとおりです。

データ型	DATALENGTH
BIT	1
TINYINT	1
SMALLINT	2
INTEGER	4
BIGINT	8
REAL	4
DOUBLE	8
TIME	8
DATE	4
TIMESTAMP	8
DATETIME	8
TIMESTAMP WITH TIME ZONE	29
UNIQUEIDENTIFIER	16

データ型	DATALENGTH
CHAR	データの長さ
VARCHAR	データの長さ
BINARY	データの長さ
VARBINARY	データの長さ
NCHAR	データの長さ
NVARCHAR	データの長さ
TEXT	データの長さ
NTEXT	データの長さ
IMAGE	データの長さ
XML	データの長さ

SQL Anywhere の場合、この関数は NCHAR の入力と出力をサポートしています。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、CompanyName カラムで最も長い文字列の長さ 27 を返します。

```
SELECT MAX( DATALENGTH( CompanyName ) )
FROM GROUPO.Customers;
```

次の文は、文字列 '8sdofinsv8s7a7s7gehe4h' の長さを返します。

```
SELECT DATALENGTH( '8sdofinsv8s7a7s7gehe4h' );
```

関連情報

SQL データ型 [124 ページ]

1.3.2.50 DATE 関数 [日付と時刻]

式を日付に変換し、時間、分、秒を削除します。

構文

```
DATE( expression )
```

パラメータ

expression

日付形式に変換される値。通常、文字列。

戻り値

DATE

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 1999-01-02 を日付として返します。

```
SELECT DATE( '1999-01-02 21:20:53' );
```

次の文は、SYSOBJECT システムビューにリストされるすべてのオブジェクトについて、作成日を返します。

```
SELECT DATE( creation_time ) FROM SYSOBJECT;
```

関連情報

SQL 文の式 [32 ページ]

1.3.2.51 DATEADD 関数 [日付と時刻]

日付の単位をその引数に追加することによって求められる TIMESTAMP または TIMESTAMP WITH TIME ZONE の値を返します。

構文

```
DATEADD( date-part, integer-expression, timestamp-expression )
```

```
date-part :
year
| quarter
| month
| week
| day
| dayofyear
| hour
| minute
| second
| millisecond
| microsecond
```

パラメータ

date-part

`integer-expression` があらわす日付要素。

integer-expression

`timestamp-expression` に追加する `date-part` 値の数。`integer-expression` には任意の数値型を指定できますが、値は INTEGER にトランケートされます。

timestamp-expression

変更される TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。

戻り値

`timestamp-expression` が TIMESTAMP WITH TIME ZONE の場合は TIMESTAMP WITH TIME ZONE を返し、それ以外の場合は TIMESTAMP を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は TIMESTAMP 値 2016/05/02 0:00:00.000 を返します。

```
SELECT DATEADD( month, 12, '2015/05/02' );
```

次の文は TIMESTAMP 値 2015/05/02 04:00:00.000 を返します。

```
SELECT DATEADD( hour, 4, '2015/05/02' );
```

次の文は、TIMESTAMP WITH TIME ZONE 値 2015/05/06 11:33:00.000+04:00 を返します。

```
SELECT DATEADD( day, 4, CAST( '2015/05/02 11:33:00.000000+04:00' as TIMESTAMP
WITH TIME ZONE ));
```

関連情報

日付の単位の指定 [206 ページ]

1.3.2.52 DATEDIFF 関数 [日付と時刻]

2 つの日付間の期間を返します。

構文

```
DATEDIFF( date-part, date-expression-1, date-expression-2 )
```

```
date-part :
year
| quarter
| month
| week
| day
| dayofyear
| hour
| minute
| second
| millisecond
| microsecond
```

パラメータ

date-part

期間を測定する日付の単位を指定します。

date-expression-1

期間の開始日。この値を date-expression-2 から引いて、2 つの引数の間の date-parts の数を返す。

date-expression-2

期間の終了日。この値から Date-expression-1 を引いて、2 つの引数の間の date-parts の数を返す。

戻り値

年、四半期、月、週、日、通し日数を示す INT。時、分、秒、ミリ秒、マイクロ秒を表す BIGINT。

備考

この関数は、指定した 2 つの日付間に存在する日付の単位の数を計算します。結果は、日付の単位の数を表す (date-expression-2 - date-expression-1) と同値の符号付き整数です。

DATEDIFF 関数の結果が日付の単位の整数倍でない場合、結果は丸められずに切り捨てになります。

day を日付の単位として使用する場合、DATEDIFF 関数は指定された 2 つの時刻の間の午前 0 時の回数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

month を日付の単位として使用する場合、DATEDIFF 関数は 2 つの日付の間に存在する月の初日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

week を日付の単位として使用する場合、DATEDIFF 関数は 2 つの日付の間に存在する日曜日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、1 を返します。

```
SELECT DATEDIFF( hour, '4:00AM', '5:50AM' );
```

次の文は、102 を返します。

```
SELECT DATEDIFF( month, '1987/05/02', '1995/11/15' );
```

次の文は、0 を返します。

```
SELECT DATEDIFF( day, '00:00', '23:59' );
```

次の文は、4 を返します。

```
SELECT DATEDIFF( day,
   '1999/07/19 00:00',
   '1999/07/23 23:59' );
```

次の文は、0 を返します。

```
SELECT DATEDIFF( month, '1999/07/19', '1999/07/23' );
```

次の文は、1 を返します。

```
SELECT DATEDIFF( month, '1999/07/19', '1999/08/23' );
```

次の例は、DATEDIFF 関数を使用し、GROUPO.Customers テーブルで 3 段階のジョインを行うミリ秒数を返す方法を示しています。出力は、データベースサーバウィンドウに送信されます。

```
BEGIN
    DECLARE startTime, endTime TIMESTAMP;
    DECLARE rowCount INT;

    SET startTime = CURRENT TIMESTAMP;
    SELECT count(*) INTO rowCount FROM GROUPO.Customers AS T1, GROUPO.Customers
AS T2, GROUPO.Customers AS T3;
    SET endTime = CURRENT TIMESTAMP;

    MESSAGE 'Time to count rows: ' || DATEDIFF( MILLISECOND, startTime, endTime )
|| ' ms';
END
```

関連情報

日付の単位の指定 [206 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]

1.3.2.53 DATEFORMAT 関数 [日付と時刻]

日付式を表す文字列を、指定した形式で返します。

構文

```
DATEFORMAT( datetime-expression, string-expression )
```

パラメータ

datetime-expression

変換される日時。

string-expression

変換後の日付の形式。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

戻り値

VARCHAR

備考

string-expression には、許容される任意の日付形式を使用できます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 Jan 01, 1989 を返します。

```
SELECT DATEFORMAT( '1989-01-01', 'Mmm dd, yyyy' );
```

1.3.2.54 DATENAME 関数 [日付と時刻]

TIMESTAMP または TIMESTAMP WITH TIME ZONE の値の特定部分の名前 (月の名前 "June" など) を文字列で返します。

構文

```
DATENAME( date-part, timestamp-expression )
```

パラメータ

date-part

名前が返される日付の単位。

timestamp-expression

日付の単位名が返される TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。意味のある結果を取得するには、`timestamp-expression` に必要な `date-part` を含めてください。

戻り値

VARCHAR

備考

結果が数値 (23 日など) の場合でも、DATENAME 関数は文字列を返します。

SQL Anywhere では、英語のロケールでは英語の名前が返され、英語以外のロケールでは英語以外の名前が返されます。たとえば、Language (LANG) 接続パラメータを使用すれば他の言語を指定できます。

日付の単位 TZOffset (TZ) が指定されていると、DATENAME はオフセットを{ + | - }hh:nn 形式の文字列で返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、英語のロケールの場合、`May` という値を返します。

```
SELECT DATENAME( month, '1987/05/02' );
```

SQL Anywhere では、ドイツ語のロケールの場合、`Mai` という値を返します。スペイン語のロケールの場合、`Mayo` という値を返します。複数のロケールがサポートされています。

次の文は、値 `-05:00` を返します。

```
SELECT DATENAME( TZ, CAST('2016/02/03 12:02:00-5:00' AS TIMESTAMP WITH TIME
ZONE) ) AS TZOffset;
```

関連情報

日付の単位の指定 [206 ページ]
DATEPART 関数 [日付と時刻] [310 ページ]
MONTHNAME 関数 [日付と時刻] [443 ページ]

1.3.2.55 DATEPART 関数 [日付と時刻]

TIMESTAMP または TIMESTAMP WITH TIME ZONE の値の一部を返します。

構文

```
DATEPART( date-part, timestamp-expression )
```

パラメータ

date-part

名前が返される日付の単位。

timestamp-expression

一部分が返される TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。

戻り値

INT

備考

意味のある結果を取得するには、`timestamp-expression` に必要な `date-part` 部分を含めてください。

曜日に対応する数値は、first_day_of_week データベースオプションの設定によって変わります。デフォルトは日曜日 = 7 です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 5 を返します。

```
SELECT DATEPART( month , '1987/05/02' );
```

次の例では、TableStatistics テーブルを作成し、SalesOrders テーブルに格納されている年ごとの注文の総数を挿入します。

```
CREATE TABLE TableStatistics (
   ID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
   Year INT,
   NumberOrders INT );
INSERT INTO TableStatistics ( Year, NumberOrders )
   SELECT DATEPART( Year, OrderDate ), COUNT(*)
   FROM GROUPO.SalesOrders
   GROUP BY DATEPART( Year, OrderDate );
```

関連情報

日付の単位の指定 [206 ページ]
SET 文 [T-SQL] [1327 ページ]
EXTRACT 関数 [日付と時刻] [367 ページ]
MICROSECOND 関数 [日付と時刻] [433 ページ]
MILLISECOND 関数 [日付と時刻] [435 ページ]

1.3.2.56 DATETIME 関数 [日付と時刻]

式を TIMESTAMP 値に変換します。

構文

```
DATETIME( expression )
```

パラメータ

expression

変換される式。通常は文字列です。

戻り値

TIMESTAMP

備考

数値を変換しようとするとエラーが返ります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 1998-09-09 12:12:12.000 のタイムスタンプを返します。

```
SELECT DATETIME( '1998-09-09 12:12:12.000' );
```

関連情報

SQL 文の式 [32 ページ]
CAST 関数 [データ型変換] [264 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIMEOFFSET データ型 [164 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.3.2.57 DAY 関数 [日付と時刻]

引数の日付を 1 ～ 31 の間の整数で返します。

構文

```
DAY( date-expression )
```

パラメータ

date-expression

DATE データ型の日付。

戻り値

SMALLINT

備考

DAY 関数は、引数の日付に対応する 1 ～ 31 の間の整数を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 12 を返します。

```
SELECT DAY( '2001-09-12' );
```

1.3.2.58 DAYNAME 関数 [日付と時刻]

日付から曜日の名前を返します。

構文

```
DAYNAME( date-expression )
```

パラメータ

date-expression

日付。

戻り値

VARCHAR

備考

返される曜日名は、日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日です。

SQL Anywhere は、英語のロケールでは英語の名前を、英語以外のロケールでは英語以外の名前を返します。たとえば、Language (LANG) 接続パラメータを使用すれば他の言語を指定できます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、英語のロケールの場合、`Saturday` という値を返します。

```
SELECT DAYNAME( '1987/05/02' );
```

ドイツ語のロケールの場合、`Samstag` という値を返します。イタリア語のロケールの場合、`sabato` という値を返します。複数のロケールがサポートされています。

1.3.2.59 DAYS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の日数を返します。

構文

0000-02-29 と TIMESTAMP 値の間の日数を返します。

```
DAYS( timestamp-expression )
```

2 つの TIMESTAMP 値の間の日数を返します。

```
DAYS( timestamp-expression, timestamp-expression )
```

TIMESTAMP に時刻を追加します。

```
DAYS( timestamp-expression, integer-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

integer-expression

`timestamp-expression` に追加する日数。`integer-expression` が負の場合、適切な日数が `timestamp-expression` から減算されます。integer-expression を指定する場合は、`timestamp-expression` を TIME、DATE または TIMESTAMP として明示的にキャストしてください。`timestamp-expression` が TIME 値の場合、現在の日付が使用されます。

戻り値

タイムスタンプに時刻を追加する場合は TIMESTAMP。それ以外の場合は INTEGER。

備考

DAYS 関数の結果は、その引数によって異なります。DAYS 関数では、引数内の時間、分、および秒は無視されます。

0000-02-29 以降の日数を返します。

1 つの `timestamp-expression` を DAYS 関数に渡すと、0000-02-29 から `timestamp-expression` までの日数を INTEGER として返します。

注記

0000-02-29 は実際の日付を指すための値ではありません。DAYS 関数で使用されるデフォルトの日付です。

2 つの TIMESTAMP 値の間の日数を返します。

2 つの TIMESTAMP 値を DAYS 関数に渡すと、その間の日数を整数で返します。

DATEDIFF 関数を使用して 2 つの日付間の間隔を取得することもできます。

TIMESTAMP に時間を追加します。

TIMESTAMP 値と整数を DAYS 関数に渡すと、整数値を `timestamp-expression` 引数に加算した TIMESTAMP 結果を返します。

DATEADD 関数を使用して、TIMESTAMP に日付要素を追加することもできます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、整数 729889 を返します。

```
SELECT DAYS( '1998-07-13 06:07:12' );
```

次の文は、整数値 -366 を返します。このことは、2 番目の DATE 値が最初の日付より 366 日前の日付であることを示します。2 つ目の例 (DATEDIFF) の使用をお奨めします。

```
SELECT DAYS( '1998-07-13 06:07:12',
             '1997-07-12 10:07:12' );
```

```
SELECT DATEDIFF( day,
   '1998-07-13 06:07:12',
   '1997-07-12 10:07:12' );
```

次の文は、TIMESTAMP 値 1999/07/14 0:00:00.000 を返します。2 つ目の例 (DATEADD) の使用をお奨めします。

```
SELECT DAYS( CAST('1998-07-13' AS DATE ), 366 );
```

```
SELECT DATEADD( day, 366, '1998-07-13' );
```

関連情報

DATEDIFF 関数 [日付と時刻] [305 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]

CAST 関数 [データ型変換] [264 ページ]

1.3.2.60 DB_EXTENDED_PROPERTY 関数 [システム]

指定したプロパティの値を返します。オプションでプロパティ固有の文字列パラメータを指定できます。

構文

```
DB_EXTENDED_PROPERTY(
{ property-id | property-name }
[, property-specific-argument
[, database-id | database-name ] ]
)
```

パラメータ

property-id

問い合わせるデータベースプロパティ ID。

property-name

問い合わせるデータベースプロパティ名。

property-specific-argument

次のデータベースプロパティを使用すると、以下に示すように、プロパティに関する特定の情報を返すための追加の引数を指定できます。

CatalogCollation, Collation, NcharCollation

これらのプロパティを問い合わせるとき、次の値を `property-specific-argument` として指定すると、照合に固有の情報が返されます。

AccentSensitivity

照合でのアクセント記号の区別の設定を返します。たとえば、次の文は、NCHAR 照合でのアクセント記号の区別の設定を返します。

```
SELECT DB_EXTENDED_PROPERTY( 'NcharCollation', 'AccentSensitivity');
```

可能な戻り値:[NULL]、[Ignore]、[Respect]、[French]。

CaseSensitivity

照合での大文字と小文字の区別の設定を返します。可能な戻り値:[NULL]、[Ignore]、[Respect]、[UpperFirst]、[LowerFirst]。

PunctuationSensitivity

照合での句読表記の区別の設定を返します。可能な戻り値:[NULL]、[Ignore]、[Primary]、[Quaternary]。

Properties

照合に指定されるすべての適合化オプションが含まれる文字列を返します。

Specification

照合で使用される完全照合指定が含まれる文字列を返します。

CharSet

規格の名称を指定して、指定した規格におけるデフォルトの CHAR 文字セットのラベルを取得します。指定可能な値は次のとおりです。ASE、IANA、MIME、JAVA、WINDOWS、UTR22、IBM、ICU。規格が指定されない場合、IANA がデフォルトとして使用されます。ただし、データベース接続が TDS によって作成された場合、デフォルトは ASE です。

DBFileFragments

DB 領域の名前またはファイル ID を指定して、ファイルのフラグメント数を取得します。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

DriveBus

(Microsoft Windows の場合のみ) DB 領域の名前またはファイル ID を指定して、その DB 領域があるドライブの設定を取得します。DriveBus は、IOCTL_STORAGE_QUERY_PROPERTY 呼び出しから BusType を返します。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

DriveModel

(Microsoft Windows の場合のみ) DB 領域の名前またはファイル ID を指定して、その DB 領域があるドライブのモデルを取得します。DriveModel は、IOCTL_STORAGE_QUERY_PROPERTY 呼び出しから、VendorId、ProductId、および ProductRevision を連結した文字列を返します。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

DriveType

DB 領域の名前またはファイル ID を指定して、ドライブのタイプを取得します。返される値は次のいずれかです。[CD]、[FIXED]、[RAMDISK]、[REMOTE]、[REMOVABLE]、[UNKNOWN]。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

File

DB 領域の名前またはファイル ID を指定して、データベースルートファイルのファイル名を、パスを含めて取得します。*'translog'* を指定してトランザクションログファイルのパスとファイル名を取得し、*'translogmirror'* を指定してトランザクションログミラーファイルのパスとファイル名を取得します。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

FileSize

DB 領域の名前またはファイル ID を指定して、指定したファイルのサイズをページ数で取得します。*'temporary'* を指定してテンポラリ DB 領域のサイズを返す、*'translog'* を指定してトランザクションログファイルのサイズを返す、および *'translogmirror'* を指定してトランザクションログファイルミラーのサイズを返すこともできます。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

FreePages

DB 領域の名前またはファイル ID を指定して、空きページ数を取得します。temporary を指定して、テンポラリ DB 領域の空きページ数を返すこともできます。または、translog を指定して、トランザクションログファイルの空きページ

数を返すこともできます。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

IOParallelism

DB 領域の名前またはファイル ID を指定して、その DB 領域がサポートする同時 I/O 操作の推定回数を取得します。DB 領域の名前またはファイル ID を指定しなかった場合、システム DB 領域が使用されます。指定した DB 領域の名前や ID が、接続中のデータベースに存在しない場合、関数は NULL を返します。

MirrorServerState

サーバ名を指定して、ミラーサーバの接続ステータスを判断します。CONNECTED、DISCONNECTED、INCOMING ONLY、OUTGOING ONLY、または NULL を返します。値は、データベースがミラーリングされていない場合は *NULL*、このサーバから指定したサーバへの接続と、指定したサーバからこのサーバへの接続がある場合は *Connected*、このサーバと指定したサーバ間の接続がない場合は *Disconnected*、指定したサーバからこのサーバへの接続のみがある場合は *Incoming*、このサーバから指定したサーバへの接続のみがある場合は *Outgoing* になります。

MirrorState

サーバ名を指定して、ミラーサーバの同期ステータスを判断します。SYNCHRONIZING、SYNCHRONIZED、または NULL を返します。ミラーサーバが接続されていない場合、すべてのプライマリサーバのログページを読み込んでいない場合、または同期モードが非同期の場合、値は *Synchronizing* になります。ミラーサーバが接続され、プライマリサーバ上でコミットされたすべての変更が反映されている場合、値は *Synchronized* になります。データベースがミラーリングされていない場合、この値は *NULL* になります。

NcharCharSet

規格の名称を指定して、その規格におけるデフォルトの NCHAR 文字セットエンコードラベルを取得します。指定可能な値は次のとおりです。ASE、IANA、MIME、JAVA、WINDOWS、UTR22、IBM、ICU。規格が指定されない場合、IANA がデフォルトとして使用されます。ただし、データベース接続が TDS によって作成された場合、デフォルトは ASE です。

NextScheduleTime

イベント名を指定して、次にスケジュールされている実行時刻を取得します。

database-id

DB_ID 関数が返すデータベース ID 番号。通常、データベース名が使用されます。

database-name

DB_NAME 関数から返されるデータベース名。

戻り値

VARCHAR

備考

DB_EXTENDED_PROPERTY 関数は DB_PROPERTY に似ています。異なる点は、DB_EXTENDED_PROPERTY 関数が `property-specific-argument` 文字列パラメータをオプションで指定できることです。`property-specific-argument` の解釈は、最初の引数で指定されたプロパティ ID またはプロパティ名によって異なります。

3 番目の引数を省略すると、現在のデータベースが使用されます。

テーブル名やプロシージャ名などのカタログ文字列を比較する場合、データベースサーバでは CHAR 照合が使用されます。UCA 照合の場合、カタログ照合は CHAR 照合と同じですが、適合化は大文字と小文字およびアクセント記号が区別せず、句読表記はレベル 1 でソートされるようになります。これまでの照合の場合、カタログ照合は CHAR 照合と同じですが、適合化は大文字と小文字を区別しないようになります。カタログ照合で使用される適合化は明示的に指定できませんが、Specification プロパティを問い合わせて、カタログ文字列を比較するためにデータベースサーバで使用される完全照合指定を取得することができます。Specification プロパティの問い合わせは、CHAR 照合とカタログ照合の違いを利用する必要がある場合に便利です。たとえば、句読表記を区別しない CHAR 照合を使用し、アップグレードスクリプトを実行するとします。このスクリプトでは、my_procedure というプロシージャを定義し、myprocedure という古いバージョンを削除しようとします。CHAR 照合を使用すると my_procedure は myprocedure と等価であるため、次の文では目的の結果を得られません。

```
CREATE PROCEDURE my_procedure( ) ...;
IF EXISTS ( SELECT * FROM SYS.SYSPROCEDURE WHERE proc_name = 'myprocedure' )
THEN DROP PROCEDURE myprocedure
END IF;
```

その代わり、次の文を実行すると目的の結果を得られます。

```
CREATE PROCEDURE my_procedure( ) ...;
IF EXISTS ( SELECT * FROM SYS.SYSPROCEDURE
   WHERE COMPARE( proc_name, 'myprocedure',
DB_EXTENDED_PROPERTY( 'CatalogCollation', 'Specification' ) ) = 0 )
THEN DROP PROCEDURE myprocedure
END IF;
```

権限

現在のデータベースに対してこの関数を実行する場合、権限は必要ありません。この関数を他のデータベースに対して実行するには、SERVER OPERATOR または MONITOR のシステム権限を持っていることが必要です。

無効なパラメータ値を指定した場合、または必要とされるシステム権限のどれかがない場合は、NULL が戻ります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在のデータベースの場所を返します。

```
SELECT DB_EXTENDED_PROPERTY( 'File' );
```

次の文は、システムの DB 領域のファイルサイズをページ単位で返します。

```
SELECT DB_EXTENDED_PROPERTY( 'FileSize' );
```

次の文は、トランザクションログのファイルサイズをページ単位で返します。

```
SELECT DB_EXTENDED_PROPERTY( 'FileSize', 'translog' );
```

たとえば、次の文は、NCHAR 照合での大文字と小文字の区別の設定を返します。

```
SELECT DB_EXTENDED_PROPERTY( 'NcharCollation',' CaseSensitivity' );
```

次の文は、データベースの CHAR 照合に指定された適合化オプションを返します。

```
SELECT DB_EXTENDED_PROPERTY ( 'Collation', 'Properties' );
```

次の文は、データベースの NCHAR 照合の完全照合指定を返します。

```
SELECT DB_EXTENDED_PROPERTY( 'NcharCollation', 'Specification' );
```

次の文は、ミラーサーバ Test の接続ステータスを返します。

```
SELECT DB_EXTENDED_PROPERTY( 'MirrorServerState', 'Test' );
```

次の文は、ミラーサーバ Test の同期ステータスを返します。

```
SELECT DB_EXTENDED_PROPERTY( 'MirrorState', 'Test' );
```

関連情報

DB_ID 関数 [システム] [321 ページ]
DB_NAME 関数 [システム] [322 ページ]
CONNECTION_PROPERTY 関数 [システム] [281 ページ]
CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]

1.3.2.61 DB_ID 関数 [システム]

データベース ID 番号を返します。

構文

```
DB_ID( [ database-name ] )
```

パラメータ

database-name

データベース名を含む文字列。database-name を指定しない場合、現在のデータベースの ID 番号が返されます。

戻り値

INT

備考

なし

権限

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

サーバ上の唯一のデータベースとしてのサンプルデータベースに対して次の文を実行すると、値 0 を返します。

```
SELECT DB_ID( 'demo' );
```

実行中の唯一のデータベースに対して実行すると、次の文は値 0 を返します。

```
SELECT DB_ID( );
```

1.3.2.62 DB_NAME 関数 [システム]

指定した ID 番号を持つデータベースの名前を返します。

構文

```
DB_NAME( [ database-id ] )
```

パラメータ

database-id

データベースの ID。`database-id` には、数値式を指定してください。

戻り値

VARCHAR

備考

データベース ID を指定しない場合は、現在のデータベース名が返されます。

権限

現在のデータベースに対してこの関数を実行する場合、権限は必要ありません。この関数を他のデータベースに対して実行するには、SERVER OPERATOR または MONITOR のシステム権限を持っていることが必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

サーバ上の唯一のデータベースとしてのサンプルデータベースに対して次の文を実行すると、データベース名 demo を返します。

```
SELECT DB_NAME( 0 );
```

関連情報

sa_db_list システムプロシージャ [1481 ページ]
NEXT_DATABASE 関数 [システム] [451 ページ]

1.3.2.63 DB_PROPERTY 関数 [システム]

指定されたデータベースのプロパティの値を返します。

構文

```
DB_PROPERTY(
{  property-id |  property-name }
[, database-id | database-name ]
)
```

Ultra Light:

```
DB_PROPERTY( property-name )
```

パラメータ

property-id

データベースプロパティ ID。

property-name

データベースプロパティ名。

database-id

DB_ID 関数が返すデータベース ID 番号。通常、データベース名が使用されます。

database-name

DB_NAME 関数から返されるデータベース名。

戻り値

VARCHAR、LONG VARCHAR

備考

文字列を返します。

2 番目の引数を省略すると、現在のデータベースが使用されます。

Ultra Light: Ultra Light でオプションを設定するには、SET OPTION 文、またはコンポーネントの API に固有のデータベースオプション設定メソッドを使用します。

権限

現在のデータベースに対してこの関数を実行する場合、権限は必要ありません。この関数を他のデータベースに対して実行するには、SERVER OPERATOR または MONITOR のシステム権限を持っていることが必要です。

無効なパラメータ値を指定した場合、または必要とされるシステム権限のどれかがない場合は、NULL が戻ります。

Ultra Light: これらの権限は Ultra Light には適用されません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在のデータベースのページサイズをバイト単位で返します。

```
SELECT DB_PROPERTY( 'PageSize' );
```

Ultra Light: 次の文は、現在のデータベースのページサイズをバイト単位で返します。

```
SELECT DB_PROPERTY( 'page_size');
```

関連情報

DB_ID 関数 [システム] [321 ページ]
DB_NAME 関数 [システム] [322 ページ]
PROPERTY 関数 [システム] [470 ページ]

1.3.2.64 DECOMPRESS 関数 [文字列]

文字列を解凍し、LONG BINARY 値を返します。

構文

```
DECOMPRESS( string-expression [, compression-algorithm-alias] )
```

パラメータ

string-expression

解凍する文字列。この関数にはバイナリ値を渡すこともできます。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

compression-algorithm-alias

解凍に使用するアルゴリズムのエイリアス (文字列)。サポートされている値は zip と gzip です (いずれも同じアルゴリズムに基づいていますが、ヘッダと後書きが異なります)。

zip は幅広くサポートされている圧縮アルゴリズムです。gzip は UNIX の gzip ユーティリティと互換性がありますが、zip アルゴリズムには互換性がありません。

アルゴリズムが指定されない場合、文字列の圧縮に使用されたアルゴリズムの検出が試行されます。指定したアルゴリズムが正しくない場合、または正しいアルゴリズムが検出できなかった場合、文字列は解凍されません。

戻り値

LONG BINARY

備考

この関数を使用して、COMPRESS 関数で圧縮された値を解凍できます。

圧縮されているカラムに格納されている値には DECOMPRESS 関数を使用する必要はありません。圧縮されているカラム値の圧縮と解凍は、データベースサーバが自動的に処理します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、DECOMPRESS 関数を使用して、架空のテーブル TableA の Attachment カラムの値を解凍します。

```
SELECT DECOMPRESS ( Attachment, 'gzip' )
FROM TableA;
```

元の値が LONG VARCHAR などの文字型だった場合、DECOMPRESS はバイナリ値を返すため、CAST を適用して人間が解読できる値を返すようにできます。

```
SELECT CAST ( DECOMPRESS ( Attachment, 'gzip' )
```

```
AS LONG VARCHAR ) FROM TableA;
```

関連情報

文字列関数 [213 ページ]
COMPRESS 関数 [文字列] [275 ページ]

1.3.2.65 DECRYPT 関数 [文字列]

指定されたキーを使用して文字列を復号化し、LONG BINARY 値を返します。

構文

```
DECRYPT( string-expression, key [, algorithm-format [, initialization-vector ] ] )
```

```
algorithm-format :
algorithm [ ( format ) ]
```

```
algorithm :
AES
| AES256
| AES_FIPS
| AES256_FIPS
| RSA
| RSA_FIPS
```

```
format :
FORMAT={ RAW[; padding ] | INTERNAL }
```

```
padding :
PADDING={ PKCS5
| ZEROES
| OAEP
| PKCS1
| ALL
| NONE }
```

パラメータ

string-expression

復号化される文字列。バイナリ値がサポートされます。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

key

string-expression の復号化に必要な暗号化キー (文字列)。AES の場合、暗号化された元の値を取得するには、このキーの値が、string-expression の暗号化に使用された暗号化キーと同じである必要があります。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

キーは、RSA 用の PEM 形式で指定します。

> ⚠ 警告
>
> データベースに強力な暗号化を適用した場合、暗号化キーのコピーを必ず安全な場所に保管してください。暗号化キーがわからなくなった場合は、テクニカルサポートに依頼してもデータにはアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

algorithm-format

このオプション文字列パラメータは、string-expression の暗号化に使用されたアルゴリズム、形式、埋め込みのタイプを指定します。

algorithm

このオプションの文字列パラメータでは、string-expression の暗号化に最初に使用されたアルゴリズムのタイプを指定します。次のいずれかの形式を指定します。

AES

AES アルゴリズムを使用してデータが暗号化されます。

algorithm-format が指定されない場合、デフォルトで AES が使用されます。

AES アルゴリズムの場合、padding は PKCS5、ZEROES、または NONE です。デフォルトの埋め込みは PKCS5 です。

AES256 AES 256 ビットアルゴリズムを使用してデータが暗号化されます。AES256 の場合、padding は PKCS5、ZEROES、および NONE (if FORMAT=RAW) です。

AES_FIPS

データは、FIPS 認定バージョンの AES アルゴリズムを使用して暗号化されます。

データベースサーバの起動時に -fips サーバオプションが使用された場合は、AES_FIPS がデフォルトとして使用されます。AES_FIPS の場合、padding は PKCS5、ZEROES、および NONE (if FORMAT=RAW) です。

AES256_FIPS データは、FIPS 認定バージョンの AES 256 ビットアルゴリズムを使用して暗号化されます。AES256_FIPS の場合、padding は PKCS5、ZEROES、および NONE (if FORMAT=RAW) です。

RSA

RSA アルゴリズムの場合、パブリックキーを使用して暗号化する場合、padding は PKCS1、OAEP、または NONE です。プライベートキーを使用して暗号化をする場合、padding は PKCS1 にする必要があります。デフォルトの埋め込みは PKCS1 です。

RSA アルゴリズムが指定された場合、initialization-vector パラメータは無視され、FORMAT=RAW も無視されます。

メッセージがパブリックキーにより暗号化されている場合、プライベートキーにより復号化する必要があります。暗号化と復号化に同じキーを使用すると、PADDING=NONE でなければ失敗します。ただし、PADDING=NONE が設定されていて間違ったキーがしていされた場合、関数は成功しますが、意味のないデータが返されます。

> **注記**
>
> RSA 暗号化の最大メッセージ長は、キーサイズから 11 バイトを引いた長さ (PKCS1 埋め込みの場合)、およびキーサイズから 42 バイト引いた長さ (OAEP 埋め込みの場合) と等しくなります。PADDING=NONE を指定した場合、メッセージはキーサイズと等しくなる必要があります。AES とは異なり、RSA 暗号化を使用する場合、出力の長さは入力の長さとは同じではありません。

RSA_FIPS データが FIPS 認定バージョンの RSA アルゴリズムを使用して暗号化されることを除き、RSA と同じです。

FORMAT clause

オプションの FORMAT 句を使用して、データの記憶フォーマットを指定します。データが独自の記憶フォーマットに格納された場合は、INTERNAL を指定します。暗号化されたデータがそのまま格納された場合 (つまり、指定したアルゴリズムを復号化可能なソフトウェアにより復号化できます)、RAW を指定します。RAW として格納されたデータの場合、`initialization-vector` パラメータを指定します。

PADDING clause

オプションの PADDING 句を使用して、AES および RSA 暗号化の埋め込みタイプを指定します。AES 暗号化の場合、FORMAT=RAW も指定する必要があります。

PADDING=ALL が使用されている場合を除き、復号化の埋め込みタイプは、暗号化に使用されたタイプと一致する必要があります。

PKCS5

PKCS#5 アルゴリズムを使用してデータが埋め込まれます。暗号化されたデータは、復号化されたデータより 1 ～ 16 バイト長くなります。このオプションは、AES 暗号化にのみ使用されます。これは、AES 暗号化のデフォルトの埋め込みです。

ZEROES

暗号化前、データにゼロ (0) が埋め込まれます。暗号化されたデータは、復号化されたデータより 0 ～ 15 バイト長くなります。暗号化されたデータが復号化されると、結果にもゼロが埋め込まれます。

OAEP 最適な非対称暗号化埋め込みを使用してデータが埋め込まれます。このオプションは、RSA 暗号化にのみ使用されます (RSA または RSA_FIPS)。

PKCS1 PKCS#1 アルゴリズムを使用してデータが埋め込まれます。このオプションは、RSA 暗号化にのみ使用されます (RSA または RSA_FIPS)。このオプションは、RSA 暗号化のデフォルトです (RSA または RSA_FIPS)。

NONE

データへの埋め込みは行われません。入力データは、AES の場合は暗号ブロック長 (16 バイト) の倍数であるか、RSA の場合はキーサイズとまったく同じでなければなりません。

ALL

それらのいずれかが機能するまで、有効な各埋め込みタイプの使用が試みられます。

initialization-vector

`format` が RAW に設定されている場合、`initialization-vector` を指定します。文字列の長さは 16 バイト以内です。16 バイト未満の値には、0 が埋め込まれます。この文字列を NULL に設定することはできません。`format` が INTERNAL に設定されている場合、`initialization-vector` は無視されます。

戻り値

LONG BINARY

備考

DECRYPT 関数は、ENCRYPT 関数を使用して暗号化された `string-expression` を復号化します。データがロー形式でない場合、この関数は入力文字列と同じバイト数の LONG BINARY 値を返します。FORMAT=RAW の場合、返される値の長さは埋め込みフォーマットによって異なります。

AES の場合、`string-expression` を正常に復号化するには、データの暗号化に使用されたのと同じ暗号化キーを使用する必要があります。FORMAT=RAW の場合は、データの暗号化に使用したものと同じ initialization-vector と埋め込みフォーマットを使用します。ロー形式のデータは、データベースサーバ外で複合化することができます。

RSA の場合、FORMAT=RAW が指定されていない場合は、不正な暗号化キーを指定するとエラーが生成されます。FORMAT=RAW を指定して不正な暗号化キーまたは不正な初期化ベクトルを指定すると、複合化が警告なしに失敗します。

警告

データに強力な暗号化を適用した場合、暗号化キーのコピーを必ず安全な場所に保管してください。暗号化キーがわからなくなったら、たとえテクニカルサポートに依頼したとしても、決してデータにアクセスできません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、user_info テーブルにあるユーザのパスワードを復号化します。DECRYPT 関数は値を LONG BINARY データ型に変換して判読不能にするため、パスワードを CHAR データ型に変換するには CAST 関数を使用します。

```
SELECT CAST( DECRYPT( user_pwd, '8U3dkA' ) AS CHAR(100) ) FROM user_info;
```

次の例では、`password` カラムの暗号化バージョンを使用して `secret` カラムが更新されます。データは、暗号化キー 'TheEncryptionKey'、未加工形式の AES 暗号化、初期化ベクトル 'ThisIsTheIV' を使用して暗号化されます。デフォルトの PKCS#5 埋め込みが使用されます。

```
CREATE OR REPLACE TABLE SensitiveData
(
    username char(30), password char(30), secret binary(48)
);
INSERT INTO SensitiveData (username, password)
VALUES
    ('Martin',  'topXsecret1'),
    ('Jasmine', 'my_big_secret'),
    ('Aidan',   'Shortcutsmakelongdelays');
```

```
UPDATE SensitiveData
    SET secret = ENCRYPT( password, 'TheEncryptionKey', 'AES(FORMAT=RAW)',
'ThisIsTheIV' );
```

secret カラムの暗号化されたテキストは、DECRYPT 関数を使用して復号化されます。

```
SELECT
    username,
    password,
    CAST(DECRYPT( secret, 'TheEncryptionKey','AES(FORMAT=RAW;PADDING=PKCS5)',
'ThisIsTheIV' ) AS LONG VARCHAR)
    AS revealed
FROM SensitiveData;
```

関連情報

文字列関数 [213 ページ]
ENCRYPT 関数 [文字列] [336 ページ]
ISENCRYPTED 関数 [システム] [409 ページ]

1.3.2.66 DEGREES 関数 [数値]

数値をラジアンから度数に変換します。

構文

```
DEGREES( numeric-expression )
```

パラメータ

numeric-expression

角度 (ラジアン)。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、numeric-expression で指定される角度を返します。パラメータが NULL 値の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 29.79380534680281 を返します。

```
SELECT DEGREES( 0.52 );
```

1.3.2.67 DENSE_RANK 関数 [ランキング]

パーティション内の値のランクを計算します。同位の値の場合、DENSE_RANK はランキングシーケンス内にギャップを残しません。

構文

```
DENSE_RANK( ) OVER ( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

戻り値

INTEGER

備考

window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。Window 関数として使用する場合、ORDER BY 句を指定する必要があります。また PARTITION BY 句は指定できますが、ROWS 句や RANGE 句はいずれも指定できません。詳細については、WINDOW 句の window-spec 定義を参照してください。

標準

ANSI/ISO SQL 標準

DENSE_RANK は、オプションの ANSI/ISO SQL 言語機能 T612、「Advanced OLAP operations」の一部で構成されています。

SQL 言語機能 F441、「Extended set function support」がサポートされています。これにより、カラム参照ではない任意の式を Window 関数のオペランドで使用できます。

オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set functions」はサポートされていません。このソフトウェアでは、DENSE_RANK 関数を含むクエリブロックからのカラム参照と外部参照の両方を含めるための、集合関数の引数はサポートされていません。

例

次の例は、ユタとニューヨークの従業員の給与ランキングを示す結果セットを返します。結果セットには 19 レコードが返されますが、リスト内の 7 番目と 8 番目の従業員は同一給与で 7 位の同順であるため、18 のランキングのみリストされています。DENSE_RANK 関数はランクにギャップを残さないため、9 番目の従業員を '9' とランキングする代わりに、その従業員は '8' とリストされます。

```
SELECT DepartmentID, Surname, Salary, State,
DENSE_RANK() OVER (ORDER BY Salary DESC) AS SalaryRank
FROM GROUPO.Employees
WHERE State IN ('NY','UT');
```

結果セットは次のとおりです。

DepartmentID	Surname	Salary	State	SalaryRank
100	Shishov	72995.000	UT	1
100	Wang	68400.000	UT	2
100	Cobb	62000.000	UT	3
400	Morris	61300.000	UT	4
300	Davidson	57090.000	NY	5
200	Martel	55700.000	NY	6
400	Blaikie	54900.000	NY	7
100	Diaz	54900.000	UT	7
100	Driscoll	48023.000	UT	8
400	Hildebrand	45829.000	UT	9
100	Whitney	45700.000	NY	10
100	Guevara	42998.000	NY	11
100	Soo	39075.000	NY	12
200	Goggin	37900.000	UT	13
400	Wetherby	35745.000	NY	14
400	Ahmed	34992.000	NY	15

DepartmentID	Surname	Salary	State	SalaryRank
500	Rebeiro	34576.000	UT	16
300	Bigelow	31200.000	UT	17
500	Lynch	24903.000	UT	18

関連情報

WINDOW 句 [1409 ページ]
CUME_DIST 関数 [ランキング] [299 ページ]
PERCENT_RANK 関数 [ランキング] [465 ページ]
RANK 関数 [ランキング] [479 ページ]

1.3.2.68 DIFFERENCE 関数 [文字列]

2 つの文字列式の SOUNDEX 値の差を返します。

構文

```
DIFFERENCE ( string-expression-1, string-expression-2 )
```

パラメータ

string-expression-1

最初の SOUNDEX 引数。

string-expression-2

2 番目の SOUNDEX 引数。

戻り値

SMALLINT

備考

DIFFERENCE 関数は 2 つの文字列の SOUNDEX 値を比較し、値の類似性を評価し、0 ～ 4 の値を返します。最も一致する場合は 4 です。

この関数は常に何らかの値を返します。結果が NULL になるのは引数の 1 つが NULL の場合のみです。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、test と chest という単語の類似点を返します。

```
SELECT DIFFERENCE( 'test', 'chest' );
```

関連情報

文字列関数 [213 ページ]
SOUNDEX 関数 [文字列] [533 ページ]

1.3.2.69 DOW 関数 [日付と時刻]

指定した日付の曜日を表す 1 ～ 7 の数を返します (日曜日 = 1、月曜日 = 2、以下同様)。

構文

```
DOW( date-expression )
```

パラメータ

date-expression

評価される値 (DATE データ型)。

戻り値

SMALLINT

備考

DOW 関数は、first_day_of_week データベースオプションで指定された値の影響を受けません。たとえば、first_day_of_week が月曜日に設定されている場合でも、DOW 関数は月曜日に 2 を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 5 を返します。

```
SELECT DOW( '1998-07-09' );
```

次の文は、値 1 を返します。

```
SELECT DOW( CAST( '2010/05/30 11:33:00.000000+04:00' as TIMESTAMP WITH TIME
ZONE ));
```

次の文は、Employees テーブルに問い合わせ、週の曜日を表す数として従業員の StartDate を返します。

```
SELECT DOW( StartDate ) FROM GROUPO.Employees;
```

1.3.2.70 ENCRYPT 関数 [文字列]

指定された暗号化キーを使用して指定された値を暗号化し、LONG BINARY 値を返します。

構文

```
ENCRYPT( string-expression, key[, algorithm-format [, initialization-vector ] ] )
```

```
algorithm-format :
algorithm [ ( format-clause ) ]
```

```
algorithm :
AES
```

```
| AES256
| AES_FIPS
| AES256_FIPS
| RSA
| RSA_FIPS
```

```
format-clause :
FORMAT={ RAW[; padding-clause ] | INTERNAL }
```

```
padding-clause :
PADDING={ PKCS5
| ZEROES
| OAEP
| PKCS1
| ALL
| NONE }
```

パラメータ

string-expression

復号化される文字列。バイナリ値がサポートされます。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

key

`string-expression` の復号化に必要な暗号化キー (文字列)。AES の場合、暗号化された元の値を取得するには、このキーの値が、`string-expression` の暗号化に使用された暗号化キーと同じである必要があります。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

キーは、RSA 用の PEM 形式で指定します。

> ⚠ 警告
>
> データベースに強力な暗号化を適用した場合、暗号化キーのコピーを必ず安全な場所に保管してください。暗号化キーがわからなくなった場合は、テクニカルサポートに依頼してもデータにはアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

algorithm-format

このオプション文字列パラメータは、`string-expression` を暗号化する場合のアルゴリズム、形式、埋め込みのタイプを指定します。

algorithm

このオプションの文字列パラメータでは、`string-expression` の暗号化に使用されたアルゴリズムのタイプを指定します。次のいずれかの形式を指定します。

AES

AES アルゴリズムを使用してデータが暗号化されます。

`algorithm-format` が指定されない場合、デフォルトで AES が使用されます。

AES アルゴリズムの場合、`padding` は PKCS5、ZEROES、または NONE です。デフォルトの埋め込みは PKCS5 です。

AES256 AES 256 ビットアルゴリズムを使用してデータが暗号化されます。AES256 の場合、`padding` は PKCS5、ZEROES、および NONE (if FORMAT=RAW) です。

AES_FIPS

データは、FIPS 認定バージョンの AES アルゴリズムを使用して暗号化されます。

データベースサーバの起動時に -fips サーバオプションが使用された場合は、AES_FIPS がデフォルトとして使用されます。AES_FIPS の場合、`padding` は PKCS5、ZEROES、および NONE (if FORMAT=RAW) です。

AES256_FIPS データは、FIPS 認定バージョンの AES 256 ビットアルゴリズムを使用して暗号化されます。AES256_FIPS の場合、`padding` は PKCS5、ZEROES、および NONE (if FORMAT=RAW) です。

RSA

RSA アルゴリズムの場合、パブリックキーを使用して暗号化する場合、`padding` は PKCS1、OAEP、または NONE です。プライベートキーを使用して暗号化をする場合、`padding` は PKCS1 にする必要があります。デフォルトの埋め込みは PKCS1 です。

RSA アルゴリズムが指定された場合、`initialization-vector` パラメータは無視され、FORMAT=RAW も無視されます。

メッセージがパブリックキーにより暗号化されている場合、プライベートキーにより復号化する必要があります。暗号化と復号化に同じキーを使用すると、PADDING=NONE でなければ失敗します。ただし、PADDING=NONE が設定されていて間違ったキーがしていされた場合、関数は成功しますが、意味のないデータが返されます。

> **注記**
>
> RSA 暗号化の最大メッセージ長は、キーサイズから 11 バイトを引いた長さ (PKCS1 埋め込みの場合)、およびキーサイズから 42 バイト引いた長さ (OAEP 埋め込みの場合) と等しくなります。PADDING=NONE を指定した場合、メッセージはキーサイズと等しくなる必要があります。AES とは異なり、RSA 暗号化を使用する場合、出力の長さは入力の長さとは同じではありません。

RSA_FIPS データが FIPS 認定バージョンの RSA アルゴリズムを使用して暗号化されることを除き、RSA と同じです。

FORMAT clause

オプションの FORMAT 句を使用して、データの記憶フォーマットを指定します。データが独自の記憶フォーマットに格納された場合は、INTERNAL を指定します。暗号化されたデータがそのまま格納された場合 (つまり、指定したアルゴリズムを復号化可能なソフトウェアにより復号化できます)、RAW を指定します。RAW として格納されたデータの場合、`initialization-vector` パラメータを指定します。

PADDING clause

オプションの PADDING 句を使用して、AES および RSA 暗号化の埋め込みタイプを指定します。AES 暗号化の場合、FORMAT=RAW も指定する必要があります。

PADDING=ALL が使用されている場合を除き、復号化の埋め込みタイプは、暗号化に使用されたタイプと一致する必要があります。

PKCS5

PKCS#5 アルゴリズムを使用してデータが埋め込まれます。暗号化されたデータは、復号化されたデータより 1 ～ 16 バイト長くなります。このオプションは、AES 暗号化にのみ使用されます。これは、AES 暗号化のデフォルトの埋め込みです。

ZEROES

暗号化前、データにゼロ (0) が埋め込まれます。暗号化されたデータは、復号化されたデータより 0 ～ 15 バイト長くなります。暗号化されたデータが復号化されると、結果にもゼロが埋め込まれます。

OAEP 最適な非対称暗号化埋め込みを使用してデータが埋め込まれます。このオプションは、RSA 暗号化にのみ使用されます (RSA または RSA_FIPS)。

PKCS1 PKCS#1 アルゴリズムを使用してデータが埋め込まれます。このオプションは、RSA 暗号化にのみ使用されます (RSA または RSA_FIPS)。このオプションは、RSA 暗号化のデフォルトです (RSA または RSA_FIPS)。

NONE

データへの埋め込みは行われません。入力データは、AES の場合は暗号ブロック長 (16 バイト) の倍数であるか、RSA の場合はキーサイズとまったく同じでなければなりません。

initialization-vector

`format` が RAW に設定されている場合、`initialization-vector` を指定します。文字列の長さは 16 バイト以内です。16 バイト未満の値には、0 が埋め込まれます。この文字列を NULL に設定することはできません。`format` が INTERNAL に設定されている場合、`initialization-vector` は無視されます。

戻り値

LONG BINARY

備考

この関数が返す LONG BINARY 値は、`string-expression` の入力値より最大で 31 バイト長くなります。この関数によって返される値は判読できません。DECRYPT 関数は、ENCRYPT 関数を使用して暗号化された `string-expression` を復号化します。AES の場合、`string-expression` を正常に復号化するには、データの暗号化に使用されたのと同じ暗号化キーとアルゴリズムを使用する必要があります。不正な暗号化キーを指定した場合は、エラーが生成されます。キーを紛失すると、データにアクセスできなくなり、そこからのリカバリも不可能になります。

暗号化された値をテーブルに格納する場合は、文字セット変換がデータに対して実行されないように、カラムを BINARY または LONG BINARY にしてください。

FORMAT=RAW を指定した場合、データは未加工の暗号化を使用して暗号化されます。暗号化キー、初期化ベクトル、オプションで埋め込みフォーマットを指定します。データを復号化するときに、同じ値が必要になります。復号化は、データベースサーバの外部で、または DECRYPT 関数を使用して実行されます。

データベースサーバ内でのみデータの暗号化と復号化を行う場合は、初期化ベクトルと埋め込みを指定しなければなりませんが、復号化中に暗号化キーを検証できないため、未加工の暗号化は使用しないでください。

> **注記**
>
> ISENCRYPTED が意味のある結果を返す場合、データは AES/AES256 による ENCRYPT 関数を使用して暗号化されている必要があり、FORMAT=RAW を使用することはできません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のトリガは、user_info テーブルの user_pwd カラムを暗号化します。このカラムにはユーザのパスワードが含まれ、パスワード値が変更されるたびにトリガが起動します。

```
CREATE TRIGGER encrypt_updated_pwd
BEFORE UPDATE OF user_pwd
ON user_info
REFERENCING NEW AS new_pwd
FOR EACH ROW
BEGIN
    SET new_pwd.user_pwd=ENCRYPT( new_pwd.user_pwd, '8U3dkA' );
END;
```

次の例では、`password` カラムの暗号化バージョンを使用して `secret` カラムが更新されます。データは、暗号化キー 'TheEncryptionKey'、未加工形式の AES 暗号化、PKCS#5 埋め込み (デフォルト)、初期化ベクトル 'ThisIsTheIV' を使用して暗号化されます。

```
CREATE OR REPLACE TABLE SensitiveData
(
    username char(30), password char(30), secret binary(48)
);
INSERT INTO SensitiveData (username, password)
VALUES
    ('Martin',  'topXsecret1'),
    ('Jasmine', 'my_big_secret'),
    ('Aidan',   'Shortcutsmakelongdelays');
UPDATE SensitiveData
    SET secret = ENCRYPT( password, 'TheEncryptionKey',
'AES(FORMAT=RAW;PADDING=PKCS5)', 'ThisIsTheIV' );
SELECT *, LENGTH(secret) FROM SensitiveData;
```

関連情報

DECRYPT 関数 [文字列] [327 ページ]
ISENCRYPTED 関数 [システム] [409 ページ]

1.3.2.71 ERROR_LINE 関数 [その他]

TRY...CATCH 文の CATCH ブロックを呼び出したエラーが発生したプロシージャまたはバッチの行番号を返します。

構文

```
ERROR_LINE( )
```

戻り値

UNSIGNED INTEGER は、エラーが発生したストアドプロシージャまたは複合文内の行番号を表しています。

備考

CATCH ブロックの任意の場所から、この関数を呼び出します。この関数は、エラーハンドラ、ネストされた複合文、関数、またはプロシージャ内でエラーが発生した場合に、エラーの現在の情報を報告します。

この関数は、プロシージャの SYSPROCEDURE システムテーブルの proc_defn カラムで見つかった行番号を返します。これらの行番号は、プロシージャの作成に使用されるソース定義の行番号とは異なる可能性があります。

標準

ANSI/ISO SQL 標準

標準になし。

例

プロシージャ u1.proc1 の行 15 でゼロによる除算のエラーにより呼び出されたハンドラ内で実行された場合、`SELECT ERROR_LINE( ), ERROR_MESSAGE( ), ERROR_PROCEDURE( )` 文が次のような結果を返します。

```
15, 'Division by zero',  '"u1"."proc1"'
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]
SYSPROCEDURE システムビュー [1824 ページ]
SYSPROCEDURE システムビュー [1824 ページ]

1.3.2.72 ERROR_MESSAGE 関数 [その他]

TRY...CATCH 文の CATCH ブロックを呼び出したエラーのメッセージテキストを返します。

構文

```
ERROR_MESSAGE( )
```

戻り値

CATCH ブロックを呼び出したエラーのエラーメッセージを含む VARCHAR です。

備考

CATCH ブロックの任意の場所から、この関数を呼び出します。この関数はエラーハンドラのあらゆる場所でアクティブなエラーメッセージを返します。一方 ERRORMSG 関数は、パラメータなしで呼び出された場合は、エラーハンドラの最初の文で呼び出されたときのエラーメッセージのみを返します。

エラーメッセージのパラメータは、実際の値に置き換えられます。

標準

ANSI/ISO SQL 標準

標準になし。

例

プロシージャ u1.proc1 の行 15 でゼロによる除算のエラーにより呼び出されたハンドラ内で実行された場合、`SELECT ERROR_LINE(), ERROR_MESSAGE(), ERROR_PROCEDURE()` 文が次の結果を返します。

```
15, 'Division by zero',  '"u1"."proc1"'
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]

ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]

1.3.2.73 ERROR_PROCEDURE 関数 [その他]

例外ハンドラが実行される契機となったエラー内のプロシージャ名を返します。

構文

```
ERROR_PROCEDURE( )
```

戻り値

例外が発生したプロシージャの修飾名を含む VARCHAR です。複合文がプロシージャ、ファンクション、トリガ、イベントのいずれの一部でもない場合は、プロシージャの所有者と名前ではなく、バッチのタイプ (`<watcom_batch>` または `<tsql_batch>`) が返されます。

備考

ERROR_PROCEDURE は、例外ハンドラ内のどこでも呼び出すことができます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のバッチは、0 による除算の例外処理を示しています。

```
BEGIN
   DECLARE divTest INT;
   SET divTest = 1 / 0;
```

```
   SELECT 'No error';
   EXCEPTION WHEN OTHERS THEN
      SELECT 'Exception: SQLCODE = ' || ERROR_SQLCODE() ||
                     ', SQLSTATE = ' || ERROR_SQLSTATE() ||
                     ', PROCEDURE = ' || ERROR_PROCEDURE();
END;
```

このバッチの実行結果は、次のとおりです。

```
Exception: SQLCODE = -628, SQLSTATE = 22012, PROCEDURE = <watcom_batch>
```

次のプロシージャも、0 による除算の例外処理を示しています。

```
CREATE OR REPLACE PROCEDURE ExceptionDemo()
BEGIN
   DECLARE divTest INT;
   SET divTest = 1 / 0;
   SELECT 'No error';
   EXCEPTION WHEN OTHERS THEN
      SELECT 'Exception: SQLCODE = ' || ERROR_SQLCODE() ||
                     ', SQLSTATE = ' || ERROR_SQLSTATE() ||
                     ', PROCEDURE = ' || ERROR_PROCEDURE();
END;
CALL ExceptionDemo()
```

このプロシージャの実行結果は、次のとおりです。

```
Exception: SQLCODE = -628, SQLSTATE = 22012, PROCEDURE = "DBA"."ExceptionDemo"
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]

1.3.2.74 ERROR_SQLCODE 関数 [その他]

エラーハンドラを呼び出したエラーの SQLCODE を返します。

構文

ERROR_SQLCODE( )

戻り値

エラーハンドラを呼び出したエラーの SQLCODE の値を持つ SIGNED INTEGER です。

備考

この関数は、エラーハンドラ内のどこでも呼び出すことができます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のバッチは、0 による除算の例外処理を示しています。

```
BEGIN
   DECLARE divTest INT;
   SET divTest = 1 / 0;
   SELECT 'No error';
   EXCEPTION WHEN OTHERS THEN
      SELECT 'Exception: SQLCODE = ' || ERROR_SQLCODE() ||
                     ', SQLSTATE = ' || ERROR_SQLSTATE() ||
                     ', PROCEDURE = ' || ERROR_PROCEDURE();
END;
```

このバッチの実行結果は、次のとおりです。

```
Exception: SQLCODE = -628, SQLSTATE = 22012, PROCEDURE = <watcom_batch>
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]

sa_stack_trace システムプロシージャ [1633 ページ]

1.3.2.75 ERROR_SQLSTATE 関数 [その他]

エラーハンドラを呼び出したエラーの SQLSTATE を返します。

構文

```
ERROR_SQLSTATE( )
```

戻り値

エラーハンドラを呼び出したエラーの SQLSTATE を表す CHAR(5) です。

備考

この関数は、エラーハンドラ内のどこでも呼び出すことができます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のバッチは、0 による除算の例外処理を示しています。

```
BEGIN
   DECLARE divTest INT;
   SET divTest = 1 / 0;
   SELECT 'No error';
   EXCEPTION WHEN OTHERS THEN
      SELECT 'Exception: SQLCODE = ' || ERROR_SQLCODE() ||
                     ', SQLSTATE = ' || ERROR_SQLSTATE() ||
                     ', PROCEDURE = ' || ERROR_PROCEDURE();
END;
```

このバッチの実行結果は、次のとおりです。

```
Exception: SQLCODE = -628, SQLSTATE = 22012, PROCEDURE = <watcom_batch>
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]

1.3.2.76 ERROR_STACK_TRACE 関数 [その他]

エラーハンドラを呼び出したエラーの呼び出しシーケンススタックトレースを返します。

構文

ERROR_STACK_TRACE( )

戻り値

エラーハンドラを呼び出したエラーのスタックトレースを表す LONG VARCHAR です。複合文がプロシージャ、ファンクション、トリガ、イベントのいずれの一部でもない場合は、プロシージャ名ではなく、バッチのタイプ (`<watcom_batch>` または `<tsql_batch>`) が返されます。

備考

結果には、改行文字 (¥n) で区切られたテキストの行が含まれます。戻り値の各行にはスタックのプロシージャの修飾名または文のバッチタイプ (該当する場合) が含まれ、文の行数が続きます。返される値の最後の行の末尾は、改行文字ではありません。

この関数は、プロシージャの SYSPROCEDURE システムテーブルの proc_defn カラムで見つかった行番号を返します。これらの行番号は、プロシージャの作成に使用されるソース定義の行番号とは異なる可能性があります。

この関数は、sa_error_stack_trace システムプロシージャと同じ情報を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のプロシージャ (例で行数が追加されているもの) を使用して、エラースタックを取得できます。

```
1 CREATE OR REPLACE PROCEDURE proc1()
2 BEGIN TRY
3    CALL proc2();
4 END TRY
5 BEGIN CATCH
6    SELECT * FROM sa_split_list(ERROR_STACK_TRACE(), '¥n' );
7 END CATCH;
1 CREATE OR REPLACE PROCEDURE proc2()
2 BEGIN
3    CALL proc3();
4 END;
1 CREATE OR REPLACE PROCEDURE proc3()
2 BEGIN
3    DECLARE v INTEGER = 0;
4    SET v = 1 / v;
5 END;
CALL proc1();
```

この呼び出しは、次の結果セットを返します。

```
line_num   row_value
       1   "DBA"."proc1" : 3
       2   "DBA"."proc2" : 3
       3   "DBA"."proc3" : 4
```

RESIGNAL をエラーハンドラで使用して送り返されたエラーが処理された場合、2 番目のハンドラで報告されたエラースタックに元のエラー、RESIGNAL のレコード、送り返された例外のスタックが含まれます。例:

```
CREATE OR REPLACE PROCEDURE proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL proc2();
    END CATCH
END TRY
BEGIN CATCH
    SELECT * FROM sa_split_list(ERROR_STACK_TRACE(), '¥n' );
END CATCH;
CREATE OR REPLACE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;
CREATE OR REPLACE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;
CALL proc1();
```

この呼び出しは、次の結果文字列を返します。

```
line_num   row_value
       1   "DBA"."proc1" : 8
       2   "DBA"."proc2" : 3
       3   RESIGNAL: "DBA"."proc3" : 3
       4   "DBA"."proc1" : 5
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]
SYSPROCEDURE システムビュー [1824 ページ]

1.3.2.77 ERRORMSG 関数 [その他]

現在のエラー、または指定した SQLSTATE 値または SQLCODE 値のエラーメッセージを返します。

構文

```
ERRORMSG( [ sqlstate | sqlcode ] )
```

```
sqlstate: string
```

```
sqlcode: integer
```

パラメータ

sqlstate

この SQLSTATE 値のエラーメッセージが返されます。

sqlcode

この SQLCODE 値のエラーメッセージが返されます。

戻り値

エラーメッセージを含む VARCHAR。

備考

引数を指定しない場合は、現在のステータスのエラーメッセージが返されます。テーブル名やカラム名などが代入されます。

引数を指定した場合は、指定した SQLSTATE または SQLCODE のエラーメッセージが返され、代入は行われません。テーブル名とカラム名は、プレースホルダ (%1) で指定されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SQLCODE -813 のエラーメッセージを返します。

```
SELECT ERRORMSG( -813 );
```

1.3.2.78 ESTIMATE 関数 [その他]

指定したパラメータに基づいてクエリオプティマイザによって計算されたパーセンテージとして、選択性推定を返します。

構文

```
ESTIMATE( column-name [, value [, relation-string ] ] )
```

パラメータ

column-name

推定で使用されるカラム。

value

カラムが比較される値。デフォルトは NULL です。

relation-string

比較に使用される比較演算子。一重引用符で囲んで指定します。このパラメータに使用できる値は、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'!>' です。デフォルトは '=' です。

戻り値

REAL

備考

この関数は、述部 `column-namerelation-stringvalue` の選択性推定を返します。`value` が NULL で、比較演算子が '=' の場合は、述部 `column-name IS NULL` の選択性推定になります。`value` が NULL で、比較演算子が '!=' または '<>' の場合は、述部 `column-name IS NOT NULL` の選択性推定になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、200 より大きいと推定される EmployeeID 値のパーセンテージを返します。正確な値は、データベースで実行したアクションに応じて異なります。

```
SELECT FIRST ESTIMATE( EmployeeID, 200, '>' )
   FROM GROUPO.Employees
   ORDER BY 1;
```

関連情報

INDEX_ESTIMATE 関数 [その他] [403 ページ]
ESTIMATE_SOURCE 関数 [その他] [352 ページ]
EXPERIENCE_ESTIMATE 関数 [その他] [360 ページ]

1.3.2.79 ESTIMATE_SOURCE 関数 [その他]

クエリオプティマイザで使用される選択性推定のソースを提供します。

構文

```
ESTIMATE_SOURCE(
column-name
[, value
[, relation-string ] ]
)
```

パラメータ

column-name

調査されるカラムの名前。

value

カラムが比較される値。デフォルトは NULL です。

relation-string

比較に使用される比較演算子。一重引用符で囲んで指定します。このパラメータに使用できる値は、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'!>' です。デフォルトは '=' です。

戻り値

次のリストは、ESTIMATE_SOURCE から返される選択性推定のソースを示します。

値	選択性推定のソース
Statistics	格納されているカラム統計
カラム	カラム統計に格納されているすべての値の平均
インデックス	インデックス調査
Guess	各タイプの述部に定義された組み込み規則。この値は、使用する関連インデックスが存在しない場合、参照カラムに対して統計が一切収集されていない場合、または述部が複雑である場合にのみ返されます。
Computed	上記以外のソース
Always	指定の述部が常に true の場合に返されます。
Combined	上記の 1 つ以上のソース
Bounded	選択性推定に上限値や下限値が配置されている場合に返されます。

備考

この関数は、述部 `column-namerelation-stringvalue` の選択性推定のソースを返します。`value` が NULL で、比較演算子が '=' の場合は、述部 `column-name IS NULL` の選択性ソースになります。`value` が NULL で、比較演算子が '!=' または '<>' の場合は、述部 `column-name IS NOT NULL` の選択性ソースになります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、EmployeeID カラムの最初の値が 200 より大きいかどうかを評価するための選択性ソースインデックスを返します。インデックスが返されることは、クエリオプティマイザがインデックスを使用して選択性を推測したことを意味します。

```
SELECT FIRST ESTIMATE_SOURCE( EmployeeID, 200, '>' )
   FROM GROUPO.Employees
   ORDER BY 1;
```

関連情報

ESTIMATE 関数 [その他] [350 ページ]
INDEX_ESTIMATE 関数 [その他] [403 ページ]

1.3.2.80 EVENT_CONDITION 関数 [システム]

イベントハンドラがトリガされる条件を指定します。

構文

```
EVENT_CONDITION( condition-name )
```

パラメータ

condition-name

イベントをトリガする条件。指定可能な値はデータベースにあらかじめ設定されています。大文字と小文字は区別されません。各条件は、特定のイベントタイプにのみ有効です。次の表は、各条件とそれらが有効となるイベントを示します。

条件名	単位	イベント	コメント
DBFreePercent	該当なし	DBDiskSpace	データベースの残り空きディスク領域の割合
DBFreeSpace	MB	DBDiskSpace	データベースの使用可能な空きディスク領域 (MB 単位)
DBSize	MB	GrowDB	データベースのサイズ (MB 単位)
ErrorNumber	該当なし	RAISERROR	エラーコード番号
IdleTime	秒	ServerIdle	サーバーがアイドル状態の時間 (秒単位)
Interval	秒	All	ハンドラが最後に実行してから経過した時間
LogFreePercent	該当なし	LogDiskSpace	ログの残り空きディスク領域の割合
LogFreeSpace	MB	LogDiskSpace	ログの使用可能な空きディスク領域 (MB 単位)
LogSize	MB	GrowLog	ログのサイズ (MB 単位)
RemainingValues	integer	GlobalAutoincrement	残りの値の数
TempFreePercent	該当なし	TempDiskSpace	テンポラリファイル領域の空きディスク領域の割合
TempFreeSpace	MB	TempDiskSpace	テンポラリファイル領域の使用可能な空きディスク領域 (MB 単位)
TempSize	MB	GrowTemp	テンポラリファイル領域のサイズ (MB 単位)

戻り値

INT

備考

イベントから呼び出されていない場合、EVENT_CONDITION 関数は NULL を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のイベント定義は、EVENT_CONDITION 関数を使用します。

```
CREATE EVENT LogNotifier
TYPE LogDiskSpace
WHERE event_condition( 'LogFreePercent' ) < 50
HANDLER
BEGIN
   MESSAGE 'LogNotifier message'
END;
```

関連情報

CREATE EVENT 文 [805 ページ]

1.3.2.81 EVENT_CONDITION_NAME 関数 [システム]

EVENT_CONDITION に指定可能なパラメータをリストします。

構文

```
EVENT_CONDITION_NAME( integer )
```

パラメータ

integer

0 以上の整数。

戻り値

VARCHAR

備考

関数が NULL を返すまで整数をループすると、EVENT_CONDITION_NAME 関数を使用して EVENT_CONDITION 関数のすべての引数のリストを取得できます。

イベントから呼び出されていない場合、EVENT_CONDITION_NAME 関数は NULL を返します。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE EVENT 文 [805 ページ]

1.3.2.82 EVENT_PARAMETER 関数 [システム]

イベントハンドラのためのコンテキスト情報を提供します。

構文

```
EVENT_PARAMETER( context-name )
```

```
context-name :
AppInfo
| ConnectionID
| DisconnectReason
| EventName
| Executions
| MirrorServerName
| NumActive
| ScheduleName
| SQLCODE
| TableName
| User
| condition-name
```

パラメータ

context-name

あらかじめ設定されている文字列の 1 つ。文字列は引用符で囲み、大文字と小文字を区別しません。次の情報を受け渡します。

AppInfo

イベントをトリガさせた接続の AppInfo 接続プロパティの値。イベントのコンテキスト外のプロパティの値を参照するには、次の文を使用します。

```
SELECT CONNECTION_PROPERTY( 'AppInfo' );
```

AppInfo 文字列には、Embedded SQL、ODBC、OLE DB、ADO.NET、SQL Anywhere JDBC ドライバの各接続に対するクライアント接続コンピュータ名とアプリケーション名が含まれています。

ConnectionID

イベントをトリガさせた接続の接続 ID。

DisconnectReason

接続が終了した理由を示す文字列。このパラメータは、Disconnect イベントに対してのみ有効です。表示される結果は、次のとおりです。

abnormal

データベース接続を切断する前にクライアントアプリケーションが異常終了したか、またはクライアントコンピュータとサーバコンピュータの間で通信エラーが発生したことによって、切断が行われました。

connect failed

接続の試行に失敗しました。

drop connection

DROP CONNECTION 文が実行されました。

from client

クライアントアプリケーションが接続を切断しました。

inactive

-ti サーバオプションで指定された期間に、要求が受信されませんでした。

liveness

-tl サーバオプションで指定された期間に、活性パケットが受信されませんでした。

EventName

トリガされたイベント名。

Executions

イベントハンドラの実行回数。

MirrorServerName

データベースミラーリングシステムのプライマリサーバとの接続を失ったミラーサーバまたは監視サーバの名前。

NumActive

イベントハンドラのアクティブインタンス数。これは、一定時間に 1 つのイベントハンドラで 1 つのインスタンスだけを実行させるように制限する場合に利用できます。

ScheduleName

イベントを起動させたスケジュール名。イベントが、TRIGGER EVENT を使用して手動で起動された場合、またはシステムイベントとして起動された場合、結果は空の文字列になります。スケジュールが作成されたときにスケジュール名が明示的に割り当てられなかった場合は、イベントの名前になります。

SQLCODE

接続の失敗時に発生したエラーの SQLCODE。このパラメータは、ConnectFailed イベントに対してのみ有効です。

TableName

RemainingValues で使用するテーブル名。

User

イベントをトリガさせたユーザのユーザ ID。

さらに、EVENT_PARAMETER 関数からは、EVENT_CONDITION 関数の有効なすべての `condition-name` 引数にアクセスできます。

次の表は、システムイベントタイプに対して有効な context-name 値を示します。

システムイベントタイプ	Context-name 値
BackupEnd	AppInfo、ConnectionID、EventName、Executions、NumActive、User
Connect	AppInfo、ConnectionID、EventName、Executions、NumActive、User
ConnectFailed	AppInfo、EventName、Executions、NumActive、SQLCODE、User
"Disconnect"	AppInfo、ConnectionID、EventName、Executions、NumActive、User
GlobalAutoincrement	ConnectionID、EventName、Executions、NumActive、TableName、User
"RAISERROR"	AppInfo、ConnectionID、EventName、Executions、NumActive、User
User events	AppInfo、ConnectionID、EventName、Executions、NumActive、User

戻り値

VARCHAR

備考

イベントに渡される値の最大サイズは、サーバの最大ページサイズ (-gp サーバオプション) によって制限されます。このサイズより長い値は、最大ページサイズを下回るサイズにトランケートされます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、イベントに文字列パラメータを渡す方法を示します。イベントは、トリガされた時刻をデータベースサーバメッセージウィンドウに表示します。

```
CREATE EVENT ev_PassedParameter
HANDLER
BEGIN
  MESSAGE 'ev_PassedParameter - was triggered at ' || event_parameter( 'time' );
END;
TRIGGER EVENT ev_PassedParameter( "Time"=string(current timestamp ) );
```

次の例では、代わりに => パラメータ構文を使用します。

```
CREATE EVENT ev_PassedParameter
HANDLER
BEGIN
  MESSAGE 'ev_PassedParameter - was triggered at ' ||
event_parameter( 'what_time' );
END;
TRIGGER EVENT ev_PassedParameter( what_time => string( current timestamp ) );
```

関連情報

EVENT_CONDITION 関数 [システム] [353 ページ]
CREATE EVENT 文 [805 ページ]
TRIGGER EVENT 文 [1365 ページ]

1.3.2.83 EXP 関数 [数値]

自然対数の底 e を、与えられた引数でべき乗した結果を返します。

構文

```
EXP( numeric-expression )
```

パラメータ

numeric-expression

指数。

戻り値

DOUBLE

備考

EXP 関数は、自然対数の底 e を `numeric-expression` で指定された値でべき乗した結果を返します。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

EXP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 3269017.3724721107 を返します。

```
SELECT EXP( 15 );
```

1.3.2.84 EXPERIENCE_ESTIMATE 関数 [その他]

指定したパラメータに基づいてクエリオプティマイザによって計算されたパーセンテージとして、選択性推定を返します。

構文

```
EXPERIENCE_ESTIMATE(
column-name
[, value
[, relation-string ] ]
)
```

パラメータ

column-name

調査されるカラムの名前。

value

カラムが比較される値。

relation-string

比較に使用される比較演算子。このパラメータに使用できる値は、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'=='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'!>' です。デフォルトは '=' です。

戻り値

REAL

備考

`value` が NULL の場合、比較演算子 = と != はそれぞれ IS NULL 条件と IS NOT NULL 条件として解釈されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、90.3262405396 を返します。

```
SELECT DISTINCT EXPERIENCE_ESTIMATE( EmployeeID, 200, '>' )
FROM GROUPO.Employees;
```

関連情報

ESTIMATE 関数 [その他] [350 ページ]
INDEX_ESTIMATE 関数 [その他] [403 ページ]
ESTIMATE_SOURCE 関数 [その他] [352 ページ]

1.3.2.85 EXPLANATION 関数 [その他]

SQL 文の最適化方法をプレーンテキスト文字列で返します。

構文

```
EXPLANATION(
string-expression
[ , cursor-type ]
[, update-status ]
)
```

Ultra Light:

```
EXPLANATION( string-expression )
```

パラメータ

string-expression

SQL 文。通常は SELECT 文ですが、UPDATE 文、MERGE 文、または DELETE 文も指定できます。

cursor-type

カーソルタイプ。文字列として表現されます。使用できる値は、asensitive、insensitive、sensitive、または keyset-driven です。`cursor-type` が指定されない場合、デフォルトで asensitive が使用されます。

update-status

次のいずれかの値を受け入れる文字列パラメータ。これらの値は、指定されたカーソルをオプティマイザがどのように処理するかを示します。

値	説明
READ-ONLY	このカーソルは読み込み専用です。
READ-WRITE (デフォルト)	このカーソルは読み込みや書き込みが可能です。
FOR UPDATE	このカーソルは読み込みや書き込みが可能です。READ-WRITE と同じです。

戻り値

LONG VARCHAR

備考

文字列として返されるクエリの実行プランです。

GRAPHICAL_PLAN 関数では、文の最適化方法に影響を及ぼすシステムプロパティなど、アクセスプランに関するさまざまな情報が提供されます。

この情報は、追加するインデックスの決定や、パフォーマンスを向上するためのデータベース構造の決定に役立ちます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。

```
SELECT EXPLANATION( 'SELECT * FROM Departments WHERE DepartmentID > 100' );
```

次の文は、'SELECT * FROM Departments WHERE....' クエリに対する INSENSITIVE カーソルのプランを短いテキスト形式で表した文字列を返します。

```
SELECT EXPLANATION( 'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100',
   'insensitive', 'read-only' );
```

関連情報

PLAN 関数 [その他] [467 ページ]
GRAPHICAL_PLAN 関数 [その他] [376 ページ]

1.3.2.86 EXPRTYPE 関数 [その他]

式のデータ型を識別する文字列を返します。

構文

```
EXPRTYPE( string-expression, integer-expression )
```

パラメータ

string-expression

SELECT 文。式のデータ型が問い合わされる場合、その式は SELECT リストに表示されます。文字列が有効な SELECT 文でない場合は、NULL を返します。

integer-expression

対象となる式の SELECT リストでの位置。SELECT リスト内の最初の項目の番号は 1 です。integer-expression 値が SELECT リスト項目に対応していない場合、NULL が返されます。

戻り値

LONG VARCHAR

備考

ユーザ定義のドメインでは、EXPRTYPE はドメイン名ではなく基本となるデータ型の説明を返します。たとえば、次のように指定して mydomain というドメインを作成し、mydomain を使用してテーブルカラムを定義するとします。

```
CREATE DOMAIN mydomain CHAR(20);
CREATE TABLE mytable( colA mydomain, colB DATETIME );
```

`SELECT EXPRTYPE( 'SELECT * FROM mytable', 1 )` を実行すると、返されるデータ型は char(20) になり、mydomain ではありません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SQL Anywhere サンプルデータベースに対して実行した場合に、`smallint` を返します。

```
SELECT EXPRTYPE( 'SELECT LineID FROM SalesOrderItems', 1 );
```

関連情報

SQL データ型 [124 ページ]

sa_describe_query システムプロシージャ [1491 ページ]

1.3.2.87 EXTENDED_PROPERTY function [システム]

指定されたデータベースのプロパティの値を返します。オプションでプロパティ固有の文字列パラメータを指定できます。

構文

```
EXTENDED_PROPERTY(
{ property-id | property-name }
[, property-specific-argument]
)
```

パラメータ

property-id

データベースサーバプロパティのプロパティ番号を表す整数。この番号は、PROPERTY_NUMBER 関数で調べることができます。プロパティのセットを繰り返し処理する場合は、`property-id` がよく使用されます。

property-name

問い合わせるデータベースサーバプロパティ名。

HasSecureFeatureKey

データベースサーバにリスト内の機能をすべてロック解除するセキュリティ保護済み機能キーがあるかどうかを調べるため、リストを指定します。`property-specific-argument` が NULL の場合は NULL が返されます。それ以外の場合は Yes または No が返されます。

HasSecuredFeature

指定された機能のいずれかがグローバルサーバレベルで保護されているかどうかを調べるため、機能のリストを指定します。`property-specific-argument` が NULL の場合は NULL が返されます。それ以外の場合は Yes または No が返されます。

property-specific-argument

次のデータベースサーバプロパティを使用すると、以下に示すように、プロパティに関する特定の情報を返すための追加の引数を指定できます。

HasSecureFeatureKey `feature-list`

`feature-list` 内のすべての機能をロック解除するセキュリティ機能キーがあるかどうかを調べるための、機能のリストを指定します。

HasSecuredFeature `feature-list`

これらの機能の 1 つ以上が保護されているかどうかを調べるための機能のリストを指定します。

戻り値

VARCHAR、LONG VARCHAR

備考

EXTENDED_PROPERTY 関数は PROPERTY に似ています。異なる点は、EXTENDED_PROPERTY 関数が `property-specific-argument` 文字列パラメータをオプションで指定できることです。`property-specific-argument` の解釈は、最初の引数で指定されたプロパティ ID またはプロパティ名によって異なります。

権限

この関数を実行するために必要な権限は何もありません。

無効なパラメータ値を指定した場合、NULL が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

現在の接続でキーを必要とせずに xp_cmdshell システムプロシージャを使用できるかどうかを調べるには、次の文を実行します。

```
SELECT EXTENDED_PROPERTY( 'HasSecuredFeature', 'cmdshell' );
```

CMDSHELL 機能がセキュリティ保護済みでない場合、この文は No を返します。CMDSHELL 機能がセキュリティ保護済みで、この機能にアクセスするためにセキュリティ保護済み機能キーが必要とされる場合は、この文は Yes を返します。

CMDSHELL 機能へのアクセスを可能にするセキュリティ保護済み機能キーがあるかどうかを調べるには、次の文を実行します。

```
SELECT EXTENDED_PROPERTY( 'HasSecureFeatureKey', 'cmdshell' );
```

セキュリティ保護済み機能キーが使用できない場合は、この文は No を返します。この機能へのアクセスを可能にするセキュリティ保護済み機能キーがある場合は、この文は Yes を返します。

現在の接続でキーを必要とせずに BACKUP 文と RESTORE 文を実行できるかどうかを調べるには、次の文を実行します。

```
SELECT EXTENDED_PROPERTY( 'HasSecuredFeature', 'backup,restore' );
```

いずれの機能もセキュリティ保護済みでない場合は、この文は No を返します。いずれかの機能がセキュリティ保護済みの場合、この文は Yes を返します。両方の機能がセキュリティ保護済みであるかどうかを調べるには、次の文を実行して両方の結果をチェックします。

```
SELECT EXTENDED_PROPERTY( 'HasSecuredFeature', 'backup' ) AS [backup],
    EXTENDED_PROPERTY( 'HasSecuredFeature', 'restore' ) AS [restore];
```

現在の接続に BACKUP 文と RESTORE 文の実行を許可するセキュリティ保護済み機能キーがあるかどうかを調べるには、次の文を実行します。

```
SELECT EXTENDED_PROPERTY( 'HasSecureFeatureKey', 'backup,restore' );
```

両方の機能を有効にするキーがある場合、この文は Yes を返します。少なくとも 1 つのキーで有効にできない機能がある場合には、この文は No を返します。

関連情報

PROPERTY 関数 [システム] [470 ページ]

1.3.2.88 EXTRACT 関数 [日付と時刻]

TIMESTAMP 式から日付要素を返します。

構文

```
EXTRACT( date-part FROM timestamp-expression )
```

パラメータ

date-part

名前が返される日付の単位。有効な値は、YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、TIMEZONE_HOUR、および TIMEZONE_MINUTE です。

timestamp-expression

TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。

戻り値

date-part が SECOND の場合、戻り値は NUMERIC 値となり、1 秒に満たない時間が含まれます (マイクロ秒までの精度)。date-part がこれ以外の値の場合、関数の戻り値は INTEGER です。

備考

EXTRACT 関数は DATEPART 関数と似ていますが、まったく同じではありません。EXTRACT 関数は、日付要素の一部しか受け付けません。また、date-part が SECOND の場合、2 つの関数の戻り値は異なります。

標準

ANSI/ISO SQL 標準

コア機能。

例

次の文は、56.789000 を返します。

```
SELECT EXTRACT( SECOND FROM '2015-07-01 12:34:56.789000' );
```

次の文は、30 を返します。

```
SELECT EXTRACT( TIMEZONE_MINUTE FROM '2015-07-01 12:34:56.789000 +05:30' );
```

次の文は、5 を返します。

```
SELECT EXTRACT( TIMEZONE_HOUR FROM '2015-07-01 12:34:56.789000 +05:30' );
```

関連情報

日付の単位の指定 [206 ページ]
SET 文 [T-SQL] [1327 ページ]
MICROSECOND 関数 [日付と時刻] [433 ページ]
DATEPART 関数 [日付と時刻] [310 ページ]
MILLISECOND 関数 [日付と時刻] [435 ページ]

1.3.2.89 FIRST_VALUE 関数 [集合]

ウィンドウの最初のローの値を返します。

構文

```
FIRST_VALUE( [ ALL ] expression [ { RESPECT | IGNORE } NULLS ] )
OVER ( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

expression

評価する式です。たとえば、カラム名です。

戻り値

ウィンドウの最初のローの値のデータ型です。

備考

FIRST_VALUE 関数を使用すると、セルフジョインを使用せずに、(何らかの順序による) 最初の値を選択できます。最初の値を計算の基準として使用する場合、この関数が役立ちます。

FIRST_VALUE 関数は、ウィンドウの最初のレコードを取得します。次に、最初のレコードに対して `expression` が比較され、結果が返されます。

IGNORE NULLS を指定すると、`expression` にある最初の NULL 以外の値が返されます。RESPECT NULLS (デフォルト) を指定すると、最初の値が、それが NULL であってもなくても返されます。

FIRST_VALUE 関数は、その他の大部分の集合関数とは異なり、ウィンドウ指定を行った場合にのみ使用できます。

`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。詳細については、WINDOW 句の `window-spec` 定義を参照してください。

標準

ANSI/ISO SQL 標準

標準になし。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、「Extended set function support」がサポートされています。これにより、カラム参照ではない任意の式を Window 関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set function」がサポートされていません。さらに、このソフトウェアでは、FIRST_VALUE 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の例は、各従業員の給料と、同じ部署内で最近採用された従業員の給料との関係を、パーセンテージで返します。

```
SELECT DepartmentID, EmployeeID,
       100 * Salary / ( FIRST_VALUE( Salary ) OVER (
                           PARTITION BY DepartmentID  ORDER BY StartDate DESC ) )
           AS percentage
    FROM GROUPO.Employees;
```

DepartmentID	EmployeeID	percentage
500	1658	100
500	1615	110.4284624
500	1570	138.8427097
500	1013	109.5851905
500	921	167.4497049
500	868	113.2393688
500	750	137.7344095
500	703	222.8679276
500	191	119.6642975
400	1751	100
400	1740	99.705647
400	1684	130.969936
400	1643	83.9734797
400	1607	175.1828989
400	1576	197.0164609
...	...	...

従業員 1658 は部署 500 の最初のローに示されていることから、従業員 1658 がこの部署で最近採用された従業員であることがわかります。パーセンテージは 100% に設定されています。部署 500 の残りの従業員のパーセンテージは、従業員 1658 に対して相対的に比較されて算出されます。たとえば、従業員 1570 は、従業員 1658 の給料の約 139% に該当する給料を受け取っています。

同じ部署内にいるその他の従業員が、最近採用された従業員と同じ額の給料を受け取っている場合は、その従業員のパーセンテージも 100 になります。

関連情報

WINDOW 句 [1409 ページ]
LAST_VALUE 関数 [集合] [412 ページ]

1.3.2.90 FLOOR 関数 [数値]

指定された値以下の、最大の整数値を返します。

構文

```
FLOOR( numeric-expression )
```

パラメータ

numeric-expression

トランケートされる値。通常は、位取りがゼロ以外の固定数値型または適切な数値型 (DOUBLE、REAL、または FLOAT) です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準

ANSI/ISO SQL 標準

FLOOR 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、123 の Floor 値を返します。

```
SELECT FLOOR (123);
```

次の文は、123 の Floor 値を返します。

```
SELECT FLOOR (123.45);
```

次の文は、-124 の Floor 値を返します。

```
SELECT FLOOR (-123.45);
```

関連情報

CEILING 関数 [数値] [267 ページ]

1.3.2.91 GET_BIT 関数 [ビット配列]

ビット配列の指定したビットの値 (1 または 0) を返します。

構文

```
GET_BIT( bit-expression, position )
```

パラメータ

bit-expression

ビットを含むビット配列。

position

ステータスを返すビットの位置。

戻り値

BIT

備考

配列の位置は左側からカウントします。初期値は 1 です。

`position` は配列の長さを超える場合、0 (false) が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 1 を返します。

```
SELECT GET_BIT( '00110011' , 4 );
```

次の文は、値 0 を返します。

```
SELECT GET_BIT( '00110011' , 5 );
```

関連情報

ビット処理演算子 [31 ページ]
SET_BIT 関数 [ビット配列] [522 ページ]
SET_BITS 関数 [集合] [524 ページ]
sa_get_bits システムプロシージャ [1510 ページ]

1.3.2.92 GET_IDENTITY 関数 [その他]

AUTOINCREMENT カラムに値を割り当てます。AUTOINCREMENT を使用して数を生成する代わりに、この関数を使用できます。

構文

```
GET_IDENTITY( table_name [, number_to_allocate ] )
```

パラメータ

table_name

テーブルの名前を指定する文字列。オプションで所有者名を含みます。

number_to_allocate

予約する値の数。デフォルトは 1 です。

戻り値

UNSIGNED BIGINT

備考

最も効率的な ID の生成方法は AUTOINCREMENT または GLOBAL AUTOINCREMENT を使用する方法ですが、この関数は代替手段として提供されています。この関数は、テーブルに AUTOINCREMENT カラムが定義されていることを前提としています。テーブルの AUTOINCREMENT カラムの生成に使用可能な次の値を返し、他の接続がデフォルトでその値を使用しないように値を予約します。

テーブルが見つからない場合はエラーを返し、テーブルに AUTOINCREMENT カラムが存在しない場合は NULL を返します。複数の AUTOINCREMENT カラムがある場合は、最初に検出されたものが使用されます。

`number_to_allocate` は予約する値の数です。`number_to_allocate` に 1 より大きい値を指定すると、残りの値も予約されます。次回の割り当てでは、現在の数に `number_to_allocate` の値を加算した数が使用されます。これによって、アプリケーションは GET_IDENTITY 関数を頻繁に実行せずにすみます。`number_to_allocate` が 0 の場合は、値が予約されずに、次に使用可能な値が返されます。

GET_IDENTITY 関数の実行後は COMMIT が不要のため、ローの挿入に使用するのと同じ接続を使用して呼び出すことができます。例に示すように、いくつかのテーブルの ID 値が必要な場合は、GET_IDENTITY 関数の呼び出しを複数持つ単一の SELECT を使用して取得できます。

GET_IDENTITY 関数は、非決定的関数です。以降、GET_IDENTITY 関数を呼び出すと異なる値が返される可能性があります。オプティマイザは、GET_IDENTITY 関数の結果をキャッシュしません。

権限

`table-name` に対する INSERT 権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Customers テーブルの AUTOINCREMENT カラム (ID) に対して次に使用できる数値を返します。返された値とその後の 9 個の値が予約されます。

```
SELECT GET_IDENTITY( 'GROUPO.Customers', 10 );
```

関連情報

CREATE TABLE 文 [952 ページ]
ALTER TABLE 文 [705 ページ]
NUMBER 関数 [その他] [461 ページ]

1.3.2.93 GETDATE 関数 [日付と時刻]

現在の年、月、日、時、分、秒、秒以下を返します。

構文

```
GETDATE( )
```

戻り値

TIMESTAMP

備考

精度はシステムクロックの精度によって制限されます。

GETDATE 関数が返す情報は、NOW 関数と CURRENT TIMESTAMP 特別値が返す情報と同じです。

注記

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこの関数の結果が計算されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、システムの日付と時刻を返します。

```
SELECT GETDATE( );
```

関連情報

CURRENT DATE 特別値 [85 ページ]
CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
SQL 文の式 [32 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.3.2.94 GRAPHICAL_PLAN 関数 [その他]

SQL 文のプランの最適化方法を、XML フォーマットの文字列で返します。

構文

```
GRAPHICAL_PLAN(
string-expression
[, statistics-level
```

```
[, cursor-type
[, update-status ] ] ] )
```

パラメータ

string-expression

SQL 文。通常は SELECT 文ですが、UPDATE 文または DELETE 文も指定できます。

statistics-level

整数。`Statistics-level` には、次のいずれかの値を指定します。

値	説明
0	オプティマイザの推定のみ (デフォルト)
2	ノード統計値を含む詳細な統計情報
3	詳細な統計情報

cursor-type

カーソルタイプ。文字列として表現されます。使用できる値は、asensitive、insensitive、sensitive、または keyset-driven です。`cursor-type` が指定されない場合、デフォルトで asensitive が使用されます。

update-status

次のいずれかの値を受け入れる文字列パラメータ。これらの値は、指定されたカーソルをオプティマイザがどのように処理するかを示します。

値	説明
READ-ONLY	このカーソルは読み込み専用です。
READ-WRITE (デフォルト)	このカーソルは読み込みや書き込みが可能です。
FOR UPDATE	このカーソルは読み込みや書き込みが可能です。READ-WRITE とまったく同じです。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の Interactive SQL の例は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。プランは `plan.saplan` ファイルに保存されます。これは、Interactive SQL を使用して読み込むことができます。

```
SELECT GRAPHICAL_PLAN(  'SELECT * FROM GROUPO.Departments WHERE DepartmentID >
100' );
OUTPUT TO 'plan.saplan' FORMAT TEXT QUOTE '' HEXADECIMAL ASIS;
```

次の文は、`SELECT * FROM Departments WHERE GROUPO.DepartmentID > 100` クエリに対するキーセット駆動型の更新可能なカーソルのグラフィカルなプランを含む文字列を返します。また、オプティマイザが使用した推定統計情報に加え、実際の実行の統計情報の注釈がサーバによってプランに付けられます。

```
SELECT GRAPHICAL_PLAN(
   'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100',
   2,
   'keyset-driven', 'for update' );
```

関連情報

PLAN 関数 [その他] [467 ページ]
EXPLANATION 関数 [その他] [362 ページ]

1.3.2.95 GREATER 関数 [その他]

2 つのパラメータ値のうち、より大きい値を返します。

構文

```
GREATER( expression-1, expression-2 )
```

パラメータ

expression-1

比較される最初のパラメータ値。

expression-2

比較される 2 番目のパラメータ値。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、比較に使用したデータ型で結果を返します。データベースサーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

備考

パラメータが等しい場合は、最初のパラメータを返します。

TABLE REF 型として定義された値は、この関数ではサポートされません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 10 を返します。

```
SELECT GREATER( 10, 5 ) FROM SYS.DUMMY;
```

関連情報

LESSER 関数 [その他] [419 ページ]

1.3.2.96 GROUPING 関数 [集合]

GROUP BY 演算の結果セット内のカラムが NULL である場合、その理由が小計ローの一部であるためか、または基本データによるためかを識別します。

構文

```
GROUPING( group-by-expression )
```

パラメータ

group-by-expression

GROUP BY 句を使用するクエリの結果セット内において、グループ化カラムとして表示される式。この関数を使用して、ROLLUP 演算または CUBE 演算で設定された結果セットに追加される小計ローを調べることができます。

戻り値

1

小計ローの一部なので、`group-by-expression` が NULL であることを示します。カラムは、そのローのプレフィクスカラムではありません。

0

小計ローのプレフィクスカラムなので、`group-by-expression` が NULL であることを示します。

標準

ANSI/ISO SQL 標準

GROUPING 関数は、オプションの ANSI/ISO SQL 言語機能 T431、「Extended grouping capabilities」の一部です。

関連情報

SELECT 文 [1291 ページ]

1.3.2.97 HASH 関数 [文字列]

指定された値をハッシュ形式で返します。

構文

```
HASH( expression[, algorithm ] )
```

パラメータ

expression

ハッシュされる文字列。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

algorithm

ハッシュに使用するアルゴリズム。使用できる値は、次のとおりです。CRC32、MD5、SHA1、SHA1_FIPS、SHA256、SHA256_FIPS。デフォルトでは、MD5 アルゴリズムが使用されます。FIPS 認定のアルゴリズムには別のライセンスが必要です。

戻り値

使用するアルゴリズムごとに返される型を次に示します。

- CRC32 は 16 進数文字列を返します。16 進数文字列を 32 ビット整数に変換するには、HEXTOINT 関数を使用します。
- MD5 は VARCHAR(32) を返します
- SHA1 は VARCHAR(40) を返します
- SHA1_FIPS は VARCHAR(40) を返します
- SHA256 は VARCHAR(64) を返します
- SHA256_FIPS は VARCHAR(64) を返します

備考

ハッシュを使用すると、値は、関数に渡されたそれぞれの値に対してユニークなバイトシーケンスに変換されます。

VARBIT カラムをハッシュ化するときは、まずカラムを文字列にキャストして、可変長 VARBIT の値がさまざまな結果にハッシュ化されるようにすることができます。

データベースサーバを -fips オプションを使用して起動すると、使用されるアルゴリズムや動作が次のように異なる場合があります。

- SHA1 を指定する場合、SHA1_FIPS が使用されます。
- SHA256 を指定する場合、SHA256_FIPS が使用されます。
- MD5 を指定する場合、エラーが返されます。
- FIPS 認定ではない CRC32 アルゴリズムは暗号化アルゴリズムとして認識されないため、FIPS で使用できます。

> ⚠ 警告
>
> すべてのアルゴリズムは一方向のハッシュです。ハッシュから元の文字列を再作成することはできません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ユーザ ID やパスワードなど、アプリケーションのユーザに関する情報を格納するテーブル user_info を作成します。テーブルにはローが 1 つ挿入されます。パスワードは、HASH 関数と SHA256 アルゴリズムを使用してハッシュされます。この方法でハッシュ済みパスワードを格納する方法は、クリアテキストでパスワードを格納せず、パスワードの比較を必要とする外部アプリケーションがある場合に有効です。

```
CREATE TABLE user_info (
  employee_id   INTEGER NOT NULL PRIMARY KEY,
  user_name CHAR(80),
  user_pwd CHAR(80) );
INSERT INTO user_info
  VALUES ( '1', 's_phillips', HASH( 'mypass', 'SHA256' ) );
```

関連情報

文字列関数 [213 ページ]
HEXTOINT 関数 [データ型変換] [383 ページ]

1.3.2.98 HEXTOBIN 関数 [データ型変換]

16 進文字列と同等の LONG BINARY を返します。

構文

```
HEXTOBIN( hexadecimal-string )
```

パラメータ

hexadecimal-string

バイナリ文字列に変換される文字列です。

戻り値

HEXTOBIN 関数は、LONG BINARY 文字列を返します。入力文字数が奇数の場合は、左側に 0 が埋め込まれます。結果の長さは、入力された文字列の長さの 2 分の 1 になります。入力文字列に 16 進数以外の文字が含まれている場合、エラーが返されます。

備考

HEXTOBIN 関数は、数値、大文字または小文字の A ～ F のみで構成される文字列キーコードまたは変数を受け入れます

BINTOHEX、CAST、CONVERT、HEXTOBIN、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、`0x313233` を含むバイナリ文字列を返します。

```
SELECT HEXTOBIN('313233');
```

関連情報

16 進値との変換 [12 ページ]
BINTOHEX 関数 [データ型変換] [248 ページ]

1.3.2.99 HEXTOINT 関数 [データ型変換]

16 進文字列と同等の 10 進整数を返します。

構文

```
HEXTOINT( hexadecimal-string )
```

パラメータ

hexadecimal-string

整数に変換される文字列。

戻り値

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。

HEXTOINT 関数は、プラットフォームに依存しない SQL INTEGER に相当する 16 進文字列を INT として返します。右から 8 桁目が数値 8 ～ 9 か、大文字または小文字の A ～ F のいずれかであり、その前の桁がすべて大文字または小文字の F の場合は、16 進値が負の整数値になります。次の HEXTOINT は無効な使用例です。これは、引数が符号付き 32 ビット整数で表現できない正の整数値を示しているためです。

```
SELECT HEXTOINT( '0x0080000001' );
```

備考

HEXTOINT 関数は、数値、大文字または小文字の A ～ F のみで構成される文字列キーコードまたは変数を受け入れます (0x プレフィクスが付いている場合、付いていない場合の両方)。次に HEXTOINT の有効な使用例をすべて示します。

```
SELECT HEXTOINT( '0xFFFFFFFF' );
SELECT HEXTOINT( '0x00000100' );
SELECT HEXTOINT( '100' );
SELECT HEXTOINT( '0xffffffff80000001' );
```

HEXTOINT 関数は 0x プレフィクスがあれば削除します。データが 8 桁を超える場合、符号付き 32 ビット整数値として表現できる値を示す必要があります。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 420 を返します。

```
SELECT HEXTOINT( '1A4' );
```

関連情報

16 進値との変換 [12 ページ]
INTTOHEX 関数 [データ型変換] [406 ページ]

1.3.2.100 HOUR 関数 [日付と時刻]

TIMESTAMP 値の時間部分を返します。

構文

```
HOUR( timestamp-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

SMALLINT

備考

TIMESTAMP 式の時間部分である、0 ～ 23 の間の SMALLINT 値が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 21 を返します。

```
SELECT HOUR( '1998-07-09 21:12:13' );
```

1.3.2.101 HOURS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の時間数を返します。

構文

0000-02-29 の真夜中と TIMESTAMP 値の間の時間数を返します。

```
HOURS ( timestamp-expression )
```

2 つの TIMESTAMP 値の間の時間数を返します。

```
HOURS ( timestamp-expression, timestamp-expression )
```

TIMESTAMP に時間を追加します。

```
HOURS ( time-or-timestamp-expression, integer-expression )
```

パラメータ

time-or-timestamp-expression

TIME または TIMESTAMP データ型の値。

timestamp-expression

TIMESTAMP データ型の値。

integer-expression

`time-or-timestamp-expression` に追加する時間数。`integer-expression` が負の場合、適切な時間数が `time-or-timestamp-expression` から減算されます。

戻り値

2 つの `time-or-timestamp-expression` 値の間の時間数を返す場合は INTEGER。

時間を `time-or-timestamp-expression` に追加する場合は TIME または TIMESTAMP。

備考

HOURS 関数の結果は、その引数によって異なります。

0000-02-29 の真夜中以降の時間数を返します。

1 つの `timestamp-expression` を HOURS 関数に渡すと、0000-02-29 の真夜中から `timestamp-expression` までの時間数を INTEGER として返します。

> **注記**
>
> 0000-02-29 は実際の日付を指すための値ではありません。HOURS 関数で使用されるデフォルトの TIMESTAMP 値です。

2 つの TIMESTAMP 値の間の時間数を返します。

2 つの TIMESTAMP 値を HOURS 関数に渡すと、その間の時間数を整数で返します。

TIMESTAMP に時間を追加します。

TIMESTAMP 値と INTEGER 値を HOURS 関数に渡すと、整数の時間数を `time-or-timestamp-expression` 引数に加算した TIMESTAMP 結果を返します。同様に、TIME 値を最初の引数として渡すと、TIME 値を結果として返します。この構文では、最初の引数の暗黙的変換をサポートしていません。最初の引数を DATE、TIME または TIMESTAMP の値に明示的にキャストしてください。最初の引数が DATE の場合、時刻部分には真夜中が使用されます。

これらの計算に DATEDIFF 関数と DATEADD 関数を使用することもできます。

標準

ANSI/ISO SQL 標準

標準になし。

> **例**
>
> 次の文は、値 4 を返します。これは、2 番目の TIMESTAMP 値が、最初のタイムスタンプの 4 時間後であることを示します。2 つ目の例 (DATEDIFF) の使用をお奨めします。

```
SELECT HOURS( '1999-07-13 06:07:12', '1999-07-13 10:07:12' );
SELECT DATEDIFF( hour, '1999-07-13 06:07:12', '1999-07-13 10:07:12' );
```

次の文は、値 17517342 を返します。

```
SELECT HOURS( '1998-07-13 06:07:12' );
```

次の文は、日時 1999/05/13 2:05:07.000 を返します。2 つ目の例 (DATEADD) の使用をお奨めします。

```
SELECT HOURS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );
SELECT DATEADD( hour, 5, '1999-05-12 21:05:07' );
```

関連情報

DATEDIFF 関数 [日付と時刻] [305 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.3.2.102 HTML_DECODE 関数 [その他]

HTML リテラル文字列で表示される特殊文字エンティティを復号化します。

構文

```
HTML_DECODE( string )
```

パラメータ

string

HTML ドキュメントで使用される任意のリテラル文字列。

戻り値

LONG VARCHAR または LONG NVARCHAR。

備考

この関数は、適切な置換を行った後に文字列引数を返します。次の表は、有効な文字エンティティのサンプルを示します。

文字	置換
&quot;	"
&#39;	'
&amp;	&
&lt;	<
&gt;	>
&#x*hexadecimal-number*;	Unicode のコードポイント。16 進数で指定します。たとえば、' は一重引用符を返します。
&#*decimal-number*;	Unicode のコードポイント。10 進数で指定します。たとえば、™ は商標記号を返します。

指定した Unicode のコードポイントが、データベース側文字セットの文字に変換できる場合は、その文字に変換されます。それ以外の場合は、解釈されないまま返されます。

SQL Anywhere は、HTML 4.01 仕様で指定されているすべての文字エンティティ参照をサポートします。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、文字列 The piano was made for 'Steinway & Sons'. を返します。

```
SELECT HTML_DECODE('&lt;p&gt;The piano was made ' ||
  'by &lsquo;Steinway &amp; Sons&rsquo;.&lt;/p&gt;')
```

次の文は、文字列 It cost €85.000,000. を返します。

```
SELECT HTML_DECODE('&lt;p&gt;It cost &euro;85.000,000.&lt;/p&gt;')
```

関連情報

Web サービス関数 [212 ページ]
HTML_ENCODE 関数 [その他] [389 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.103 HTML_ENCODE 関数 [その他]

HTML ドキュメントに挿入する文字列内の特殊文字をエンコードします。

構文

```
HTML_ENCODE( string )
```

パラメータ

string

HTML ドキュメントで使用される任意の文字列。

戻り値

LONG VARCHAR または LONG NVARCHAR。

備考

この関数は、次の一連の置換を行った後に文字列引数を返します。

文字	置換
"	&quot;
'	&#39;
&	&amp;
<	&lt;
>	&gt;
0x20 より小さいコード nn	&#xnn;

この関数は NCHAR の入力または出力をサポートしています。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、文字列 '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"> ' を返します。

```
SELECT HTML_ENCODE('<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">')
```

関連情報

Web サービス関数 [212 ページ]
HTML_DECODE 関数 [その他] [388 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.104 HTTP_BODY 関数 [Web サービス]

HTTP 要求の本文をバイナリ形式で返します。たとえば、POST 要求では、これは未加工の POST データになります。

構文

```
HTTP_BODY( )
```

パラメータ

なし

戻り値

HTTP 要求の本文を含む LONG VARCHAR をバイナリ形式で返し、文字セット変換は行われません。

備考

要求の本文が存在しない場合、または関数が Web サービスから呼び出されない場合は、NULL 値が返されます。

この関数は、PHP 外部環境で便利です。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

Web サービス関数 [212 ページ]
sa_http_php_page システムプロシージャ [1528 ページ]
sa_http_php_page_interpreted システムプロシージャ [1530 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.105 HTTP_DECODE 関数 [Web サービス]

HTTP のコード化された文字列を復号化します。これは URL 復号化とも呼ばれます。

構文

```
HTTP_DECODE( string )
```

パラメータ

string

URL またはコード化された URL の要求本文から取得された任意の文字列。

戻り値

LONG VARCHAR または LONG NVARCHAR

備考

この関数は、%nn という形式 (nn は 16 進値) のすべての文字シーケンスをコード nn の文字で置換した後に、文字列引数を返します。また、すべてのプラス記号 (+) はスペースで置換されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、文字列 http://test.sap.com を返します。

```
SELECT HTTP_DECODE('http%3A%2F%2Ftest.sap.com')
```

関連情報

Web サービス関数 [212 ページ]
HTTP_ENCODE 関数 [Web サービス] [393 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.106 HTTP_ENCODE 関数 [Web サービス]

HTTP で使用するために文字列をコード化します。これは URL コード化とも呼ばれます。

構文

```
HTTP_ENCODE( string )
```

パラメータ

string

HTTP トランスポート用にコード化される任意の文字列。

戻り値

LONG VARCHAR または LONG NVARCHAR

備考

この関数は、次の一連の置換を行った後に文字列引数を返します。また、16 進コードが 20 より小さいか 7E より大きいすべての文字は、%nn (nn は文字コード) で置換されます。

文字	置換
スペース	%20
"	%22
#	%23
%	%25
&	%26

文字	置換
,	%2C
;	%3B
<	%3C
>	%3E
[	%5B
¥	%5C
]	%5D
`	%60
{	%7B
|	%7C
}	%7D
0x20 未満で 0x7f を超える文字コード nn	%nn

この関数は NCHAR の入力または出力をサポートしています。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、文字列 /opt%26id=123%26text='oid:c%09d%20ef' を返します。

```
SELECT HTTP_ENCODE('/opt&id=123&text=''oid:c¥x09d ef''')
```

関連情報

Web サービス関数 [212 ページ]
HTTP_DECODE 関数 [Web サービス] [392 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.107 HTTP_HEADER 関数 [Web サービス]

HTTP 要求ヘッダの値を返します。

構文

```
HTTP_HEADER( header-field-name [, instance ] )
```

パラメータ

header-field-name

HTTP 要求ヘッダフィールドの名前。

instance

取り出すヘッダのインスタンス。複数のヘッダが同じ名前の場合、インスタンスはフィールドインスタンスの番号です。値 0 または NULL は、ヘッダの最も新しいインスタンスを返します。デフォルトは 0 です。

戻り値

LONG VARCHAR。

備考

この関数は、指定された HTTP 要求ヘッダフィールドの値を返します。ヘッダが存在しない場合、または HTTP サービスから呼び出されていない場合は NULL を返します。Web サービスを介して HTTP 要求を処理する場合に使用します。

HTTP Web サービス要求を処理するとき、次のようなヘッダが有用です。

Cookie

要求された URI に関連付けられた cookie 値 (クライアントに格納されている場合)。

Referer

要求された URI へのリンクが指定されたページの URL (`http://documents.sample.com:80/index.html` など)。

ホスト

ユーザが指定するかリソースを参照して元の URI から取得された、要求されたリソースのインターネットホスト名または IP アドレスとポート番号 (`webserver.sample.com:8082` など)。

User-Agent

クライアントアプリケーション名 (`Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0` など)。

Accept-Encoding

クライアントアプリケーションが使用できる応答のエンコーディングリスト (`gzip, deflate` など)。

このようなヘッダの詳細については、HTTP ヘッダフィールドの定義 を参照してください。

次の特別なヘッダでは、クライアント要求の要求行で要素にアクセスできます。

@HttpMethod

処理されている要求の種類を返します。可能な値には DELETE、HEAD、GET、PUT、POST があります。

@HttpURI

HTTP 要求で指定された、要求の完全な URI (`/myservice?&id=-123&version=109&lang=en` など)。

@HttpVersion

要求の HTTP バージョン (`HTTP/1.0` または `HTTP/1.1` など)。

@HttpQueryString

要求された URI が存在する場合に、そのクエリ部分を返します (`id=-123&version=109&lang=en` など)。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、Cookie ヘッダの値の 5 番目のインスタンスが取得されます。

```
SET cookie_header = HTTP_HEADER( 'Cookie', 5 );
```

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、データベースサーバメッセージウィンドウの HTTP 要求ヘッダの名前と値が表示されます。

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name  = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_HEADER( header_name );
    MESSAGE 'HEADER: ', header_name, '=',
            header_value TO CONSOLE;
  END LOOP;
END;
```

関連情報

Web サービス関数 [212 ページ]
NEXT_HTTP_HEADER 関数 [Web サービス] [452 ページ]
sa_set_http_header システムプロシージャ [1619 ページ]
sa_http_header_info システムプロシージャ [1526 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.108 HTTP_RESPONSE_HEADER 関数 [Web サービス]

HTTP 応答ヘッダの値を返します。

構文

```
HTTP_RESPONSE_HEADER( header-field-name [, instance ] )
```

パラメータ

header-field-name

HTTP 応答ヘッダフィールドの名前。

instance

取り出すヘッダのインスタンス。複数のヘッダが同じ名前の場合、インスタンスはフィールドインスタンスの番号です。値 0 または NULL は、ヘッダの最も新しいインスタンスを返します。デフォルトは 0 です。

戻り値

LONG VARCHAR

備考

この関数は、指定された HTTP 応答ヘッダフィールドの値を返します。指定された `header-field-name` のヘッダが存在しない場合、または HTTP サービスから呼び出されていない場合は NULL を返します。

HTTP Web サービス応答を処理するとき、次のようなヘッダが有用です。

Connection

[Connection] フィールドで送信者が特定の接続に必要なオプションを指定できます。SQL Anywhere HTTP サーバ応答では、このオプションは常に「閉じる」です。

Content-Length

[Content-Length] フィールドには、応答本文のサイズが 10 進数のオクテットで表示されます。

Content-Type

[Content-Type] フィールドには、受信者に送信される本文のメディアタイプが表示されます。例: text/xml

Date

[Date] フィールドは、応答が発信された日付と時刻を表します。

Expires

[Expires] フィールドには、それ以降、応答が古くなったとみなされる日付と時刻が表示されます。

Location

[Location] フィールドは、受信者を新しいリソースの要求または識別を完了するロケーションにリダイレクトします。

Server

[Server] フィールドには元のサーバで要求を処理するために使用されるソフトウェアに関する情報が含まれます。SQL Anywhere HTTP サーバ応答では、Web サーバ名がバージョン番号と一緒に表示されます。

Transfer-Encoding

[Transfer-Encoding] フィールドには、送信者と受信者の間でメッセージを安全に転送するためにメッセージ本文に適用される変換があれば、その変換の種類が表示されます。

User-Agent

[User-Agent] フィールドには、要求の発信元のユーザエージェントに関する情報が表示されます。SQL Anywhere HTTP サーバ応答では、Web サーバ名がバージョン番号と一緒に表示されます。

WWW-Authenticate

[WWW-Authenticate] フィールドは、401 (認証されていない) 応答メッセージに含まれます。

このようなヘッダの詳細については、HTTP ヘッダフィールドの定義 を参照してください。

次の特別なヘッダでは、サーバ応答の応答でステータスにアクセスできます。

@HttpStatus

処理された要求のステータスコードを返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、データベースサーバメッセージウィンドウの HTTP 応答ヘッダの名前と値が表示されます。

```
BEGIN
```

```
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_RESPONSE_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_RESPONSE_HEADER( header_name );
    MESSAGE 'RESPONSE HEADER: ', header_name, '=', header_value TO CONSOLE;
  END LOOP;
```

関連情報

Web サービス関数 [212 ページ]
NEXT_HTTP_RESPONSE_HEADER 関数 [Web サービス] [454 ページ]
sa_set_http_header システムプロシージャ [1619 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.109 HTTP_VARIABLE 関数 [Web サービス]

HTTP 変数の値を返します。

構文

```
HTTP_VARIABLE( var-name [ , instance [ , attribute ] ] )
```

パラメータ

var-name

HTTP 変数の名前。

instance

同じ名前の変数が複数ある場合、フィールドインスタンスのインスタンス番号、または最初の NULL。複数選択を許可している SELECT リストで使用すると便利です。

attribute

マルチパート要求では、マルチパート名のヘッダの値を返すヘッダフィールド名を属性で指定できます。

属性を指定しないと、戻り値が % で復号化され、文字セットがデータベースの文字セットに変換されます。このモードでは、UTF % でエンコードされたデータがサポートされています。

属性には、次のいずれかのモードも使用できます。

'@BINARY'

x-www-form-urlencoded バイナリデータ値を返します。このモードは、戻り値が % で復号化されており、変換された文字セットではないことを示します。UTF-8 % エンコードは、このモードではサポートされません。% でエンコードされたデータは単純に等価のバイト表現にデコードされるからです。

'@TRANSPORT'

未加工の HTTP トランスポート形式の値を返し、% でのエンコードが保持されます。

戻り値

LONG VARCHAR。

備考

この関数は、指定された HTTP 変数の値を返します。Web サービス内で HTTP 要求を処理する場合に使用されます。

`var-name` が存在しない場合、戻り値は NULL になります。

Web サービス要求が POST で、変数データが multipart/form-data と通知された場合、HTTP サーバは個々の変数の HTTP ヘッダを受信します。`attribute` パラメータを指定すると、特定の変数の POST 要求から関連する multipart/form-data ヘッダ値が HTTP_VARIABLE 関数から返されます。ファイルを表す変数の場合、Content-Disposition、Content-Type、@BINARY の属性はそれぞれファイル名、メディアタイプ、ファイルの内容を返します。

通常は、クライアント (ブラウザなど) の文字セットとデータベース側文字セットの間で、どの入力データも文字セットの変換が行われます。ただし、`attribute` に @BINARY を指定すると、文字セットの変換または % による復号化を行わずに変数値が返されます。@BINARY は、画像などのデータをクライアントから受信するときに便利です。

この関数は、指定されたインスタンスが存在しない場合、または Web サービスの実行外から関数が呼び出された場合には、NULL を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、サンプル URL で示される HTTP 変数の値が取得されます。

```
-- http://sample.com/demo/ShowDetail?product_id=300&customer_id=101
BEGIN
  DECLARE v_customer_id LONG VARCHAR;
  DECLARE v_product_id LONG VARCHAR;
```

```
  SET v_customer_id = HTTP_VARIABLE( 'customer_id' );
  SET v_product_id = HTTP_VARIABLE( 'product_id' );
  CALL ShowSalesOrderDetail( v_customer_id, v_product_id );
END;
```

HTTP Web サービスによって呼び出されるストアドプロシージャ内で使用された場合、次の文はイメージ変数の Content-Disposition ヘッダと Content-Type ヘッダを要求します。

```
SET v_name = HTTP_VARIABLE( 'image', NULL, 'Content-Disposition' );
SET v_type = HTTP_VARIABLE( 'image', NULL, 'Content-Type' );
```

HTTP Web サービスによって呼び出されるストアドプロシージャ内で使用された場合、次の文は現在の文字セットでのイメージ変数値 (文字セットを変換しない値) を要求します。

```
SET v_image = HTTP_VARIABLE( 'image', NULL, '@BINARY' );
```

関連情報

Web サービス関数 [212 ページ]
NEXT_HTTP_VARIABLE 関数 [Web サービス] [455 ページ]
sa_http_variable_info システムプロシージャ [1532 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.110 IDENTITY 関数 [その他]

クエリの連続した各ローに対して、1 から開始する整数値を生成します。

構文

```
IDENTITY( expression )
```

パラメータ

expression

式。この式は解析されますが、関数の実行時には無視されます。

戻り値

INT

備考

IDENTITY 関数の機能は NUMBER 関数の機能と同じです。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、連番が付けられた従業員リストを返します。

```
SELECT IDENTITY( 10 ), Surname FROM GROUPO.Employees;
```

関連情報

NUMBER 関数 [その他] [461 ページ]

1.3.2.111 IFNULL 関数 [その他]

ある式が NULL かどうかと、値を返すかどうかを評価します。

構文

```
IFNULL( expression-1, expression-2 [ , expression-3 ] )
```

パラメータ

expression-1

評価される式。この値によって、`expression-2` と `expression-3` のどちらを返すかが決まります。

expression-2

`expression-1` が NULL の場合の戻り値。

expression-3

`expression-1` が NULL でない場合の戻り値。

戻り値

返されるデータ型は、expression-2 と expression-3 のデータ型によって異なります。

備考

最初の式が NULL 値の場合は、2 番目の式の値を返します。最初の式が NULL でない場合は、3 番目の式の値を返します。最初の式が NULL ではなく、3 番目の式が存在しない場合は、NULL を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 -66 を返します。

```
SELECT IFNULL( NULL, -66 );
```

次の文は、最初の式が NULL ではなく、3 番目の式が存在しないため、を返します。

```
SELECT IFNULL( -66, -66 );
```

1.3.2.112 INDEX_ESTIMATE 関数 [その他]

指定したパラメータに基づいてクエリオプティマイザによって計算されたパーセンテージとして、インデックスの選択性推定を返します。

構文

```
INDEX_ESTIMATE( column-name [ , value [ , relation-string ] ] )
```

パラメータ

column-name

推定で使用されるカラム。

value

カラムが比較される値。デフォルトは NULL です。

relation-string

比較に使用される比較演算子。一重引用符で囲んで指定します。このパラメータに使用できる値は、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'='、'>'、'<'、'>='、'<='、'<>'、'!='、'!<'、'!>' です。デフォルトは '=' です。

戻り値

REAL

備考

この関数は、述部 `column-namerelation-stringvalue` のインデックスから選択性推定を返します。`value` が NULL で、比較演算子が '=' の場合は、述部 `column-name IS NULL` の選択性推定になります。`value` が NULL で、比較演算子が '!=' または '<>' の場合は、述部 `column-name IS NOT NULL` の選択性推定になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、200 より大きいと推定される EmployeeID 値のパーセンテージを返します。

```
SELECT INDEX_ESTIMATE( EmployeeID, 200, '>' )
FROM GROUPO.Employees;
```

関連情報

ESTIMATE 関数 [その他] [350 ページ]
ESTIMATE_SOURCE 関数 [その他] [352 ページ]
EXPERIENCE_ESTIMATE 関数 [その他] [360 ページ]

1.3.2.113 INSERTSTR 関数 [文字列]

別の文字列の指定された位置に文字列を挿入します。

構文

```
INSERTSTR( integer-expression, string-expression-1, string-expression-2 )
```

パラメータ

integer-expression

文字列の挿入位置。文字列の先頭に挿入する場合は、0 を使用します。

string-expression-1

別の文字列が挿入される文字列。

string-expression-2

挿入する文字列。

戻り値

入力式のデータ型に応じて、LONG BINARY、LONG VARCHAR、または LONG NVARCHAR。

備考

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 `backoffice` を返します。

```
SELECT INSERTSTR( 0, 'office ', 'back' );
```

関連情報

文字列関数 [213 ページ]
STUFF 関数 [文字列] [550 ページ]

1.3.2.114 INTTOHEX 関数 [データ型変換]

整数に対応する 16 進値の文字列を返します。

構文

```
INTTOHEX( integer-expression )
```

パラメータ

integer-expression

16 進に変換される整数。

戻り値

VARCHAR

備考

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 0000009c を返します。

```
SELECT INTTOHEX( 156 );
```

関連情報

16 進値との変換 [12 ページ]
HEXTOINT 関数 [データ型変換] [383 ページ]

1.3.2.115 ISDATE 関数 [データ型変換]

文字列引数を日付に変換できるかどうかをテストします。

構文

```
ISDATE( string )
```

パラメータ

string

文字列が有効な日付を表すかどうかを判断するために分析される文字列。

戻り値

INT

備考

変換できる場合は 1 を返し、できない場合は 0 を返します。引数が NULL の場合は 0 を返します。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、外部ファイルからサンプルデータベースにデータをインポートし、無効な値があるローをエクスポートし、残りのローを永久テーブルにコピーします。

```
CREATE GLOBAL TEMPORARY TABLE MyData(
      person VARCHAR(100),
      birth_date VARCHAR(30),
      height_in_cms VARCHAR(10)
   ) ON COMMIT PRESERVE ROWS;
   LOAD TABLE MyData FROM 'exported.dat';
   UNLOAD
      SELECT * FROM MyData
      WHERE ISDATE( birth_date ) = 0
  OR ISNUMERIC( height_in_cms ) = 0
   TO 'badrows.dat';
   INSERT INTO PermData
      SELECT person, birth_date, height_in_cms
      FROM MyData
      WHERE ISDATE( birth_date ) = 1
  AND ISNUMERIC( height_in_cms ) = 1;
   COMMIT;
   DROP TABLE MyData;
```

関連情報

CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
NOW 関数 [日付と時刻] [458 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.3.2.116 ISENCRYPTED 関数 [システム]

ENCRYPT 関数および指定のキーを使用して文字列が暗号化されているかどうかを判断します。

構文

```
ISENCRYPTED( string, key[, algorithm ] )
```

パラメータ

string

暗号化されているかどうかを判断するために分析される文字列。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

key

`string` の暗号化に使用される暗号化キー。大文字と小文字を区別しないデータベース定義であっても、パラメータの大文字と小文字は区別されます。

algorithm

このオプションのパラメータは、`string` の暗号化に使用されたアルゴリズムを指定します。サポートされているアルゴリズムには、AES、AES256、AES_FIPS、AES256_FIPS などがあります。

FIPS 認定の暗号化をサポートしているプラットフォームでは、`algorithm` に FIPS 認定のアルゴリズムのいずれかを指定できます。

`algorithm` が指定されない場合、デフォルトで AES が使用されます。データベースサーバの起動時に -fips サーバオプションが使用された場合、デフォルトは AES_FIPS になります。

戻り値

INT

備考

ISENCRYPTED は、入力文字列が指定のキーを使用して暗号化されている場合は 1 を返し、それ以外の場合は 0 を返します。

注記

ISENCRYPTED が意味のある結果を返す場合、データは AES/AES256 による ENCRYPT 関数を使用して暗号化されている必要があり、FORMAT=RAW を使用することはできません。

標準

ANSI/ISO SQL 標準

標準になし。

例

入力文字列が暗号化されているため、次の文は値 1 を返します。

```
SELECT ISENCRYPTED( ENCRYPT ('test_string', 'key' ), 'key');
```

関連情報

ENCRYPT 関数 [文字列] [336 ページ]
DECRYPT 関数 [文字列] [327 ページ]

1.3.2.117 ISNULL 関数 [その他]

リストの中から NULL でない最初の式を返します。この関数は COALESCE 関数と同じです。

構文

```
ISNULL( expression, expression [, ...] )
```

パラメータ

expression

NULL かどうかがテストされる式。

2 つ以上の式を関数に渡します。すべての式は比較可能であることが必要です。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、リストから NULL でない最初の式を返します。データベースサーバは、比較が可能な共通のデータ型を見つけることができないと、エラーを返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 -66 を返します。

```
SELECT ISNULL( NULL ,-66, 55, 45, NULL, 16 );
```

関連情報

COALESCE 関数 [その他] [272 ページ]

1.3.2.118 ISNUMERIC 関数 [その他]

文字列の引数が有効な数値であるかどうかを判断します。

構文

```
ISNUMERIC( string )
```

パラメータ

string

文字列が有効な数値を表すかどうかを判断するために分析される文字列。

戻り値

INT

備考

ISNUMERIC は、入力文字列が有効な整数または浮動小数点値であると評価される場合は 1、そうでない場合は 0 を返します。文字列に空白だけが含まれるか、NULL である場合も 0 を返します。

次の例も、ISNUMERIC 関数から 0 を返します。

- 文字 d または D を指数セパレータに使用する値。たとえば 1d2 です。
- NAN、0x12、INF、INFINITY などの特殊な値。
- NULL (例: `SELECT ISNUMERIC( NULL );`)。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、外部ファイルからデータをインポートし、無効な値があるローをエクスポートし、残りのローを永久テーブルにコピーします。この例では、ISNUMERIC 文を使用して height_in_cms 値が数値であることを検証します。

```
CREATE GLOBAL TEMPORARY TABLE MyData(
      person VARCHAR(100),
      birth_date VARCHAR(30),
      height_in_cms VARCHAR(10)
   ) ON COMMIT PRESERVE ROWS;
   LOAD TABLE MyData FROM 'exported.dat';
   UNLOAD
      SELECT *
      FROM MyData
      WHERE ISDATE( birth_date ) = 0
  OR ISNUMERIC( height_in_cms ) = 0
   TO 'badrows.dat';
   INSERT INTO PermData
      SELECT person, birth_date, height_in_cms
      FROM MyData
      WHERE ISDATE( birth_date ) = 1
  AND ISNUMERIC( height_in_cms ) = 1;
   COMMIT;
   DROP TABLE MyData;
```

1.3.2.119 LAST_VALUE 関数 [集合]

ウィンドウの最後のローの値を返します。

構文

```
LAST_VALUE( [ ALL ] expression[ { RESPECT | IGNORE } NULLS ] )
OVER( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

expression

評価する式です。たとえば、カラム名です。

戻り値

引数のデータ型です。

備考

LAST_VALUE 関数を使用すると、セルフジョインを使用せずに、(何らかの順序による) 最後の値を選択できます。最後の値を計算の基準として使用する場合、この関数が役立ちます。

LAST_VALUE 関数は、ORDER BY を実行した後の分割から最後のレコードを取得します。次に、最後のレコードに対して `expression` が比較され、結果が返されます。

IGNORE NULLS を指定すると、`expression` にある最後の NULL 以外の値が返されます。RESPECT NULLS (デフォルト) を指定すると、最後の値が、それが NULL であってもなくても返されます。

LAST_VALUE 関数は、その他の大部分の集合関数とは異なり、ウィンドウ指定を行った場合にのみ使用できます。

`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。WINDOW 句の `window-spec` 定義を参照してください。

標準

ANSI/ISO SQL 標準

標準になし。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、「Extended set function support」がサポートされています。これにより、カラム参照ではない任意の式を Window 関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set function」がサポートされていません。さらに、このソフトウェアでは、LAST_VALUE 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の例は、各従業員の給料と、同じ部署内で最高額の給料を受け取っている従業員の名前を返します。

```
SELECT GivenName + ' ' + Surname AS employee_name,
       Salary, DepartmentID,
       LAST_VALUE( employee_name ) OVER Salary_Window AS highest_paid
FROM GROUPO.Employees
WINDOW Salary_Window AS ( PARTITION BY DepartmentID ORDER BY Salary
                          RANGE BETWEEN UNBOUNDED PRECEDING
                          AND UNBOUNDED FOLLOWING );
```

employee_name	Salary	DepartmentID	highest_paid
Michael Lynch	24903	500	Jose Martinez
Joseph Barker	27290	500	Jose Martinez
Sheila Romero	27500	500	Jose Martinez
Felicia Kuo	28200	500	Jose Martinez
Jeannette Bertrand	29800	500	Jose Martinez
Jane Braun	34300	500	Jose Martinez
Anthony Rebeiro	34576	500	Jose Martinez
Charles Crowley	41700	500	Jose Martinez
Jose Martinez	55500.8	500	Jose Martinez
Doug Charlton	28300	400	Scott Evans
Elizabeth Lambert	29384	400	Scott Evans
Joyce Butterfield	34011	400	Scott Evans
Robert Nielsen	34889	400	Scott Evans
Alex Ahmed	34992	400	Scott Evans
Ruth Wetherby	35745	400	Scott Evans
...	...	...	...

部署 500 で最高額の給料を受け取っている従業員は Jose Martinez で、部署 400 で最高額の給料を受け取っている従業員は Scott Evans です。

関連情報

WINDOW 句 [1409 ページ]
FIRST_VALUE 関数 [集合] [369 ページ]

1.3.2.120 LCASE 関数 [文字列]

文字列中のすべての文字を小文字に変換します。

構文

```
LCASE( string-expression )
```

パラメータ

string-expression

小文字に変換される文字列。

戻り値

データベース照合が UCA の場合、NCHAR データに対して使用されるときは LONG NVARCHAR、CHAR データに対して使用されるときは LONG VARCHAR が返されます。それ以外の場合、データ型は入力データ型と同じになります。

Ultra Light: 返されるデータ型は入力データ型と同じになります。

備考

LCASE 関数は LOWER 関数と同じです。

標準

ANSI/ISO SQL 標準

標準になし。同等の関数 LOWER はコア機能です。

例

次の文は、値 `chocolate` を返します。

```
SELECT LCASE( 'ChoCOlatE' );
```

関連情報

文字列関数 [213 ページ]
LOWER 関数 [文字列] [427 ページ]
UCASE 関数 [文字列] [577 ページ]
UPPER 関数 [文字列] [581 ページ]

1.3.2.121 LEFT 関数 [文字列]

文字列の先頭からいくつかの文字を返します。

構文

```
LEFT( string-expression, integer-expression )
```

パラメータ

string-expression

文字列。

integer-expression

返す文字数。

戻り値

LONG VARCHAR または LONG NVARCHAR

Ultra Light: LONG VARCHAR

備考

文字列にマルチバイト文字があり、適切な照合が使用されている場合、返されるバイト数が指定した文字数よりも大きい場合があります。

引数 string-expression の値より大きな `integer-expression` を指定できます。この場合、全体の値が返されます。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Customers テーブルに含まれる各 Surname 値の最初の 5 文字を返します。

```
SELECT LEFT( Surname, 5) FROM GROUPO.Customers;
```

関連情報

文字列関数 [213 ページ]
RIGHT 関数 [文字列] [508 ページ]

1.3.2.122 LENGTH 関数 [文字列]

指定した文字列の文字数を返します。

構文

```
{ LENGTH | LEN }( string-expression )
```

Ultra Light:

```
LENGTH( string-expression )
```

パラメータ

string-expression

文字列。

戻り値

INT

備考

この関数を使用すると、文字列の長さがわかります。たとえば、`string-expression` にカラム名を指定すると、カラムの値の長さがわかります。

文字列にマルチバイト文字があり、適切な照合が使用されている場合、LENGTH はバイト数ではなく、文字数を返します。文字列が BINARY データ型の場合、LENGTH 関数は BYTE_LENGTH 関数のように動作します。

データ型が CHAR、VARCHAR、LONG VARCHAR、NCHAR の場合、LENGTH 関数と CHAR_LENGTH 関数を使用すると同じ結果が得られます。ただし、LENGTH 関数は BINARY 型およびビット配列データ型に使用します。この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

LENGTH 関数は標準ではありませんが、セマンティックは ANSI/ISO SQL 標準の CHAR_LENGTH 関数と同じです。NCHAR データ型の文字列式での LENGTH の使用は、オプションの ANSI/ISO SQL 言語機能 F421 の一部です。

例

次の文は、値 9 を返します。

```
SELECT LENGTH( 'chocolate' );
```

関連情報

文字列関数 [213 ページ]
BYTE_LENGTH 関数 [文字列] [256 ページ]

1.3.2.123 LESSER 関数 [その他]

2 つのパラメータ値のうち、より小さい値を返します。

構文

```
LESSER( expression-1, expression-2 )
```

パラメータ

expression-1

比較される最初のパラメータ値。

expression-2

比較される 2 番目のパラメータ値。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、比較に使用したデータ型で結果を返します。データベースサーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

備考

パラメータが等しい場合は、最初の値を返します。

TABLE REF 型として定義された値は、この関数ではサポートされません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 5 を返します。

```
SELECT LESSER( 10, 5 ) FROM SYS.DUMMY;
```

関連情報

GREATER 関数 [その他] [378 ページ]

1.3.2.124 LIST 関数 [集合]

グループ内のローごとに値のデリミタ付きリストを返します。

構文

```
LIST( [ALL | DISTINCT ] string-expression [, delimiter-string ] [ ORDER BY order-by-expression [ ASC | DESC ], ... ] )
```

Ultra Light:

```
LIST(
[ DISTINCT ] string-expression
[   , delimiter-string ] )
```

パラメータ

string-expression

文字列式。通常はカラム名です。ALL が指定されていると (デフォルト)、グループ内のローごとに string-expression の値が結果文字列に追加されます。各値は、delimiter-string で区切られます。DISTINCT を指定すると、ユニークな string-expression 値のみが追加されます。

Ultra Light: グループ内のローごとに string-expression の値が結果文字列に追加されます。各値は、delimiter-string で区切られます。

delimiter-string

リスト項目のデリミタ文字列。デフォルト設定はカンマです。NULL 値または空の文字列を指定した場合は、デリミタはありません。delimiter-string は定数でなければなりません。

order-by-expression

関数によって返された項目を並べ替えます。この引数の前にカンマは必要ありません。このため、delimiter-string を指定しない場合、使用が簡単になります。

order-by-expression には整数リテラルを指定できません。ただし、整数リテラルを含む変数を指定できます。

ORDER BY 句に定数が含まれている場合、それらの定数はオプティマイザによって解釈され、同義の ORDER BY 句に置き換えられます。たとえば、オプティマイザは ORDER BY 'a' を ORDER BY 式として解釈します。

クエリブロックに、有効な ORDER BY 句が指定された複数の集合関数が含まれているとき、それらの ORDER BY 句を単一の ORDER BY 句に論理的に結合できる場合は、そのクエリブロックを実行できます。たとえば、次の ORDER BY 句の場合は、

```
ORDER BY expression1, 'a', expression2
```

```
ORDER BY expression1, 'b', expression2, 'c', expression3
```

次の ORDER BY 句として結合されます。

```
ORDER BY expression1, expression2, expression3
```

戻り値

LONG VARCHAR
LONG NVARCHAR

> **注記**
>
> Ultra Light は LONG NVARCHAR を返しません。

備考

LIST 関数は、グループの各ローの、NULL 以外のすべての X の値を (デリミタ付きで) 連結して返します。グループ内に明確な X 値を持ったローが 1 つ以上存在しない場合、LIST(X) は空の文字列を返します。

LIST 関数では、NULL 値と空の文字列は無視されます。

LIST 関数は Window 関数として使用できませんが、Window 関数の入力には使用できます。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、"Extended set function support" がサポートされています。これにより、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、"Mixed column references in set function" がサポートされていません。このソフトウェアでは、LIST 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の文は、値 `487 Kennedy Court,547 School Street` を返します。

```
SELECT LIST( Street ) FROM GROUPO.Employees
WHERE GivenName = 'Thomas';
```

次の文は、従業員 ID をリストします。結果セットの各ローには、部門ごとの従業員 ID のカンマで区切られたリストが入っています。

```
SELECT LIST( EmployeeID )
FROM GROUPO.Employees
GROUP BY DepartmentID;
```

LIST(EmployeeID)
102,105,160,243,247,249,266,278,...
129,195,299,467,641,667,690,856,...
148,390,586,757,879,1293,1336,...
184,207,318,409,591,888,992,1062,...
191,703,750,868,921,1013,1570,...

次の文は、従業員の姓を基準に従業員 ID をソートします。

```
SELECT LIST( EmployeeID ORDER BY Surname ) AS "Sorted IDs"
FROM GROUPO.Employees
GROUP BY DepartmentID;
```

Sorted IDs
1013,191,750,921,868,1658,...
1751,591,1062,1191,992,888,318,...
1336,879,586,390,757,148,1483,...
1039,129,1142,195,667,1162,902,...
160,105,1250,247,266,249,445,...

次の文は、セミコロンで区切られたリストを返します。ORDER BY 句とリストセパレータの位置に注意してください。

```
SELECT LIST( EmployeeID, ';' ORDER BY Surname ) AS "Sorted IDs"
FROM GROUPO.Employees
GROUP BY DepartmentID;
```

Sorted IDs
1013;191;750;921;868;1658;703;...
1751;591;1062;1191;992;888;318;...
1336;879;586;390;757;148;1483;...
1039;129;1142;195;667;1162;902; ...
160;105;1250;247;266;249;445;...

前の文と次の文は区別してください。次の文では、`( Surname, ';' )` を複合ソートキーに使用してソートした従業員 ID のカンマ区切りのリストが返ります。

```
SELECT LIST( EmployeeID ORDER BY Surname, ';' ) AS "Sorted IDs"
FROM GROUPO.Employees
GROUP BY DepartmentID;
```

Ultra Light: 次の文は、Employees テーブルのすべての住所を返します。

```
SELECT LIST( Street ) FROM GROUPO.Employees;
```

関連情報

sa_split_list システムプロシージャ [1630 ページ]
ARRAY_AGG 関数 [集合] [236 ページ]

1.3.2.125 LOCATE 関数 [文字列]

異なる文字列内のある文字列の位置を返します。

構文

```
LOCATE( string-expression-1, string-expression-2 [, integer-expression ] )
```

パラメータ

string-expression-1

検索される文字列。

string-expression-2

検索する文字列。

この文字列は 255 バイトに制限されています。

integer-expression

文字列内の、検索を開始する位置。最初の文字の位置は 1 です。開始オフセットが負の場合は、最初に一致した文字列ではなく、最後に一致した文字列のオフセットを返します。負のオフセットは、文字列の末尾のどれだけの部分が検索から除外されるかを示します。除外されるバイト数は、(-1 * offset) - 1 で計算されます。

戻り値

INT

備考

`integer-expression` を指定すると、文字列のオフセットから検索が開始します。

最初の文字列には長い文字列 (255 バイト以上) を指定できますが、2 番目の文字列は 255 バイトに制限されます。長い文字列を 2 番目の引数として指定すると、LOCATE 関数は NULL 値を返します。文字列が見つからない場合は、0 を返します。長さが 0 の文字列を検索すると、1 が返されます。いずれかの引数が NULL の場合、結果も NULL になります。

マルチバイト文字が使用され、適切な照合が使用されている場合は、開始位置と返される値はバイトで計算した位置と異なる場合があります。

引数 `string-expression-1` および `string-expression-2` がバイナリデータ型の場合、LOCATE 関数の動作は BYTE_LOCATE 関数と同じになります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 8 を返します。

```
SELECT LOCATE(
   'office party this week - rsvp as soon as possible',
   'party',
   2 );
```

次の文は、

```
BEGIN
   DECLARE STR LONG VARCHAR;
   DECLARE POS INT;
   SET str = 'c:¥test¥functions¥locate.sql';
   SET pos = LOCATE( str, '¥', -1 );
   select str, pos,
      SUBSTR( str, 1, pos -1 ) AS path,
      SUBSTR( str, pos +1 ) AS filename;
END;
```

次の出力を返します。

str	pos	path	filename
c:¥test¥functions¥locate.sql	18	c:¥test¥functions	locate.sql

関連情報

文字列関数 [213 ページ]
CHARINDEX 関数 [文字列] [270 ページ]

1.3.2.126 LOG 関数 [数値]

数の自然対数を返します。

構文

```
LOG( numeric-expression )
```

パラメータ

numeric-expression

数値。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

備考

引数は、組み込みの数値データ型の値を返す式です。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 標準では、キーワード LN を使用して自然対数関数が定義されます。自然対数関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、50 の自然対数を返します。

```
SELECT LOG( 50 );
```

関連情報

LOG10 関数 [数値] [426 ページ]

1.3.2.127 LOG10 関数 [数値]

数値の対数 (基数 10) を返します。

構文

```
LOG10( numeric-expression )
```

パラメータ

numeric-expression

数値。

戻り値

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。パラメータが NULL 値の場合、結果は NULL 値になります。

備考

引数は、組み込みの数値データ型の値を返す式です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、50 の底が 10 の対数を返します。

```
SELECT LOG10( 50 );
```

関連情報

LOG 関数 [数値] [425 ページ]

1.3.2.128 LOWER 関数 [文字列]

文字列中のすべての文字を小文字に変換します。

構文

```
LOWER( string-expression )
```

パラメータ

string-expression

小文字に変換される文字列。

戻り値

NCHAR データで使用された場合は LONG NVARCHAR
データベース照合が UCA の場合、CHAR データで使用されるときは LONG VARCHAR
それ以外の場合、データ型は入力データ型と同じになります。
Ultra Light では、入力データ型と同じデータ型が返されます。

備考

LCASE 関数は LOWER 関数と同じです。

標準

ANSI/ISO SQL 標準

コア機能。NCHAR データ型の式での LOWER の使用は、オプションの言語機能 F421 の一部です。

例

次の文は、値 `chocolate` を返します。

```
SELECT LOWER( 'chOCOLate' );
```

関連情報

文字列関数 [213 ページ]
LCASE 関数 [文字列] [415 ページ]
UCASE 関数 [文字列] [577 ページ]
UPPER 関数 [文字列] [581 ページ]

1.3.2.129 LTRIM 関数 [文字列]

文字列の先行ブランクを削除します。

構文

```
LTRIM( string-expression )
```

パラメータ

string-expression

削除する文字列。

戻り値

VARCHAR
NVARCHAR
LONG VARCHAR
LONG NVARCHAR
Ultra Light の場合、VARCHAR または LONG VARCHAR だけが返されます。

備考

結果の実際の長さは、式の長さから、削除する文字数を引いた値です。すべての文字を削除すると、結果は空の文字列になります。

パラメータに NULL を指定できる場合、結果は NULL になります。

パラメータが NULL の場合、結果は NULL 値になります。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

ANSI/ISO SQL 標準で定義される TRIM 仕様 (LEADING と TRAILING) は、それぞれ SQL Anywhere の LTRIM 関数と RTRIM 関数で提供されます。

例

次の文は、すべての先行ブランクを削除して、値 `Test Message` を返します。

```
SELECT LTRIM( '     Test Message' );
```

関連情報

文字列関数 [213 ページ]
RTRIM 関数 [文字列] [515 ページ]
TRIM 関数 [文字列] [572 ページ]

1.3.2.130 MAX 関数 [集合]

各ローグループで見つかった式の最大値を返します。

構文

式:

```
MAX( [ ALL | DISTINCT ] expression )
```

Window 関数:

```
MAX( [ ALL ] expression ) OVER ( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

Ultra Light 式

```
MAX( [ DISTINCT ] expression )
```

パラメータ

expression

最大値が計算される式。通常はカラム名です。

DISTINCT expression

MAX(`expression`) の場合と同じ値を返します。万全を期すために含まれています。

戻り値

引数と同じデータ型。

備考

`expression` が NULL になるローは無視されます。グループにローが含まれていない場合は、NULL を返します。

TABLE REF 型として定義された値は、この関数ではサポートされません。

また、2 つの式を単純に比較する場合は、GREATER 関数を使用することができます。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。この関数は NCHAR の入力または出力をサポートしています。

標準

ANSI/ISO SQL 標準

コア機能。Window 関数として使用する場合、MAX はオプションの ANSI/ISO SQL 言語機能 T611、"Basic OLAP operations" の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、"Full value expressions" の一部です。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、"Extended set function support" もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、"Mixed column references in set function" がサポートされていません。また、MAX 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、Employees テーブル内の給与の最大値 138948.000 を返します。

```
SELECT MAX( Salary )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
GREATER 関数 [その他] [378 ページ]
MIN 関数 [集合] [436 ページ]

1.3.2.131 MEDIAN 関数 [集合]

ローのセットの数値式の中央値を計算します。

構文

式

```
MEDIAN( [ ALL | DISTINCT ] numeric-expression )
```

Window 関数

```
MEDIAN( [ ALL ]  numeric-expression) OVER ( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

パラメータ

numeric-expression

ローセットで中央値が計算される式。

DISTINCT 句

入力に含まれるユニークな値の中央値を計算する前に、重複した値を除外します。

ALL 句

入力に含まれるすべての値 (重複した値も含む) の中央値を計算します。これはデフォルトの動作です。

戻り値

戻り値のデータ型は、入力値と同じです。

中央値の計算では、NULL は無視されます。ただし、ローが含まれないグループについては NULL 値が返されます。

備考

`numeric-expression` 値には、BIT を除くどの数値データ型でも使用できます。

数値の有限リストの中央値は、リスト内の値を最小値から最大値まで順に並べ、その中央にある値を選択することで特定できます。数値が偶数存在する場合は、中央値がユニークにならないため、2 つの中央値の平均が MEDIAN 関数から返されます。多くとも、母集団の半分の要素が中央値より小さく、残りの半分が中央値より大きくなります。両方のグループの要素の数がそれぞれ母集団の半分未満の場合は、中央値と完全に等しい値を持つ要素が存在します。たとえば、`a < b < c` の場合、リスト `{a, b, c}` の中央値は b になります。`a < b < c < d` の場合は、リスト `{a, b, c, d}` の中央値は b と c の平均である (`(b + c)/2`) になります。

中間の 2 つの要素を平均した結果に小数点以下の数値が含まれ、かつ入力データ型で小数部分を表すことができない場合は、小数点以下の数値がトランケートされます。このトランケートを回避するには、小数点以下桁数が有効な数値型に入力をキャストします。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

`window-spec` にはパーティションのみを指定できます (ROW または RANGE の指定を含めることはできません)。WINDOW 句を使用した場合、DISTINCT はサポートされません。CUBE、ROLLUP、および GROUPING SETS は、構文 1 でサポートされます。

標準

ANSI/ISO SQL 標準

標準になし。ただし、Window 関数は、オプションの ANSI/ISO SQL 言語機能 T611、「Basic OLAP operations」の一部です。

このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、「Extended set function support」がサポートされています。これにより、カラム参照ではない任意の式を Window 関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set function」がサポートされていません。このソフトウェアでは、MEDIAN 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、Employees テーブルから給与の中央値を返します。

```
SELECT MEDIAN( Salary ) FROM GROUPO.Employees;
```

次の文は、Employees テーブルから州ごとの給与の中央値を返します。

```
SELECT EmployeeID, Surname, Salary, State,
  MEDIAN( Salary ) OVER Salary_Window
FROM GROUPO.Employees
WINDOW Salary_Window AS ( PARTITION BY State )
ORDER BY State, Surname;
```

関連情報

数値データ型 [137 ページ]
WINDOW 句 [1409 ページ]
SUM 関数 [集合] [554 ページ]
COUNT 関数 [集合] [289 ページ]

1.3.2.132 MICROSECOND 関数 [日付と時刻]

TIMESTAMP 式のマイクロ秒の部分を返します。

構文

```
MICROSECOND( timestamp-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

INTEGER

備考

この関数は、1 秒に満たない時間 (マイクロ秒ともいいます) を表す 0 から 999999 までの間の値を返します。この関数は、`DATEPART( MICROSECOND, timestamp-expression )` と同等です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文を実行すると、マイクロ秒の値 789012 が返されます。

```
SELECT MICROSECOND( '12:34:56.789012' );
```

関連情報

日付の単位の指定 [206 ページ]
SET 文 [T-SQL] [1327 ページ]
EXTRACT 関数 [日付と時刻] [367 ページ]
DATEPART 関数 [日付と時刻] [310 ページ]
MILLISECOND 関数 [日付と時刻] [435 ページ]

1.3.2.133 MILLISECOND 関数 [日付と時刻]

TIMESTAMP 式のミリ秒の部分を返します。

構文

```
MILLISECOND( timestamp-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

INTEGER

備考

この関数は、1 秒に満たない時間をミリ秒単位で表す 0 から 999 までの間の値を返します。タイムスタンプに 1 ミリ秒より小さい時間が含まれている場合は、ミリ秒単位になるよう切り捨てられます。

この関数は、`DATEPART( MILLISECOND, timestamp-expression )` と同等です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文を実行すると、ミリ秒の値 789 が返されます。

```
SELECT MILLISECOND( '12:34:56.78901' );
```

関連情報

日付の単位の指定 [206 ページ]
SET 文 [T-SQL] [1327 ページ]
MICROSECOND 関数 [日付と時刻] [433 ページ]
DATEPART 関数 [日付と時刻] [310 ページ]
EXTRACT 関数 [日付と時刻] [367 ページ]

1.3.2.134 MIN 関数 [集合]

各ローグループで見つかった式の最小値を返します。

構文

式

```
MIN( [ DISTINCT ] expression )
```

Window 関数

```
MIN( [ ALL ] expression ) OVER ( window-spec )
```

```
window-spec: 以下の備考部分を参照してください。
```

Ultra Light 式

```
MIN( [ ALL | DISTINCT ] expression )
```

パラメータ

expression

最小値が計算される式。通常はカラム名です。

DISTINCT expression

MIN(expression) の場合と同じ値を返します。万全を期すために含まれています。

戻り値

引数と同じデータ型。

備考

expression が NULL になるローは無視されます。グループにローが含まれていない場合は、NULL を返します。

TABLE REF 型として定義された値は、この関数ではサポートされません。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light: Ultra Light では NCHAR の入力または出力がサポートされていません。

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

コア機能。Window 関数として使用する場合、MIN はオプションの ANSI/ISO SQL 言語機能 T611、"Basic OLAP operations" の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、"Full value expressions" の一部です。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、"Extended set function support" もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、"Mixed column references in set function" がサポートされていません。また、MIN 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、Employees テーブル内の給与の最小値 24903.000 を返します。

```
SELECT MIN( Salary )
FROM GROUPO.Employees;
```

関連情報

LESSER 関数 [その他] [419 ページ]
WINDOW 句 [1409 ページ]
MAX 関数 [集合] [430 ページ]

1.3.2.135 MINUTE 関数 [日付と時刻]

TIMESTAMP 値の分部分を返します。

構文

```
MINUTE( timestamp-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

SMALLINT

備考

TIMESTAMP 式の分部分である、0 ～ 59 の間の SMALLINT 値が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 22 を返します。

```
SELECT MINUTE( '1998-07-13 12:22:34' );
```

1.3.2.136 MINUTES 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の分境界の数を返します。

構文

0000-02-29 の真夜中と TIMESTAMP 値の間の分数を返します。

```
MINUTES( timestamp-expression )
```

2 つの TIMESTAMP 値の間の分数を返します。

```
MINUTES( timestamp-expression, timestamp-expression )
```

分を TIMESTAMP 値に追加します。

```
MINUTES( timestamp-or-time-expression, integer-expression )
```

パラメータ

timestamp-expression

TIMESTAMP データ型の式。

timestamp-or-time-expression

TIME または TIMESTAMP データ型の式。

integer-expression

`timestamp-or-time-expression` に追加する分数。`integer-expression` が負の場合、適切な分数が `timestamp-or-time-expression` から減算されます。

戻り値

使用方法に応じて INTEGER、TIME、または TIMESTAMP。

備考

MINUTES 関数の結果は、その引数によって異なります。

0000-02-29 の真夜中以降の分数を返します。

1 つの `timestamp-expression` を MINUTES 関数に渡すと、0000-02-29 の真夜中から `timestamp-expression` までの秒数を INTEGER として返します。

注記

0000-02-29 は実際の日付を指すための値ではありません。MINUTES 関数で使用されるデフォルトの日付です。

2 つの TIMESTAMP 値の間の分数を返します。

2 つの TIMESTAMP 値を MINUTES 関数に渡すと、その間の分境界の数を整数で返します。

分を TIMESTAMP 値に追加します。

TIMESTAMP 値と INTEGER 値を MINUTES 関数に渡すと、整数の分数を `timestamp-expression` 引数に加算した TIMESTAMP 結果を返します。同様に、MINUTES の最初の引数が TIME 値の場合は、結果も TIME 値になります。この構文では、最初の引数の暗黙的変換をサポートしていません。最初の引数を DATE、TIME または TIMESTAMP の値に明示的にキャストしてください。最初の引数が DATE データ型の場合、時刻部分には真夜中が使用されます。

MINUTES は整数を返すので、4083-03-23 02:08:00 以上の TIMESTAMP 値で使用すると、オーバーフローが起こる場合があります。

計算の一部に DATEDIFF 関数と DATEADD 関数を使用することもできます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、同一の値 240 を返します。これは、2 番目の TIMESTAMP 値が、最初のタイムスタンプの 240 分後であることを示します。DATEDIFF を使用することをおすすめします。

```
SELECT
  MINUTES( '1999-07-13 06:07:12',
           '1999-07-13 10:07:12' ),
  DATEDIFF( minute,
           '1999-07-13 06:07:12',
           '1999-07-13 10:07:12' );
```

次の文は、値 1051040527 を返します。

```
SELECT MINUTES( '1998-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 1999-05-12 21:10:07.000 を返します。最初の文では、リテラル文字列パラメータの明示的なキャストが必要です。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT MINUTES( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5);
SELECT DATEADD( minute, 5, '1999-05-12 21:05:07' );
```

関連情報

DATEDIFF 関数 [日付と時刻] [305 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.3.2.137 MOD 関数 [数値]

整数を整数で割ったときの余りを返します。

構文

```
MOD( dividend, divisor)
```

パラメータ

dividend

被除数 (分数の分子)。

divisor

除数 (分数の分母)。

戻り値

- SMALLINT
- INT
- NUMERIC

備考

被除数が負の場合、除算の結果は負または 0 になります。除数の符号は計算結果に影響を与えません。

標準

ANSI/ISO SQL 標準

MOD 関数は、オプションの ANSI/ISO SQL 言語機能 T441 の一部です。

例

次の文は、値 2 を返します。

```
SELECT MOD( 5, 3 );
```

関連情報

REMAINDER 関数 [数値] [499 ページ]

1.3.2.138 MONTH 関数 [日付と時刻]

指定した日付の月を返します。

構文

```
MONTH( date-expression )
```

パラメータ

date-expression

DATE データ型の値。

戻り値

SMALLINT

備考

戻り値は、指定の日付の月に対応する 1 ～ 12 の間の数です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 7 を返します。

```
SELECT MONTH( '1998-07-13' );
```

1.3.2.139 MONTHNAME 関数 [日付と時刻]

日付から月の名前を返します。

構文

```
MONTHNAME( date-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

VARCHAR

備考

結果が数値 (2 月を示す 2 など) の場合でも、MONTHNAME 関数は文字列を返します。

SQL Anywhere では、英語のロケールでは英語の名前が返され、英語以外のロケールでは英語以外の名前が返されます。たとえば、Language (LANG) 接続パラメータを使用すれば他の言語を指定できます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、英語のロケールの場合、`September` という値を返します。

```
SELECT MONTHNAME( '1998-09-05' );
```

SQL Anywhere では、フランス語のロケールの場合、`septembre` という値を返します。スペイン語のロケールの場合、`Septiembre` という値を返します。複数のロケールがサポートされています。

関連情報

DATENAME 関数 [日付と時刻] [308 ページ]
DATEPART 関数 [日付と時刻] [310 ページ]

1.3.2.140 MONTHS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の月境界の数を返します。

構文

0000-02 と TIMESTAMP 値の間の月数を返します。

```
MONTHS( timestamp-expression )
```

2 つの TIMESTAMP 値の間の月数を返します。

```
MONTHS( timestamp-expression, timestamp-expression )
```

月を TIMESTAMP 値に追加します。

```
MONTHS( timestamp-expression, integer-expression )
```

パラメータ

timestamp-expression

TIMESTAMP データ型の日付と時刻。

integer-expression

`timestamp-expression` に加算される整数の月数 (SMALLINT データ型)。`integer-expression` が負の場合、適切な月数が `timestamp-expression` から減算されます。`integer-expression` を指定する場合は、`timestamp-expression` を TIME、DATE または TIMESTAMP データ型として明示的にキャストしてください。`timestamp-expression` が TIME 値の場合、現在の月が使用されます。

戻り値

使用方法に応じて INTEGER または TIMESTAMP。

備考

MONTHS 関数の結果は、その引数によって異なります。MONTHS 関数では、引数内の時間、分、および秒は無視されます。

0000-02 以降の月数を返します。

1 つの `timestamp-expression` を MONTHS 関数に渡すと、0000-02 から `timestamp-expression` までの月数を INTEGER として返します。

> **注記**
>
> 0000-02-29 は実際の日付を指すための値ではありません。MONTHS 関数で使用されるデフォルトの日付です。

2 つの TIMESTAMP 値の間の月数を返します。

2 つの TIMESTAMP 値を MONTHS 関数に渡すと、その間の月境界を整数で返します。

月を TIMESTAMP 値に追加します。

TIMESTAMP 値と SMALLINT 値を MONTHS 関数に渡すと、整数の月数を `timestamp-expression` に加算した TIMESTAMP 結果を返します。

DATEDIFF 関数と DATEADD 関数を使用すると、これらの計算の一部を実行できます。

MONTHS の値を求めるには、2 つの日付の間に月の最初の日がいくつあるかを計算します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 2 を返します。これは、2 番目の日付が、最初の日付の 2 か月後であることを示します。2 つ目の例 (DATEDIFF) の使用をお奨めします。

```
SELECT MONTHS( '1999-07-13 06:07:12', '1999-09-13 10:07:12' );
```

```
SELECT DATEDIFF( month,
   '1999-07-13 06:07:12',
   '1999-09-13 10:07:12' );
```

次の文は、値 23981 を返します。

```
SELECT MONTHS( '1998-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 1999/10/12 21:05:07.000 を返します。2 つ目の例 (DATEADD) の使用をお奨めします。

```
SELECT MONTHS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5);
```

```
SELECT DATEADD( month, 5, '1999-05-12 21:05:07' );
```

関連情報

DATEDIFF 関数 [日付と時刻] [305 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.3.2.141 NCHAR 関数 [文字列]

Unicode のコードポイントがパラメータに指定された 1 文字を含む NCHAR 文字列を返します。値が有効なコードポイント値ではない場合は、NULL を返します。

構文

```
NCHAR( integer )
```

パラメータ

integer

対応する Unicode コードポイントに変換される数値。

戻り値

NVARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、アラビア文字の ALEF (Unicode コードポイント U+627) を返します。

```
SELECT NCHAR( 1575 );
```

関連情報

CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]
TO_NCHAR 関数 [文字列] [564 ページ]
TO_CHAR 関数 [文字列] [562 ページ]
UNICODE 関数 [文字列] [578 ページ]
UNISTR 関数 [文字列] [579 ページ]

1.3.2.142 NEWID 関数 [その他]

UUID (ユニバーサルユニーク識別子) 値を生成します。UUID は、GUID (グローバルユニーク識別子) と同じです。

構文

```
NEWID( )
```

パラメータ

NEWID 関数に関連付けられているパラメータはありません。

戻り値

UNIQUEIDENTIFIER

備考

NEWID 関数は、カラムの DEFAULT 句で使用できます。

UUID を使用して、テーブルのローをユニークに識別できます。コンピュータが異なると生成される値も異なるので、値は同期環境やレプリケーション環境でキーとして使用できます。

他の RDBMS と互換性を保つために、UUID にはハイフンが含まれます。このデフォルト設定は、uuid_has_hyphens option を Off にして変更することができます。

Ultra Light の UUID にはハイフンは含まれません。

NEWID 関数は、非決定的関数です。NEWID 関数を連続して呼び出すと、毎回異なる値が返されます。クエリオプティマイザは、NEWID 関数の結果をキャッシュしません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、2 つのカラムを持つテーブル mytab を作成します。カラム pk は uniqueidentifier データ型とし、NEWID 関数をデフォルト値として割り当てます。カラム c1 は integer データ型です。

```
CREATE TABLE mytab(
   pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
   c1 INT );
```

次の文は、ユニーク識別子を文字列として返します。

```
SELECT UUIDTOSTR( NEWID() );
```

たとえば、戻り値が 96603324-6FF6-49DE-BF7D-F44C1C7E6856 の場合もあります。

関連情報

STRTOUUID 関数 [文字列] [548 ページ]
UUIDTOSTR 関数 [文字列] [584 ページ]

UNIQUEIDENTIFIER データ型 [175 ページ]

1.3.2.143 NEXT_CONNECTION 関数 [システム]

次の接続の識別番号を返します。

構文

```
NEXT_CONNECTION( connection-id [, database-id ] )
```

パラメータ

connection-id

整数。通常は、前回の NEXT_CONNECTION の呼び出しから返された整数です。`connection-id` が NULL の場合、NEXT_CONNECTION は最新の接続 ID を返します

database-id

現在のサーバ上のいずれかのデータベースを表す整数。`database-id` を指定しない場合は、現在のデータベースが使用されます。NULL を指定した場合、NEXT_CONNECTION はデータベースに関係なく、次の接続を返します。

戻り値

INT

備考

NEXT_CONNECTION を使用して、データベースへの接続を列挙できます。通常、接続 ID は単調に増加する昇順で作成されます。この関数は次の接続 ID を降順で返します。

最新の接続の接続 ID 値を取得するには、`connection-id` に NULL を入力します。以降の接続を取得するには、その前の戻り値を入力します。その順序で接続がなくなると、NULL を返します。

NEXT_CONNECTION は、特定の時間の前にすべての接続を切断するのに役立ちます。ただし、NEXT_CONNECTION は接続 ID を降順で返すため、関数の開始後に作成された接続は返されません。すべての接続を確実に切断するには、NEXT_CONNECTION を実行する前に新しい接続を作成しないようにします。

権限

この関数を実行して現在の接続 ID を返す場合、権限は必要ありません。他の接続に対してこの関数を実行する場合は、SERVER OPERATOR または DROP MONITOR のシステム権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在のデータベースでの最初の接続に使用する識別子を整数値として返します。

```
SELECT NEXT_CONNECTION( NULL );
```

次の文は、5 のような整数値を返します。

```
SELECT NEXT_CONNECTION( 10 );
```

次の呼び出しは、現在のデータベースについて、指定された connection-id から降順で次の接続 ID を返します。

```
SELECT NEXT_CONNECTION( connection-id );
```

次の呼び出しは、データベースに関係なく、指定された connection-id から降順で次の接続 ID を返します。

```
SELECT NEXT_CONNECTION( connection-id, NULL );
```

次の呼び出しは、指定されたデータベースについて、指定された connection-id から降順で次の接続 ID を返します。

```
SELECT NEXT_CONNECTION( connection-id, database-id );
```

次の呼び出しは、データベースに関係なく最初の接続を返します。

```
SELECT NEXT_CONNECTION( NULL, NULL );
```

次の呼び出しは、指定されたデータベースの最初の接続を返します。

```
SELECT NEXT_CONNECTION( NULL, database-id );
```

1.3.2.144 NEXT_DATABASE 関数 [システム]

データベースの識別番号を返します。

構文

```
NEXT_DATABASE( database-id )
```

パラメータ

database-id

データベースの ID 番号を指定する整数。

戻り値

INT

備考

NEXT_DATABASE 関数を使用して、データベースサーバで実行中のデータベースを列挙できます。最初のデータベースを取得するには、NULL を渡指定します。その後の各データベースを取得するには、前回の戻り値を指定ます。データベースがなくなると、NULL を返します。データベース ID 番号は、特定の順序で返されるわけではありませんが、データベース ID を使用して、サーバでデータベースが起動された順序を示すことができます。サーバで最初に起動されたデータベースには値 0 が割り当てられ、サーバでそれ以降に起動されたデータベースは、データベース ID が 1 ずつ増えていきます。

権限

この関数を実行して現在のデータベースを返す場合、権限は必要ありません。この関数を他のデータベースに対して実行するには、SERVER OPERATOR または MONITOR のシステム権限を持っていることが必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、最初のデータベースの値である 0 を返します。

```
SELECT NEXT_DATABASE( NULL );
```

次の文は、1 つのデータベースが起動している場合のみ NULL を返します。

```
SELECT NEXT_DATABASE( 0 );
```

関連情報

DB_NAME 関数 [システム] [322 ページ]
sa_db_list システムプロシージャ [1481 ページ]

1.3.2.145 NEXT_HTTP_HEADER 関数 [Web サービス]

次の HTTP ヘッダの名前を返します。

構文

```
NEXT_HTTP_HEADER( header-name )
```

パラメータ

header-name

前の要求ヘッダの名前。header-name が NULL の場合、この関数は最初の HTTP 要求ヘッダの名前を返します。

戻り値

LONG VARCHAR。

注記

結果のデータ型は LONG VARCHAR です。SELECT INTO 文で NEXT_HTTP_HEADER を使用する場合は、Unstructured Data Analytics Option ライセンスが必要になります。または CAST を使用して HTML_DECODE を正しいデータ型とサイズに設定する必要があります。

備考

この関数を HTTP 要求ヘッダで繰り返し使用することで、次の HTTP ヘッダ名が返されます。NULL を指定して呼び出すと、最初のヘッダの名前が返されます。後続のヘッダは、前のヘッダの名前を関数に渡すことによって取得されます。この関数を最後のヘッダ名を使用して呼び出した場合、または Web サービスから呼び出していない場合、NULL を返します。

この関数を繰り返し呼び出すと、すべてのヘッダフィールドが一度だけ返されます。ただし、必ずしも HTTP 要求での表示順に表示されるとはかぎりません。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、データベースサーバメッセージウィンドウの HTTP 要求ヘッダの名前と値が表示されます。

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name  = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_HEADER( header_name );
    MESSAGE 'HEADER: ', header_name, '=',
            header_value TO CONSOLE;
  END LOOP;
END;
```

関連情報

Web サービス関数 [212 ページ]
HTTP_HEADER 関数 [Web サービス] [395 ページ]
sa_http_header_info システムプロシージャ [1526 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.146 NEXT_HTTP_RESPONSE_HEADER 関数 [Web サービス]

次の HTTP 応答ヘッダの名前を取得します。

構文

```
NEXT_HTTP_RESPONSE_HEADER( header-name )
```

パラメータ

header-name

前の応答ヘッダの名前。header-name が NULL の場合、この関数は最初の HTTP 応答ヘッダの名前を返します。

戻り値

LONG VARCHAR

備考

この関数を HTTP 応答ヘッダで繰り返し使用することで、次の HTTP 応答ヘッダ名が返されます。NULL を指定して呼び出すと、最初の応答ヘッダの名前が返されます。後続の応答ヘッダは、前の応答ヘッダの名前を関数に渡すことによって取得されます。この関数は、最後の応答ヘッダ名を使用して呼び出した場合、または Web サービスから呼び出していない場合、NULL を返します。

この関数を繰り返し呼び出すと、すべての応答ヘッダフィールドが 1 回のみ返されます。ただし、必ずしも HTTP 応答での表示順に表示されるとはかぎりません。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、データベースサーバメッセージウィンドウの HTTP 応答ヘッダの名前と値が表示されます。

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name  = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_RESPONSE_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_RESPONSE_HEADER( header_name );
    MESSAGE 'RESPONSE HEADER: ', header_name, '=', header_value TO CONSOLE;
  END LOOP;
```

関連情報

Web サービス関数 [212 ページ]
HTTP_RESPONSE_HEADER 関数 [Web サービス] [397 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.147 NEXT_HTTP_VARIABLE 関数 [Web サービス]

次の HTTP 変数の名前を返します。

構文

```
NEXT_HTTP_VARIABLE( var-name)
```

パラメータ

var-name

前の変数の名前。`var-name` が NULL の場合、この関数は最初の HTTP 変数の名前を返します。

戻り値

LONG VARCHAR。

備考

この関数は、要求内の HTTP 変数に対して反復して適用されます。NULL を指定して呼び出すと、最初の変数の名前が返されます。後続の変数は、前の変数の名前を関数に渡すことによって取得されます。この関数は、最後の変数名を使用して呼び出した場合、または Web サービスから呼び出していない場合、NULL を返します。

この関数を繰り返し呼び出すと、すべての変数が一度だけ返されます。ただし、必ずしも HTTP 要求での表示順に表示されるとはかぎりません。URL PATH が ON または ELEMENTS に設定される場合、変数 url または url1、url2、...、url10 が個々に含められます。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で次の文が使用されると、最初の HTTP 変数の名前が返されます。

```
BEGIN
DECLARE variable_name LONG VARCHAR;
DECLARE variable_value LONG VARCHAR;
SET variable_name = NULL;
SET variable_name = NEXT_HTTP_VARIABLE( variable_name );
SET variable_value = HTTP_VARIABLE( variable_name );
END;
```

関連情報

Web サービス関数 [212 ページ]
HTTP_VARIABLE 関数 [Web サービス] [399 ページ]
NEXT_HTTP_HEADER 関数 [Web サービス] [452 ページ]
sa_http_variable_info システムプロシージャ [1532 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.148 NEXT_SOAP_HEADER 関数 [SOAP]

SOAP 要求ヘッダの次のヘッダキーを返します。

構文

```
NEXT_SOAP_HEADER( header-key )
```

パラメータ

header-key

指定したヘッダエントリの最上位 XML 要素の XML ローカル名。

戻り値

LONG VARCHAR

備考

`header-key` に NULL を指定すると、この関数は SOAP ヘッダで見つかった最初のヘッダエントリのヘッダキーが返されます。

最後の `header-key` を使用して呼び出すと、この関数は NULL が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で使用された場合、次の文は SOAP 要求ヘッダにあるすべてのキーを処理します。認証キーを処理するときに、キーの値も取得します。

```
BEGIN
  DECLARE hd_key LONG VARCHAR;
  DECLARE hd_entry LONG VARCHAR;
header_loop:
  LOOP
```

```
    SET hd_key = NEXT_SOAP_HEADER( hd_key );
    IF hd_key IS NULL THEN
      -- no more header entries
      LEAVE header_loop;
    END IF;
    IF hd_key = 'Authentication' THEN
      SET hd_entry = SOAP_HEADER( hd_key );
    END IF;
  END LOOP header_loop;
END;
```

関連情報

Web サービス関数 [212 ページ]
SOAP_HEADER 関数 [SOAP] [529 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.149 NOW 関数 [日付と時刻]

現在の日付と時刻を TIMESTAMP 値で返します。精度はシステムクロックの精度によって制限されます。

構文

```
NOW( [ * ] )
```

戻り値

TIMESTAMP

備考

NOW は、GETDATE 関数および CURRENT TIMESTAMP 特別値と同義です。NOW(*) と NOW() の構成は同義です。

要求に含まれる NOW 関数の各インスタンスは、多くても 1 回のみ評価されます。同じ要求に含まれる NOW の複数のインスタンスは、同じ TIMESTAMP 値を共有する場合もあれば、そうでない場合もあります。

注記

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこの関数の結果が計算されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在の日付と時刻を返します。

```
SELECT NOW( * );
```

関連情報

CURRENT TIME 特別値 [90 ページ]
CURRENT TIMESTAMP 特別値 [92 ページ]
CURRENT UTC TIMESTAMP 特別値 [95 ページ]
DATE データ型 [161 ページ]
DATE 関数 [日付と時刻] [303 ページ]
DATETIME データ型 [162 ページ]
DATETIME 関数 [日付と時刻] [311 ページ]
DATETIMEOFFSET データ型 [164 ページ]
SQL 文の式 [32 ページ]
GETDATE 関数 [日付と時刻] [375 ページ]
ISDATE 関数 [データ型変換] [407 ページ]
SMALLDATETIME データ型 [166 ページ]
TIME データ型 [167 ページ]
TIMESTAMP 特別値 [106 ページ]
TIMESTAMP データ型 [168 ページ]
UTC TIMESTAMP 特別値 [109 ページ]

1.3.2.150 NULLIF 関数 [その他]

式を比較して、CASE 式の省略形を提供します。

構文

```
NULLIF( expression-1, expression-2 )
```

パラメータ

expression-1

比較される式。

expression-2

比較される式。

戻り値

最初の引数のデータ型。

備考

NULLIF は 2 つの式の値を比較します。

1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。

1 番目の式と 2 番目の式が異なる場合、または 2 番目の式が NULL の場合、NULLIF は 1 番目の式を返します。

NULLIF 関数は、いくつかの CASE 式の簡単な作成方法を提供します。

標準

ANSI/ISO SQL 標準

コア機能。

例

次の文は、値 `a` を返します。

```
SELECT NULLIF( 'a', 'b' );
```

次の文は、`NULL` を返します。

```
SELECT NULLIF( 'a', 'a' );
```

関連情報

CASE 式 [36 ページ]

1.3.2.151 NUMBER 関数 [その他]

クエリの結果の連続した各ローに対して、1 から開始する番号を生成します。NUMBER 関数は主に、SELECT リストで使用するために提供されています。

構文

```
NUMBER( [ * ] )
```

戻り値

INT

備考

以下で説明する NUMBER 関数には制限があるため、代わりに ROW_NUMBER 関数を使用します。ROW_NUMBER 関数には NUMBER と同じ機能ですが、NUMBER 関数にある制限はありません。

SELECT リストで NUMBER(*) を使用すると、結果セットのローに連番を付けることができます。NUMBER(*) は、各結果ローの ANSI ロー番号の値を返します。NUMBER 関数は、アプリケーションがどのように結果セットをスクロールするかに応じて、正または負の値を返します。insensitive カーソルの場合は、OPEN 時に結果セット全体が実体化されるため、NUMBER(*) は常に正の値を返します。

また、カーソルタイプによってはロー番号が変更される場合もあります。insensitive カーソルとスクロールカーソルの値は固定されています。同時更新を実行すると、動的カーソルと sensitive カーソルの値が変更される場合があります。

DELETE 文、WHERE 句、HAVING 句、ORDER BY 句、サブクエリ、集合を含むクエリ、いずれかの制約、GROUP BY 句、DISTINCT 句、集合演算子 (UNION、EXCEPT、INTERSECT)、派生テーブルで NUMBER 関数を使用すると、構文エラーになります。

NUMBER(*) は、ビューで使用できますが (前述の制限を受ける)、NUMBER(*) を伴う式に対応したビューのカラムは、クエリまたは外部ビューで多くても一度しか参照できません。また、ビューは左外部ジョインまたは全外部ジョインの NULL 入力テーブルとして使用できません。

Embedded SQL で、NUMBER(*) 関数があるクエリを参照するカーソルを使用する場合は、十分に注意してください。特に、データベースカーソルがカーソルの最後からの相対位置 (負のオフセットによる絶対位置) に配置されている場合、この関数は負の数を返します。

NUMBER は、UPDATE 文の SET 句の代入式の右側で使用できます。たとえば、`SET x = NUMBER(*)` のように記述します。

また、SELECT 文から INSERT を使用すると、NUMBER 関数を使用してプライマリキーを生成できます。ただし、連続したプライマリキーの生成には、AUTOINCREMENT 句の使用をお奨めします。

セマンティック上、NUMBER(*) と NUMBER() は同義です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、連番が付けられた部署リストを返します。

```
SELECT NUMBER( * ), DepartmentName
FROM GROUPO.Departments
WHERE DepartmentID > 5
ORDER BY DepartmentName;
```

関連情報

ROW_NUMBER 関数 [その他] [512 ページ]
CREATE TABLE 文 [952 ページ]
INSERT 文 [1167 ページ]

1.3.2.152 PATINDEX 関数 [文字列]

文字列のパターンが最初に出現した開始位置を表す整数を返します。

構文

```
PATINDEX( '%pattern%', string-expression )
```

パラメータ

pattern

検索するパターン。先頭の % ワイルドカードを省略すると、PATINDEX 関数は、パターンが文字列の先頭に出現する場合は 1 を返し、それ以外の場合は 0 を返します。

パターンは、LIKE 比較と同じワイルドカードを使用します。これらのワイルドカードを次の表に示します。

Ultra Light のパターンは、次の表のワイルドカードを使用します。

ワイルドカード	一致するもの
_ (アンダースコア)	任意の 1 文字
% (パーセント記号)	0 個以上の文字からなる任意の文字列
[]	指定範囲内、または一連の指定文字の任意の 1 文字
[^]	指定範囲外、または一連の指定文字以外の任意の 1 文字

string-expression

パターンを検索する文字列。

戻り値

INT

備考

PATINDEX 関数は、パターンが最初に出現した開始位置を返します。パターンが見つからない場合は、0 を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 2 を返します。

```
SELECT PATINDEX( '%hoco%', 'chocolate' );
```

次の文は、値 11 を返します。

```
SELECT PATINDEX( '%4_5_', '0a1A 2a3A 4a5A' );
```

次の文は、14 を返します。これは、文字列式の最初の英数字以外の文字です。データベースで大文字と小文字が区別されない場合は、`'%[^a-zA-Z0-9]%'` の代わりにパターン `'%[^a-z0-9]%'` を使用できます。

```
SELECT PATINDEX( '%[^a-zA-Z0-9]%', 'SQLAnywhere17 has many new features' );
```

次の文を使用して、文字列の最初の英数字以外の文字までのすべての文字を取得することができます。

```
SELECT LEFT( @string, PATINDEX( '%[^a-zA-Z0-9]%', @string ) );
```

次の文は、myTable というテーブルを作成し、英数字文字、スペース (ブランク)、英数字以外の文字が含まれる各種の文字列を入力します。続いて、SELECT 文とその後の結果により、PATINDEX を使用して文字列内のスペースと英数字以外の文字の開始位置を検索する方法がわかります。

```
CREATE TABLE myTable( col1 LONG VARCHAR );
INSERT INTO myTable (col1) VALUES( 'the quick brown fox jumped over the lazy
dog' ),
( 'the quick brown fox $$$$ jumped over the lazy dog' ),
( 'the quick brown fox 0999 jumped over the lazy dog' ),
( 'the quick brown fox ** jumped over the lazy dog' ),
( 'thequickbrownfoxjumpedoverthelazydog' ),
( 'thequickbrownfoxjum999pedoverthelazydog' ),
( 'thequick$$$$brownfox' ),
( 'the quick brown fox$$ jumped over the lazy dog' );
SELECT col1,
   //position of first non-alphanumeric character or space:
   PATINDEX( '%[^a-z0-9]%', col1) AS blank_posn,
   //position of first non-alphanumeric char that isn't a space:
   PATINDEX( '%[^ a-z0-9]%', col1) AS non_alpha_char,
   //everything up to and including first non-alphanumeric char that isn't a
space:
   LEFT ( col1, PATINDEX( '%[^ a-zA-Z0-9]%', col1) ) AS left_str,
   //first non-alphanumeric char that isn't a space, and everything to the right:
   SUBSTRING ( col1, PATINDEX( '%[^ a-zA-Z0-9]%', col1) ) AS sub_str
FROM myTable;
```

col1	blank_posn	non_alpha_char	left_str	sub_str
the quick brown fox jumped over the lazy dog	4	0		the quick brown fox jumped over the lazy dog
the quick brown fox $$$$ jumped over the lazy dog	4	21	the quick brown fox $	$$$$ jumped over the lazy dog
the quick brown fox 0999 jumped over the lazy dog	4	0		the quick brown fox 0999 jumped over the lazy dog
the quick brown fox ** jumped over the lazy dog	4	21	the quick brown fox *	** jumped over the lazy dog
thequickbrownfoxjumpedoverthelazydog	0	0		thequickbrownfoxjumpedoverthelazydog
thequickbrownfoxjum999pedoverthelazydog	0	0		thequickbrownfoxjum999pedoverthelazydog
thequick$$$$brownfox	9	9	thequick$	$$$$brownfox

col1	blank_posn	non_alpha_char	left_str	sub_str
the quick brown fox$$ jumped over the lazy dog	4	20	the quick brown fox$	$$ jumped over the lazy dog

関連情報

文字列関数 [213 ページ]
LIKE 探索条件 [63 ページ]
LOCATE 関数 [文字列] [423 ページ]

1.3.2.153 PERCENT_RANK 関数 [ランキング]

ロー X が関数の引数と ORDER BY 指定で定義される場合、PERCENT_RANK 関数は、グループのロー数で割ったロー X - 1 のランクを計算します。

構文

PERCENT_RANK( ) OVER (`window-spec` *)*

`window-spec`: 以下の備考部分を参照してください。

戻り値

PERCENT_RANK 関数は、0 ～ 1 の DOUBLE 値を返します。

備考

`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。Window 関数として使用する場合、ORDER BY 句を指定する必要があります。また PARTITION BY 句は指定できますが、ROWS 句や RANGE 句はいずれも指定できません。WINDOW 句の `window-spec` 定義を参照してください。

標準

ANSI/ISO SQL 標準

PERCENT_RANK は、オプションの ANSI/ISO SQL 言語機能 T612、「Advanced OLAP operations」の一部です。

例

次の例は、ニューヨークの従業員の給与ランキングを性別ごとに示す結果セットを降順で返します。

```
SELECT DepartmentID, Surname, Salary, Sex,
PERCENT_RANK() OVER (PARTITION BY Sex
ORDER BY Salary DESC) "Rank"
FROM GROUPO.Employees
WHERE State IN ('NY');
```

DepartmentID	Surname	Salary	Sex	Rank
200	Martel	55700.000	M	0
100	Guevara	42998.000	M	0.333333333
100	Soo	39075.000	M	0.666666667
400	Ahmed	34992.000	M	1
300	Davidson	57090.000	F	0
400	Blaikie	54900.000	F	0.333333333
100	Whitney	45700.000	F	0.666666667
400	Wetherby	35745.000	F	1

関連情報

WINDOW 句 [1409 ページ]
CUME_DIST 関数 [ランキング] [299 ページ]
DENSE_RANK 関数 [ランキング] [332 ページ]
RANK 関数 [ランキング] [479 ページ]

1.3.2.154 PI 関数 [数値]

PI の数値を返します。

構文

```
PI( [ * ] )
```

戻り値

DOUBLE

備考

この関数は DOUBLE 値を返します。

セマンティック上、PI(*) と PI() は同義です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 3.141592653(...) を返します。

```
SELECT PI( * );
```

1.3.2.155 PLAN 関数 [その他]

SQL 文の長いプランの最適化方法を文字列で返します。

構文

```
PLAN( string-expression, [ cursor-type [ update-status ] ]  )
```

パラメータ

string-expression

SQL 文。通常は SELECT 文ですが、UPDATE 文、MERGE 文、または DELETE 文も指定できます。

cursor-type

文字列。`cursor-type` に使用できる値は asensitive (デフォルト)、insensitive、sensitive、または keyset-driven です。

update-status

次のいずれかの値を受け入れる文字列パラメータ。これらの値は、指定されたカーソルをオプティマイザがどのように処理するかを示します。

値	説明
READ-ONLY	このカーソルは読み込み専用です。
READ-WRITE (デフォルト)	このカーソルは読み込みや書き込みが可能です。
FOR UPDATE	このカーソルは読み込みや書き込みが可能です。READ-WRITE とまったく同じです。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。

```
SELECT PLAN(
   'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100' );
```

この情報は、追加するインデックスの決定や、良いパフォーマンスを得るためのデータベース構造の決定に役立ちます。

次の文は、`SELECT * FROM Departments WHERE DepartmentID > 100;` クエリに対する INSENSITIVE カーソルのプランをテキスト形式で表した文字列を返します。

```
SELECT PLAN(
   'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100',
   'insensitive',
   'read-only' );
```

関連情報

EXPLANATION 関数 [その他] [362 ページ]
GRAPHICAL_PLAN 関数 [その他] [376 ページ]

1.3.2.156 POWER 関数 [数値]

数のべき乗を表す数を計算します。

構文

```
POWER( numeric-expression-1, numeric-expression-2 )
```

パラメータ

numeric-expression-1

べき乗の底。

numeric-expression-2

指数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。引数のいずれかが NULL の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

POWER 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 64 を返します。

```
SELECT POWER( 2, 6 );
```

1.3.2.157 PROPERTY 関数 [システム]

指定したデータベースサーバのプロパティの値を文字列で返します。

構文

```
PROPERTY( { property-id | property-name } [, second-parameter ] )
```

パラメータ

property-id

データベースサーバプロパティのプロパティ番号を表す整数。この番号は、PROPERTY_NUMBER 関数で調べることができます。プロパティのセットを繰り返し処理する場合は、`property-id` がよく使用されます。

property-name

データベースプロパティの名前を指定する文字列。

second-parameter

次に示すように、一部のプロパティには 2 つ目のパラメータを指定できます。

プロパティ	2 つ目のパラメータ	説明
EventTypeDesc	`positive-integer`	イベントタイプの説明を返すイベント ID を指定します。
EventTypeName	`positive-integer`	イベントタイプ名を返すイベント ID を指定します。
FunctionMaxParms	`positive-integer`	関数に指定できるパラメータの最大数を返す関数番号を指定します。
FunctionMinParms	`positive-integer`	関数に指定する必要のあるパラメータの最小数を返す関数番号を指定します。
FunctionName	`positive-integer`	関数名を返す関数番号を指定します。
Message	`positive-integer`	データベースサーバメッセージウィンドウの対応する行の内容を返す行番号を指定します。行の先頭にはメッセージが表示された日付と時刻が追加されます。
MessageText	`positive-integer`	データベースサーバメッセージウィンドウ内の指定された行番号に対応するテキスト (日付と時刻を含まない) を返す行番号を指定します。
MessageTime	`positive-integer`	データベースサーバメッセージウィンドウ内の指定された行番号に対応する日付と時刻を返す行番号を指定します。

プロパティ	2 つ目のパラメータ	説明
RemoteCapability	positive-integer	ID に対応するリモート機能名を返すリモート機能 ID を指定します。

戻り値

VARCHAR、LONG VARCHAR

備考

各プロパティには、番号と名前の両方があります。ただし、番号はリリース間で変更されることがあるため、特定のプロパティの信頼できる識別子としては使用しないでください。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在のデータベースサーバ名を返します。

```
SELECT PROPERTY( 'Name' );
```

関連情報

DB_PROPERTY 関数 [システム] [324 ページ]

1.3.2.158 PROPERTY_DESCRIPTION 関数 [システム]

プロパティの説明を返します。

構文

```
PROPERTY_DESCRIPTION( { property-id | property-name } )
```

パラメータ

property-id

データベースプロパティのプロパティ番号を表す整数。この番号は、PROPERTY_NUMBER 関数で調べることができます。プロパティのセットを繰り返し処理する場合は、`property-id` がよく使用されます。

property-name

データベースプロパティの名前を指定する文字列。

戻り値

VARCHAR

備考

各プロパティには、番号と名前の両方があります。ただし、番号はリリース間で変更されることがあるため、特定のプロパティの信頼できる識別子としては使用しないでください。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、IndAdd プロパティの説明である `Number of index insertions` を返します。

```
SELECT PROPERTY_DESCRIPTION( 'IndAdd' );
```

1.3.2.159 PROPERTY_IS_TRACKABLE 関数 [システム]

追跡された値を格納することによって指定したデータベースサーバプロパティの履歴データを更新できるかどうかを返します。

構文

```
PROPERTY_IS_TRACKABLE( property-ID )
```

パラメータ

property-ID

データベースサーバプロパティの PropNum。sa_eng_properties システムプロシージャを実行するか、PROPERTY_NUMBER 関数を呼び出すことで、データベースサーバプロパティの PropNum を確認できます。

戻り値

データベースサーバプロパティを追跡できる場合は 1、それ以外の場合は 0 を返します。

備考

数値を返すデータベースプロパティのみ追跡できます。

例

次の例は、追跡可能なすべてのデータベースサーバプロパティを返します。

```
SELECT PropName FROM sa_eng_properties( ) WHERE PROPERTY_IS_TRACKABLE( PropNum )
= 1;
```

関連情報

sp_property_history システムプロシージャ [1707 ページ]
sa_server_option システムプロシージャ [1606 ページ]
sa_db_option システムプロシージャ [1482 ページ]
sa_eng_properties システムプロシージャ [1502 ページ]
PROPERTY_NUMBER 関数 [システム] [474 ページ]

1.3.2.160 PROPERTY_NAME 関数 [システム]

指定した接続レベルで、指定したプロパティ ID を持つプロパティの名前を返します。

構文

```
PROPERTY_NAME( property-id [, property-scope ] )
```

```
property-scope:
```

```
NULL
| 'server'
| 'database'
| 'db'
| 'connection'
| 'conn'
```

パラメータ

property-id

データベースプロパティのプロパティ ID。

property-scope

プロパティのスコープまたは NULL。

戻り値

VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、プロパティ ID 102 に対応するサーバレベルのプロパティを返します。

```
SELECT PROPERTY_NAME( 102, 'server' );
```

1.3.2.161 PROPERTY_NUMBER 関数 [システム]

指定したプロパティ名を持つプロパティのプロパティ番号を返します。

構文

```
PROPERTY_NUMBER( property-name )
```

パラメータ

property-name

プロパティ名。

戻り値

INT

備考

各プロパティには、番号と名前の両方があります。ただし、番号はリリース間で変更されることがあるため、特定のプロパティの信頼できる識別子としては使用しないでください。プロパティ番号またはプロパティ名をいずれかを使用できる場合は、プロパティ名を使用することをお奨めします。確実に現在サーバで使用しているプロパティ番号にするために、常に PROPERTY_NUMBER を使用します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、PAGESIZE プロパティのプロパティ数を整数で返します。

```
SELECT PROPERTY_NUMBER( 'PAGESIZE' );
```

1.3.2.162 QUARTER 関数 [日付と時刻]

指定した TIMESTAMP 式から、四半期を示す数を返します。

構文

```
QUARTER( timestamp-expression )
```

パラメータ

timestamp-expression

四半期にする日付。

戻り値

INTEGER

備考

各四半期は次のように定めます。

四半期	期間 (開始日と終了日を含む)
1	1 月 1 日～ 3 月 31 日
2	4 月 1 日～ 6 月 30 日
3	7 月 1 日～ 9 月 30 日
4	10 月 1 日～ 12 月 31 日

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 2 を返します。

```
SELECT QUARTER( '1987/05/02' );
```

1.3.2.163 RADIANS 関数 [数値]

度数をラジアンに変換します。

構文

```
RADIANS( numeric-expression )
```

パラメータ

numeric-expression

度数。この角度をラジアンに変換します。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、約 0.5236 の値を返します。

```
SELECT RADIANS( 30 );
```

1.3.2.164 RAND 関数 [数値]

オプションのシードから、間隔 0 ～ 1 の乱数を返します。

構文

```
RAND( [integer-expression] )
```

パラメータ

integer-expression

乱数の作成に使用するオプションのシード。この引数を使用して、繰り返し可能な乱数列を作成できます。

戻り値

DOUBLE

備考

RAND 関数は、乗算線形合同法で乱数を生成します。詳細については、CACM 31(10) の Park と Miller (1988) の寄稿 (1192 ～ 1201 ページ) と、Press 他 (1992) の『Numerical Recipes in C』(第 2 版の第 7 章 279 ページ、邦訳は『ニューメリカルレシピインシー日本語版 - C 言語による数値計算のレシピ』) を参照してください。RAND 関数の呼び出し結果は、0 < n < 1 の疑似乱数 n です (0.0 と 1.0 はどちらも結果に含まれません)。

サーバへの接続が確立するときに、この乱数生成関数は初期値のシードを指定します。各接続で異なる乱数シーケンスが生成されるように、各接続には一意のシードが指定されます。また、シード値 (integer-expression) を引数に指定することもできます。通常、シード値の指定は、以降の RAND 関数の呼び出しで乱数シーケンスを要求する前に 1 回だけ実行します。複数回シード値を初期化すると、シーケンスは再開されます。同じシード値を指定すると、同じシーケンスが生成されます。値が近いシード値の場合、似た初期シーケンスが生成されますが、シーケンスが進むにつれて相違が多くなります。

適切な乱数結果を取得するためでも、あるシード値から生成されたシーケンスと、別のシード値から生成されたシーケンスとは結合しないでください。つまり、乱数値のシーケンスを生成しているときに、シード値をリセットしないでください。

RAND 関数は非決定的関数として扱われます。クエリオプティマイザは、RAND 関数の結果をキャッシュしません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は偶数の乱数結果を生成します。以降、シードを指定しないで RAND 関数を呼び出すと、毎回異なる結果が生成されます。

```
SELECT RAND( 1 );
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
```

次の文では、シーケンスが同じ 2 つの結果セットが生成されます。これはシード値が 2 度指定されるためです。

```
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
```

次の例では、値が互いに近く、分布の点からは乱数とは言えない 5 つの結果が生成されます。このため、シード値が似た RAND 関数を複数回呼び出すことはお奨めしません。

```
SELECT RAND( 1 ), RAND( 2 ), RAND( 3 ), RAND( 4 ), RAND( 5 );
```

次の例は、5 つのまったく同じ結果が生成されるため、回避する必要があります。

```
SELECT RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 );
```

1.3.2.165 RANK 関数 [ランキング]

値のグループ内でのランクの値を計算します。同位の場合、RANK 関数はランキングシーケンス内にギャップを残します。

構文

```
RANK( ) OVER ( window-spec )
```

```
window-spec : see the Remarks section below
```

戻り値

INTEGER

備考

window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。Window 関数として使用する場合、ORDER BY 句を指定する必要があります。また PARTITION BY 句は指定できますが、ROWS 句や RANGE 句はいずれも指定できません。WINDOW 句の window-spec 定義を参照してください。

標準

ANSI/ISO SQL 標準

RANK は、オプションの ANSI/ISO SQL 言語機能 T612、「Advanced OLAP operations」の一部です。

例

次の例は、ユタとニューヨークの従業員の給与を降順にランキングします。7 番目と 8 番目の従業員は給与が同一であるため、どちらも 7 位にランキングされます。これに続く従業員は 9 位にランキングされ、ランキングシーケンスにギャップが残されます (8 位のランキングはありません)。

```
SELECT Surname, Salary, State,
RANK() OVER (ORDER BY Salary DESC) "Rank"
FROM GROUPO.Employees WHERE State IN ('NY','UT');
```

Surname	Salary	State	Rank
Shishov	72995.000	UT	1
Wang	68400.000	UT	2
Cobb	62000.000	UT	3
Morris	61300.000	UT	4
Davidson	57090.000	NY	5
Martel	55700.000	NY	6
Blaikie	54900.000	NY	7
Diaz	54900.000	NY	7
Driscoll	48023.690	UT	9
Hildebrand	45829.000	UT	10
Whitney	45700.000	NY	11
...	...	...	...
Lynch	24903.000	UT	19

関連情報

WINDOW 句 [1409 ページ]
CUME_DIST 関数 [ランキング] [299 ページ]
DENSE_RANK 関数 [ランキング] [332 ページ]
ROW_NUMBER 関数 [その他] [512 ページ]
PERCENT_RANK 関数 [ランキング] [465 ページ]

1.3.2.166 READ_CLIENT_FILE 関数 [文字列]

クライアントコンピュータ上で指定したファイルからデータを読み込みます。

構文

```
READ_CLIENT_FILE( client-filename-expression )
```

パラメータ

client-filename-expression

クライアントコンピュータ上でのファイル名を示す CHAR 値。パスは、クライアントコンピュータ上で、クライアントアプリケーションの現在の作業フォルダとの相対パスとして解決されます。

戻り値

LONG BINARY

備考

READ_CLIENT_FILE 関数が返す値は、指定されたクライアントファイルの内容を表します。BINARY 式が使用できる構文には、必ずこの関数を使用できます。

データはバイナリ文字列として返されるため、データが別の文字セットの場合、または圧縮または暗号化されている場合は、データの文字セットの変換、解凍、または復号化も必要となることがあります。

READ_CLIENT_FILE の評価中に、データベースサーバは指定されたファイルのクライアントからの転送を開始します。クライアントは、転送要求を受信すると、クライアントファイルの共有ロックを取得し、データベースサーバがクライアントに転送要求の終了を要求するまで、ロックを保持します。

クライアントのソフトウェアライブラリによってファイルの読み込みが行われ、**Command Sequence** (**CmdSeq**) 通信プロトコルを使用してデータの転送が実行されます。クライアント側データの転送は、**Tabular Data Stream** (**TDS**) プロトコルではサポートされていません。

権限

クライアントコンピュータにあるファイルからデータを読み込む場合

- READ CLIENT FILE システム権限が必要です。

- 読み込み元のディレクトリに対する読み込みパーミッションが必要です。
- allow_read_client_file データベースオプションが有効であることが必要です。
- READ_CLIENT_FILE 機能を有効になっている必要があります (-sf サーバオプション)。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

DECOMPRESS 関数 [文字列] [325 ページ]
DECRYPT 関数 [文字列] [327 ページ]
CSCONVERT 関数 [文字列] [297 ページ]

1.3.2.167 READ_SERVER_FILE 関数 [文字列]

サーバ上の指定したファイルからデータを読み込み、ファイルの完全な内容または一部の内容を LONG BINARY 値として返します。

構文

```
READ_SERVER_FILE( filename ) [, start [ , length] ]
```

パラメータ

filename

サーバ上のファイルのパスと名前を示す LONG VARCHAR 値。

start

バイト単位で指定した、読み込むファイルの開始位置。ファイルの先頭バイトの位置を 1 とします。負の開始位置は、先頭からではなく、ファイルの末尾からのバイト数を指定します。

- `start` が指定されていない場合、値 1 が使用されます。
- `start` が 0 で `length` が負でない場合は、値 1 が使用されます。
- `start` が 0 で `length` が負の場合は、値 -1 が使用されます。

length

バイト単位で指定した、読み込むファイルの長さ。

- length が指定されていない場合、関数はファイルの開始位置から末尾まで読み込みます。
- length が正の場合、関数は開始位置から最大 length バイトを読み込みます。
- length が負の場合、関数は開始位置までの最大 length バイトを読み込みます。

戻り値

LONG BINARY

備考

この関数は、指定されたファイルの完全な内容または一部の内容 (start や length が指定されている場合) を LONG BINARY 値として返します。ファイルが存在しない場合、または読み込むことができない場合、NULL が返されます。

filename には、データベースサーバの開始ディレクトリからの相対ファイル名を指定します。

READ_SERVER_FILE 関数では、2GB より大きいファイルの読み込みがサポートされます。ただし、返される内容は 2GB に制限されます。返される内容がこの制限が超えた場合、SQL エラーが返されます。

データファイルの文字セットが異なる場合は、CSCONVERT 関数を使用すると文字セットを変換できます。また、CSCONVERT 関数を使用すると、READ_SERVER_FILE サーバ関数を使用するときに発生する文字セット変換の要件に対処できます。

ディスクサンドボックスが有効になっている場合、filename で参照されるファイルはアクセス可能な場所に存在している必要があります。

権限

クライアントコンピュータにあるファイルからデータを読み込む場合

- READ FILE システム権限が必要です。
- 読み込むディレクトリの読み込みパーミッションが必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ファイル内の 20 バイトを読み込みます (ファイルのバイト 100 以降)。

```
SELECT READ_SERVER_FILE( 'c:¥¥data.txt', 100, 20 )
```

関連情報

xp_read_file システムプロシージャ [1767 ページ]
CSCONVERT 関数 [文字列] [297 ページ]

1.3.2.168 REGEXP_SUBSTR 関数 [文字列]

正規表現を使用して文字列から部分文字列を抽出します。

構文

```
REGEXP_SUBSTR( expression,
regular-expression
[, start-offset [ , occurrence-number [, escape-expression ] ] ] )
```

パラメータ

expression

検索される文字列。
regular-expression

一致させようとするパターン。
start-offset

検索を開始する `expression` へのオフセット。`start-offset` は、正の整数で表され、文字列の左端から数えた文字数を反映します。デフォルトは 1 (文字列の起点) です。
occurrence-number

`expression` 内で複数の一致がある場合に、検出する出現箇所を示す整数を指定します。たとえば、3 を指定すると、3 番目の出現箇所が検出されます。デフォルトは 1 です。
escape-expression

`regular-expression` に使用するエスケープ文字。デフォルトは円記号 (¥) です。

戻り値

LONG VARCHAR

備考

`regular-expression` が検出されない場合、REGEXP_SUBSTR は NULL を返します。

REGEXP 探索条件と同様に、REGEXP_SUBSTR 関数は一致と範囲評価にコードポイントを使用します。データベースの大文字と小文字の区別は結果に影響しません。

部分文字クラスのみを含む文字クラスに一致させる場合は、外側の角カッコと、部分文字クラス用の角カッコを必ず使用してください (たとえば、`REGEXP_SUBSTR (expression, '[[:digit:]]')`)。

`start-offset` が指定されている場合、そのオフセットは、一致させる式の開始を指定します。特に ^ は、`start-offset` で始まる式の先頭と一致します。

標準

ANSI/ISO SQL 標準

標準になし。SQL 標準で対応する関数は SUBSTRING_REGEX 関数であり、パラメータがよく似ています。SUBSTRING_REGEX は、オプションの言語機能 F844 の一部です。

例

次の文では、Employees.Street カラムの値を街路番号と街路名に分割します。

```
SELECT REGEXP_SUBSTR( Street, '^¥S+' ) as street_num,
 REGEXP_SUBSTR( Street, '(?<=^¥S+¥s+).*$' ) AS street_name
    FROM GROUPO.Employees;
```

street_num	street_name
9	East Washington Street
7	Pleasant Street
539	Pond Street
1244	Great Plain Avenue
...	...

現在の接続の IP アドレスが IP アドレスの範囲内 (この場合は、10.25.101.xxx または 10.25.102.xxx) かどうかを確認するには、次の文を実行します。

```
IF REGEXP_SUBSTR( CONNECTION_PROPERTY( 'NodeAddress' ), '¥¥d+¥¥.¥¥d+¥¥.¥¥d+' )
   IN ( '10.25.101' , '10.25.102' ) THEN
      MESSAGE 'In range' TO CLIENT;
ELSE
```

```
      MESSAGE 'Out of range' TO CLIENT;
END IF;
```

関連情報

LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件 [60 ページ]
正規表現の概要 [38 ページ]
正規表現の構文 [39 ページ]
正規表現:特殊部分文字クラス [42 ページ]
REGEXP 探索条件 [67 ページ]

1.3.2.169 REGR_AVGX 関数 [集合]

回帰直線の独立変数の平均を計算します。

構文

式

```
REGR_AVGX( dependent-expression , independent-expression)
```

Window 関数

```
REGR_AVGX( dependent-expression , independent-expression)
OVER( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を除外した後で、次の計算が実行されます。式の x は independent-expression を表します。

```
AVG( x )
```

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_AVGX は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、独立変数である従業員の年齢の平均を計算します。

```
SELECT REGR_AVGX( Salary, ( 2008 - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
AVG 関数 [集合] [244 ページ]

1.3.2.170 REGR_AVGY 関数 [集合]

回帰直線の従属変数の平均を計算します。

構文

式

```
REGR_AVGY( dependent-expression , independent-expression)
```

Window 関数

```
REGR_AVGY( dependent-expression , independent-expression)
OVER( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` と `independent-expression`) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を除外した後で、次の計算が実行されます。式の `y` は `dependent-expression` を表します。

```
AVG( y )
```

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_AVGY は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、独立変数である従業員の給与の平均を計算します。

```
SELECT REGR_AVGY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
AVG 関数 [集合] [244 ページ]

1.3.2.171 REGR_COUNT 関数 [集合]

回帰直線の調整に使用される NULL 以外の数値のペアの数を表す整数を返します。

構文

式

```
REGR_COUNT( dependent-expression , independent-expression)
```

Window 関数

```
REGR_COUNT( dependent-expression , independent-expression)
OVER ( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

INTEGER

備考

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_COUNT は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、回帰直線の調整に使用された NULL 以外のペアの数を返します。

```
SELECT REGR_COUNT( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
COUNT 関数 [集合] [289 ページ]
AVG 関数 [集合] [244 ページ]
SUM 関数 [集合] [554 ページ]

1.3.2.172 REGR_INTERCEPT 関数 [集合]

従属変数と独立変数に最適な線形回帰直線の y 切片を計算します。

構文

式

```
REGR_INTERCEPT( dependent-expression , independent-expression)
```

Window 関数

```
REGR_INTERCEPT( dependent-expression , independent-expression)
OVER( window-spec )
```

window-spec: 以下の備考部分を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` と `independent-expression`) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を排除した後、次の計算が行われます (`y` は `dependent-expression` を表し、`x` は `independent-expression` を表します)。

```
AVG( y ) - REGR_SLOPE( y, x ) * AVG( x )
```

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_INTERCEPT は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、線形回帰直線の y 切片を返します。

```
SELECT REGR_INTERCEPT( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
AVG 関数 [集合] [244 ページ]

1.3.2.173 REGR_R2 関数 [集合]

回帰直線の決定係数 (*R-squared* または適合度の統計情報とも呼びます) を計算します。

構文

式

```
REGR_R2( dependent-expression , independent-expression)
```

Window 関数

```
REGR_R2( dependent-expression , independent-expression)
OVER( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアのセットに適用されます。

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_R2 は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、回帰直線の決定係数を返します。

```
SELECT REGR_R2( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]

1.3.2.174 REGR_SLOPE 関数 [集合]

NULL 以外のペアに調整された線形回帰直線の傾きを計算します。

構文

式

```
REGR_SLOPE( dependent-expression , independent-expression)
```

Window 関数

```
REGR_SLOPE( dependent-expression , independent-expression)
OVER ( window-spec )
```

window-spec: 以下の備考部分を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` と `independent-expression`) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を排除した後、次の計算が行われます (`y` は `dependent-expression` を表し、`x` は `independent-expression` を表します)。

```
COVAR_POP( y, x ) / VAR_POP( x )
```

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_SLOPE は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 935.3429749445614 を返します。

```
SELECT REGR_SLOPE( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
COVAR_POP 関数 [集合] [294 ページ]
VAR_POP 関数 [集合] [586 ページ]

1.3.2.175 REGR_SXX 関数 [集合]

線形回帰モデルに使用される独立した式の平方値の合計を返します。REGR_SXX 関数は、回帰モデルの統計的な有効性を評価するときに使用できます。

構文

式

```
REGR_SXX( dependent-expression , independent-expression)
```

Window 関数

```
REGR_SXX( dependent-expression , independent-expression)
OVER ( window-spec )
```

```
window-spec: 次の「備考」の項を参照してください。
```

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を表し、x は independent-expression を表します)。

```
REGR_COUNT( y, x ) * VAR_POP( x )
```

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_SXX は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 5916.4800000000105 を返します。

```
SELECT REGR_SXX( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
VAR_POP 関数 [集合] [586 ページ]

1.3.2.176 REGR_SXY 関数 [集合]

従属変数と独立変数の積和を返します。REGR_SXY 関数は、回帰モデルの統計的な有効性を評価するときに使用できます。

構文

式

```
REGR_SXY( dependent-expression , independent-expression)
```

Window 関数

```
REGR_SXY( dependent-expression , independent-expression)
OVER( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行い、結果を DOUBLE で返します。関数が空のセットに適用される場合は、NULL を返します。

関数は、`dependent-expression` または `independent-expression` が NULL のペアをすべて排除した後、(`dependent-expression` と `independent-expression`) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を排除した後、次の計算が行われます (`y` は `dependent-expression` を表し、`x` は `independent-expression` を表します)。

```
REGR_COUNT( y, x ) * COVAR_POP( y, x )
```

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_SXY は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、従属変数と独立変数の積和を返します。

```
SELECT REGR_SXY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]

1.3.2.177 REGR_SYY 関数 [集合]

回帰モデルの統計的有効性を評価できる値を返します。

構文

式

```
REGR_SYY( dependent-expression , independent-expression)
```

Window 関数

```
REGR_SYY( dependent-expression , independent-expression)
OVER( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

dependent-expression

独立変数に影響される変数。

independent-expression

結果に影響する変数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。関数が空のセットに適用される場合は、NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアのセットに適用されます。関数は、データの 1 回のパススルー中に同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を表し、x は independent-expression を表します)。

```
REGR_COUNT( y, x ) * VAR_POP( y )
```

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

REGR_SYY は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 26、708、672,843.3002 を返します。

```
SELECT REGR_SYY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]

1.3.2.178 REMAINDER 関数 [数値]

整数を整数で割ったときの余りを返します。

構文

```
REMAINDER( dividend, divisor )
```

パラメータ

dividend

被除数 (分数の分子)。

divisor

除数 (分数の分母)。

戻り値

- INTEGER
- NUMERIC

備考

MOD 関数を使用して、余りを返すこともできます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 2 を返します。

```
SELECT REMAINDER( 5, 3 );
```

関連情報

MOD 関数 [数値] [441 ページ]

1.3.2.179 REPEAT 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

```
REPEAT( string-expression, integer-expression )
```

パラメータ

string-expression

繰り返される文字列。

integer-expression

文字列を繰り返す回数。`integer-expression` が正数でない場合は、空の文字列を返します。

戻り値

LONG VARCHAR
LONG NVARCHAR
Ultra Light の場合、LONG VARCHAR が返されます。

備考

実際の結果文字列の長さが戻り値の最大長を超えると、エラーが発生します。この場合、結果は文字列に使用できる最大サイズまでトランケートされます。

この関数の動作は、REPLICATE 関数と同じです。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 repeatrepeatrepeat を返します。

```
SELECT REPEAT( 'repeat', 3 );
```

関連情報

文字列関数 [213 ページ]
REPLICATE 関数 [文字列] [504 ページ]

1.3.2.180 REPLACE 関数 [文字列]

文字列を別の文字列で置換し、新しい結果を返します。

構文

```
REPLACE( original-string, search-string, replace-string )
```

パラメータ

いずれかの引数が NULL の場合、NULL を返します。

original-string

検索される文字列。任意の長さの文字列を指定できます。

search-string

検索して replace-string に置き換えられる文字列。この文字列は 255 バイトに制限されています。search-string が空の文字列の場合は、元の文字列がそのまま返されます。

replace-string

置換文字列。search-string を置き換えます。任意の長さの文字列を指定できます。replace-string が空の文字列の場合は、検索されたすべての search-string が削除されます。

戻り値

LONG BINARY
LONG VARCHAR

LONG NVARCHAR
Ultra Light は NVARCHAR を返しません。

備考

この関数は、すべての search-string を replace-string に置き換えます。

すべての引数がバイナリデータ型の場合、REPLACE 関数の動作は BYTE_REPLACE と同じになります。

大文字と小文字を区別するデータベースでの比較では、大文字と小文字が区別されます。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 `xx.def.xx.ghi` を返します。

```
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' );
```

次の文は、ALTER PROCEDURE 文を含む結果セットを生成します。ALTER PROCEDURE を実行すると、名前が変更されたテーブルを参照する格納済みプロシージャが修復されます (テーブル名を一意にすることをおすすめします)。

```
SELECT REPLACE(
   REPLACE( proc_defn, 'OldTableName', 'NewTableName' ),
   'CREATE PROCEDURE',
   'ALTER PROCEDURE')
FROM SYS.SYSPROCEDURE
WHERE proc_defn LIKE '%OldTableName%';
```

関連情報

文字列関数 [213 ページ]
SUBSTRING 関数 [文字列] [551 ページ]
CHARINDEX 関数 [文字列] [270 ページ]

1.3.2.181 REPLICATE 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

```
REPLICATE( string-expression, integer-expression )
```

パラメータ

string-expression

繰り返される文字列。

integer-expression

文字列を繰り返す回数。

戻り値

LONG VARCHAR
LONG NVARCHAR
Ultra Light は NVARCHAR を返しません。

備考

実際の結果文字列の長さが戻り値の最大長を超えると、エラーが発生します。この場合、結果は文字列に使用できる最大サイズまでトランケートされます。

この関数の動作は、REPEAT 関数と同じです。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 repeatrepeatrepeat を返します。

```
SELECT REPLICATE( 'repeat', 3 );
```

関連情報

文字列関数 [213 ページ]
REPEAT 関数 [文字列] [501 ページ]

1.3.2.182 REVERSE 関数 [文字列]

文字式の逆の値を返します。

構文

```
REVERSE( string-expression )
```

パラメータ

string-expression

逆転される文字列。

戻り値

- LONG VARCHAR
- LONG NVARCHAR

備考

この関数は NCHAR の入力または出力をサポートしています。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 cba を返します。

```
SELECT REVERSE( 'abc' );
```

関連情報

文字列関数 [213 ページ]

1.3.2.183 REWRITE 関数 [その他]

書き換えられた SELECT 文、UPDATE 文または DELETE 文を返します。

構文

```
REWRITE( select-statement [, 'ANSI' ] )
```

パラメータ

select-statement

関数の結果を生成するために書き換えの最適化を適用する SQL 文。

戻り値

LONG VARCHAR

備考

ANSI 引数を指定せずに REWRITE 関数を使用すると、オプティマイザがどのように特定のクエリのアクセスプランを生成したかを理解できます。特に、文の WHERE 句、ON 句、HAVING 句の条件がデータベースサーバによってどのように書き換えられたかを確認して、要求の実行時間を改善するために適用できるインデックスの有無を特定することができます。

REWRITE から返された文は、元の文のセマンティックと一致しない場合があります。これは、一部のリライト最適化では、SQL に直接変換できない内部メカニズムが導入されるためです。たとえば、サーバではローの識別子を使用して重複を排除しますが、これは SQL に変換できません。

REWRITE 関数によって書き換えられたクエリは、実行するためのクエリではありません。リライトフェーズの後に実際にオプティマイザに何が渡されるかを示すことで、パフォーマンス問題を分析するためのツールです。

REWRITE の出力に反映されないリライト最適化もあります。それらは、LIKE 最適化、minimum 関数または maximum 関数の最適化、上限／下限の排除、述部の仮定条件などです。

ANSI を指定すると、REWRITE は文に相当する ANSI を返します。この場合は、次のリライト最適化のみが適用されます。

- Transact-SQL 外部ジョインが、ANSI SQL 外部ジョインに書き換えられる
- 重複した相関名が削除される
- KEY ジョインと NATURAL ジョインが ANSI SQL ジョインに書き換えられる

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、クエリでリライト最適化を 2 回実行します。最初の最適化では、サブクエリのネストを解除して、Employees テーブルと SalesOrders テーブルをジョインします。2 回目の最適化では、Employees と SalesOrders の間のプライマリキーと外部キーのジョインを削除してクエリを簡素化します。このリライト最適化の一部では、ジョインの述部 e.EmployeeID=s.SalesRepresentative が述部 s.SalesRepresentative IS NOT NULL に置換されます。

```
SELECT REWRITE( 'SELECT s.ID, s.OrderDate
    FROM GROUPO.SalesOrders s
    WHERE EXISTS ( SELECT *
       FROM GROUPO.Employees e
          WHERE e.EmployeeID = s.SalesRepresentative)' ) FROM SYS.DUMMY;
```

このクエリは、ラインセパレータを使用して書き換えられたクエリがある単一カラムの結果セットを返します。

```
'select s.ID,s.OrderDate
  from GROUPO.SalesOrders as s'
```

クエリのセマンティック分析を行うと、WHERE 句の削除が可能になります (SalesRepresentative の Employees テーブルには FOREIGN KEY 制約があります)。

次の REWRITE 文は ANSI 引数を使用します。このクエリでは、tsql_outer_joins オプションを設定する必要があります。

```
SET TEMPORARY OPTION tsql_outer_joins = 'On';
```

```
SELECT REWRITE( 'SELECT DISTINCT s.ID, s.OrderDate, e.GivenName, e.EmployeeID
    FROM GROUPO.SalesOrders s, GROUPO.Employees e
          WHERE e.EmployeeID *= s.SalesRepresentative', 'ANSI' ) FROM SYS.DUMMY;
```

結果は、この文に相当する ANSI です。この場合は、Transact-SQL 外部ジョインが ANSI 外部ジョインに変換されます。このクエリは、ラインセパレータを使用して書き換えられた文がある単一カラムの結果セットを返します。

```
'select distinct s.ID,s.OrderDate,e.GivenName,e.EmployeeID
  from GROUPO.Employees as e left outer join GROUPO.SalesOrders as s
    on e.EmployeeID = s.SalesRepresentative'
```

1.3.2.184 RIGHT 関数 [文字列]

文字列の一番右側にある文字を返します。

構文

```
RIGHT( string-expression, integer-expression )
```

パラメータ

string-expression

一番右側にある文字を返す文字列。

integer-expression

返される文字列の末尾の文字数。

戻り値

LONG VARCHAR
LONG NVARCHAR
Ultra Light は NVARCHAR を返しません。

備考

文字列にマルチバイト文字が含まれている場合は、返されるバイト数が指定した文字数よりも大きい場合があります。

カラムの値より大きな `integer-expression` を指定できます。この場合、全体の値が返されます。

この関数は NCHAR の入力または出力をサポートしています。入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Customers テーブルに含まれる各 Surname 値の最後の 5 文字を返します。

```
SELECT RIGHT( Surname, 5 ) FROM GROUPO.Customers;
```

関連情報

文字列関数 [213 ページ]
LEFT 関数 [文字列] [416 ページ]

1.3.2.185 ROUND 関数 [数値]

`numeric-expression` を `integer-expression` で指定した小数点以下の桁数に丸めます。

構文

```
ROUND( numeric-expression, integer-expression )
```

パラメータ

numeric-expression

関数に渡される、丸めの対象となる数。

integer-expression

正の整数は、丸めを行う小数点の右側の有効桁数を指定します。負の式は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値

NUMERIC

備考

この関数の結果は numeric または double です。数値の結果があり、整数 `integer-expression` が負の値の場合、精度は 1 ずつ増えます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 123.200 を返します。

```
SELECT ROUND( 123.234, 1 );
```

関連情報

TRUNCNUM 関数 [数値] [574 ページ]

1.3.2.186 ROW コンストラクタ [複合]

ペア (`field namedata type`, ...) (名前は **fields**) のシーケンスを返します。

構文

```
ROW(
expression [, expression ... ]
| single-row-query-expression
)
```

パラメータ

expression

単一のフィールドを表す式。

single-row-query-expression

単一のフィールドを返すクエリ文。

戻り値

フィールド値

備考

すべてのフィールドが NULL に初期化され、特定のフィールド内に値が明示的または暗黙的に配置されるまで NULL のままとなります。

ビュー定義の最も外側の SELECT リスト、あるいはクライアントに返されたトップレベルの SELECT ブロックまたはクエリ式で、ROW 型を指定することはできません。ベーステーブルまたはテンポラリテーブルで ROW 型をカラムとして保存することはできません。

ROW 型には、複雑な構造型となるネストの任意のレベルを含めることができます。各行のサブタイプには名前が付けられ、それらの名前は特定の値を参照する dot 表記を使用して参照できます。

dot 表記により、同じクエリブロックまたはネストされているクエリブロックの他の式を使用して、ROW 型のすべてまたは一部を参照して他の RAW 型の比較または初期化を行うことができます。

修飾名を持つカラム内のフィールドを参照する場合は、カラム名を括弧で囲みます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Products テーブルの各製品の製品情報の構造体を含む ROW 型の構成方法を示します。

```
SELECT ROW( ID, NAME, DESCRIPTION ).id AS pInfo FROM GROUPO.Products;
```

pInfo ROW 型には、ID、NAME、DESCRIPTION という名前の要素があります。これらは Products テーブルの属性名から借りたものです。

次の文は、CAST 関数を使用して明示的な名前を暗黙的な ROW 型に割り当て、キャストされたローから ProductID フィールドを抽出する方法を示しています。

```
SELECT CAST( ROW( 303, 'Tee Shirt', 'My tee shirt' )
    AS ROW(ProductID INTEGER,
        ProductName CHAR(25),
        ProductDescription CHAR(35)  )
    ).ProductID
    AS pInfo FROM SYS.DUMMY;
```

このクエリの結果を入力として使用する SQL 式は、dot 式を pInfo 行の名前で使用して、pInfo 行のコンポーネントを参照できます。ROW 型は、シングルロークエリ式を使用して構成することもできます。

次の文は、ROW を構築し、そこから Name フィールドを抽出する別の方法を示しています。

```
SELECT ROW( SELECT Name, Quantity FROM Products WHERE ID = 300 ).Name;
```

次の文は、ROW、student、でフィールド last_name を名前別に設定する方法を示します。

```
CREATE VARIABLE student ROW(first_name LONG VARCHAR, last_name LONG VARCHAR);
SET student.last_name = 'Johnson';
```

次の例は、ROW 型の中でネストを使用する方法を示します。

```
DECLARE sample ROW( x INT, w ROW( y INT, z INT ) );
SET Sample = ROW( 3, ROW( 6,7 ) );
SELECT (Sample).w.y FROM dummy;
```

次の例は、修飾名を持つカラム内のフィールドを参照する方法を示します。

```
SELECT ( myderivedtable.myrowcolumn ).id FROM
 ( SELECT ROW( id, name )
         FROM GROUPO.Products )
  AS myderivedtable( myrowcolumn );
```

関連情報

複合データ型 ROW および ARRAY [178 ページ]
複合型の比較 [190 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.3.2.187 ROW_NUMBER 関数 [その他]

各ローにユニークな番号を割り当てます。この関数は NUMBER 関数の代わりに使用できます。

構文

ROW_NUMBER( ) OVER (`window-spec` *)*

`window-spec`: 以下の備考部分を参照してください。

戻り値

INTEGER

備考

window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。Window 関数として使用する場合、ORDER BY 句を指定する必要があります。また PARTITION BY 句は指定できますが、ROWS 句や RANGE 句はいずれも指定できません。WINDOW 句の window-spec 定義を参照してください。

標準

ANSI/ISO SQL 標準

ROW_NUMBER は、オプションの ANSI/ISO SQL 言語機能 T611、「Elementary OLAP operations」の一部です。

例

次の文は、ニューヨークとユタの各従業員のユニークなロー番号を示す結果セットを返します。クエリは Salary の降順に配列されるため、最初のロー番号は、データセット内で最も給与の高い従業員に割り当てられます。2 人の従業員の給与が同一ですが、2 人の従業員にはユニークなロー番号が割り当てられるため、同順は解決されません。

```
SELECT Surname, Salary, State,
ROW_NUMBER() OVER (ORDER BY Salary DESC) "Rank"
FROM GROUPO.Employees WHERE State IN ('NY','UT');
```

Surname	Salary	State	Rank
Shishov	72995.000	UT	1
Wang	68400.000	UT	2
Cobb	62000.000	UT	3
Morris	61300.000	UT	4
Davidson	57090.000	NY	5
Martel	55700.000	NY	6
Blaikie	54900.000	NY	7
Diaz	54900.000	NY	8
Driscoll	48023.690	UT	9
Hildebrand	45829.000	UT	10
...	...	...	...
Lynch	24903.000	UT	19

関連情報

WINDOW 句 [1409 ページ]
NUMBER 関数 [その他] [461 ページ]

RANK 関数 [ランキング] [479 ページ]
ROWID 関数 [その他] [514 ページ]

1.3.2.188 ROWID 関数 [その他]

テーブル内のローを一意に識別する UNSIGNED BIGINT ビット値を返します。

構文

```
ROWID( correlation-name )
```

パラメータ

correlation-name

クエリで使用されるテーブルの相関名。相関名は、ベーステーブル、テンポラリテーブル、グローバルテンポラリテーブル、またはプロキシテーブル (基本となるプロキシサーバが同様の機能をサポートしている場合にのみ使用できます) を指す必要があります。ROWID 関数の引数では、ビュー、派生テーブル、共通のテーブル式、またはプロシージャを参照しないでください。

戻り値

UNSIGNED BIGINT

備考

指定された相関名に対応するテーブル内のローの、ロー識別子を返します。

この関数によって返される値は、複数のクエリ間で一貫しているとはかぎりません。データベースで実行されるさまざまな演算の結果、テーブルのロー識別子が変更される可能性があるためです。特に、REORGANIZE TABLE 文を実行すると、ロー識別子が変更される可能性があります。また、ローが削除された後にロー識別子が再利用される場合もあります。そのため、通常は ROWID 関数の使用を避け、代わりにプライマリキー値に基づいてロー識別子を取得してください。ROWID は診断を実行する場合にのみ使用することをお奨めします。

関数の結果は UNSIGNED BIGINT ですが、この値について数学演算を行っても意味がありません。たとえば、ロー識別子に 1 を追加しても、次のローのロー識別子にはなりません。また、ROWID を使用する場合、等号と IN 述部のみを検索引数にすることができます。必要に応じて、ROWID を使用する述部 (`ROWID( T ) = literal` など) は 64 ビット UNSIGNED INTEGER 値へのキャストに使用できます。変換を実行できない場合、データの例外処理が発生します。`literal` の値が無効なロー識別子の場合、比較述部は FALSE と評価されます。

ROWID 関数をテーブルまたはカラムの検査制約内で使用することはできません。また、計算カラムの COMPUTE 式でも使用できません。

OPENSTRING 関数と同時に使用すると、ROWID 値は入力文字列内のロー番号になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Employee のローから、105 のロー識別子を返します。

```
SELECT ROWID( Employees ) FROM GROUPO.Employees WHERE Employees.EmployeeID = 105;
```

次の文は、Employees テーブルのローのロックリストとそのローの内容を返します。

```
SELECT *
  FROM sa_locks() S JOIN GROUPO.Employees WITH( NOLOCK )
     ON ROWID( Employees ) = S.row_identifier
  WHERE S.table_name = 'Employees';
```

関連情報

ROW_NUMBER 関数 [その他] [512 ページ]

1.3.2.189 RTRIM 関数 [文字列]

文字列の後続ブランクを削除します。

構文

```
RTRIM( string-expression )
```

パラメータ

string-expression

削除する文字列。

戻り値

VARCHAR
NVARCHAR
LONG VARCHAR
LONG NVARCHAR
Ultra Light の場合、VARCHAR および LONG VARCHAR

備考

結果の実際の長さは、式の長さから、削除する文字数を引いた値です。すべての文字を削除すると、結果は空の文字列になります。

引数が NULL の場合、結果は NULL 値になります。

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

ANSI/ISO SQL 標準 (LEADING および TRAILING) により定義された TRIM 仕様は、それぞれ LTRIM 関数と RTRIM 関数 (ソフトウェアに用意されています) により指定されます。

例

次の文は、すべての後続ブランクが削除された文字列 `Test Message` を返します。

```
SELECT RTRIM( 'Test Message     ' );
```

関連情報

文字列関数 [213 ページ]
TRIM 関数 [文字列] [572 ページ]
LTRIM 関数 [文字列] [428 ページ]

1.3.2.190 SECOND 関数 [日付と時刻]

TIMESTAMP 引数の秒の値を返します。

構文

```
SECOND( timestamp-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

SMALLINT

備考

指定した TIMESTAMP 引数値の秒に相当する 0 ～ 59 の数を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 25 を返します。

```
SELECT SECOND( '1998-07-13 21:21:25' );
```

1.3.2.191 SECONDS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の秒境界の数を返します。

構文

0000-02-29 の真夜中と TIMESTAMP 値の間の月数を返します。

```
SECONDS( timestamp-expression )
```

2 つの TIMESTAMP 値の間の秒数を返します。

```
SECONDS( timestamp-expression, timestamp-expression )
```

秒を TIMESTAMP 値に追加します。

```
SECONDS( time-or-timestamp-expression, integer-expression )
```

パラメータ

timestamp-expression

TIMESTAMP 値。

time-or-timestamp-expression

TIME または TIMESTAMP データ型の値。

integer-expression

`time-or-timestamp-expression` に追加する秒数。`integer-expression` が負の場合、適切な秒数が `time-or-timestamp-expression` から減算されます。integer-expression を指定する場合は、`time-or-timestamp-expression` を TIME、DATE または TIMESTAMP データ型として明示的にキャストしてください。`time-or-timestamp-expression` が DATE データ型の場合、時刻部分には真夜中が使用されます。

戻り値

0000-02-29 の真夜中と TIMESTAMP 値の間の月数を返す場合は UNSIGNED BIGINT

2 つの TIMESTAMP 値の間の秒数を返す場合は SIGNED BIGINT

TIMESTAMP 値に秒を追加する場合は TIME または TIMESTAMP

備考

SECONDS 関数の結果は、その引数によって異なります。

0000-02-29 の真夜中と TIMESTAMP 値の間の月数を返します。

1 つの timestamp-expression を SECONDS 関数に渡すと、0000-02-29 の真夜中から timestamp-expression までの秒数を UNSIGNED BIGINT として返します。

> **注記**
>
> 0000-02 は実際の日付を指すための値ではありません。SECONDS 関数で使用されるデフォルトの日付です。

2 つの TIMESTAMP 値の間の秒数を返します。

2 つの TIMESTAMP 値を SECONDS 関数に渡すと、その間の秒境界の数を整数の SIGNED BIGINT 値で返します。秒を **TIMESTAMP** 値に追加します。

TIMESTAMP 値と INTEGER 値を SECONDS 関数に渡すと、整数の秒数を time-or-timestamp-expression に加算した TIMESTAMP 結果を返します。同様に、TIME 値を SECONDS 関数に渡すと、TIME データ型の値を返します。

DATEDIFF 関数と DATEADD 関数を使用すると、これらの計算の一部を実行できます。

標準

ANSI/ISO SQL 標準

標準になし。

> **例**
>
> 次の文は、同一の値 14400 を返します。これは、2 番目の TIMESTAMP 値が、最初のタイムスタンプの 14400 秒後であることを示します。

```
SELECT
  SECONDS( '1999-07-13 06:07:12',
           '1999-07-13 10:07:12' ),
  DATEDIFF( second,
           '1999-07-13 06:07:12',
           '1999-07-13 10:07:12' );
```

次の文は、値 63062431632 を返します。

```
SELECT SECONDS( '1998-07-13 06:07:12' );
```

次の文は TIMESTAMP 値 1999/05/12 21:05:12.000 を返します。

```
SELECT SECONDS( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5);
SELECT DATEADD( second, 5, '1999-05-12 21:05:07' );
```

関連情報

CAST 関数 [データ型変換] [264 ページ]

DATEADD 関数 [日付と時刻] [304 ページ]
DATEDIFF 関数 [日付と時刻] [305 ページ]

1.3.2.192 SECURE_SIGN_MESSAGE 関数 [文字列]

メッセージをデジタル署名します。

構文

```
SECURE_SIGN_MESSAGE(
message
, key
[, hash-algorithm ]
)
```

```
hash-algorithm:
'SHA-1'
| 'SHA-256'
| 'MD5'
```

パラメータ

message 署名するメッセージを指定する LONG BINARY 値。
key 使用するキー (PEM 形式)。メッセージは、プライベートキーで署名され、パブリックキーで検証されます。
hash-algorithm (オプション)。署名プロセスで使用されるハッシュアルゴリズムを指定します。デフォルトは 'SHA-1' です。

戻り値

LONG BINARY

備考

この関数は、メッセージの暗号化ハッシュを作成し、送信者のプライベートキーを使用してハッシュを暗号化します。

例

1. RSA キーペアを作成してパブリックキーを公開し、メッセージの受信者がアクセスできるようにします。
2. メッセージの送信者は、次の文を実行してシグニチャを作成し、シグニチャを使用してメッセージを署名します。

```
CREATE VARIABLE @signature LONG BINARY;
```

```
SELECT SECURE_SIGN_MESSAGE( document, private-key ) INTO @signature;
```

3. メッセージの送信者は、メッセージの受信者に、シグニチャとともにドキュメントを送信します。
4. メッセージの受信者は、次の文を実行し、メッセージが認証されたことと、送信されてから変更されていないことを確認します。

```
CREATE VARIABLE @verified INTEGER;
SELECT SECURE_VERIFY_MESSAGE( document, signature, senders-public-key ) INTO
@verified;
```

@verified 変数が 1 と等しい場合、メッセージは認証されています。

1.3.2.193 SECURE_VERIFY_MESSAGE 関数 [文字列]

メッセージをデジタル検証します。

構文

```
SECURE_VERIFY_MESSAGE(
string-expression
, signature
, key
[, hash-algorithm ]
)
```

```
hash-algorithm:
'SHA-1'
| 'SHA-256'
| 'MD5'
```

パラメータ

string-expression 検証するメッセージ。

signature 検証するシグニチャ。

key 使用するキー (PEM 形式)。メッセージは、プライベートキーで署名され、パブリックキーで検証されます。

hash-algorithm (オプション)。検証プロセス中に使用されるハッシュアルゴリズムを指定します。デフォルトは 'SHA-1' です。

戻り値

メッセージとシグニチャが正常に検証された場合は 1、それ以外の場合は 0 を返します。

備考

この関数は、送信者のキーを使用してシグニチャを復号化することによりデジタル署名を検証し、メッセージのハッシュを計算して 2 つのハッシュを比較します。2 つのハッシュが一致する場合、受信者は、メッセージが送信者のプライベートキーにアクセスできるユーザか送信されたことと、メッセージが変更されていないことを確認できます。

例

1. RSA キーペアを作成してパブリックキーを公開し、メッセージの受信者がアクセスできるようにします。
2. メッセージの送信者は、次の文を実行して署名を作成し、シグニチャを使用してメッセージを署名します。

   ```
   CREATE VARIABLE @signature LONG BINARY;
   SELECT SECURE_SIGN_MESSAGE( document, private-key ) INTO @signature;
   ```
3. メッセージの送信者は、メッセージの受信者に、シグニチャとともにドキュメントを送信します。
4. メッセージの受信者は、次の文を実行し、メッセージが認証されたことと、送信されてから変更されていないことを確認します。

   ```
   CREATE VARIABLE @verified INTEGER;
   SELECT SECURE_VERIFY_MESSAGE( document, signature, senders-public-key ) INTO
   @verified;
   ```

   @verified 変数が 1 と等しい場合、メッセージは認証されています。

1.3.2.194 SET_BIT 関数 [ビット配列]

ビット配列の特定ビットの値を設定します。

構文

```
SET_BIT( [ bit-expression, ]bit-position [, value ] )
```

パラメータ

bit-expression

ビットを変更するビット配列。

bit-position

設定するビットの位置。これは符号なしの整数にしてください。

value

ビットに設定する値。

戻り値

LONG VARBIT

備考

`bit-expression` のデフォルト値は、すべてのビットが 0 (FALSE) に設定された長さ `bit-position` のビット配列です。

`value` のデフォルト値は 1 (TRUE) です。

いずれかのパラメータが NULL の場合、結果は NULL です。

配列の位置は左側からカウントします。初期値は 1 です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 00100011 を返します。

```
SELECT SET_BIT( '00110011', 4 , 0);
```

次の文は、値 00111011 を返します。

```
SELECT SET_BIT( '00110011', 5 , 1);
```

次の文は、値 00111011 を返します。

```
SELECT SET_BIT( '00110011', 5 );
```

次の文は、値 00001 を返します。

```
SELECT SET_BIT( 5 );
```

関連情報

GET_BIT 関数 [ビット配列] [372 ページ]
SET_BITS 関数 [集合] [524 ページ]
INTEGER データ型 [144 ページ]
ビット処理演算子 [31 ページ]

sa_get_bits システムプロシージャ [1510 ページ]

1.3.2.195 SET_BITS 関数 [集合]

ローのセットに含まれる値に対応する特定ビットを 1 (TRUE) に設定するときに、ビット配列を作成します。

構文

```
SET_BITS( expression )
```

パラメータ

expression

1 に設定するビットを決定するときに使用する式。これは一般的にカラム名です。

戻り値

LONG VARBIT

備考

指定した値が NULL のローは無視されます。

ローがない場合、NULL が返されます。

結果の長さは、1 に設定された最大の位置です。

SET_BITS 関数も次の文と同様ですが、より高速です。

```
SELECT BIT_OR( SET_BIT( expression ) )
FROM table;
```

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、2 番目、5 番目、10 番目の各ビットが 1 に設定されたビット配列 (つまり 0100100001) を返します。

```
CREATE TABLE t( r INTEGER );
INSERT INTO t values( 2 );
INSERT INTO t values( 5 );
INSERT INTO t values(10 );
SELECT SET_BITS( r ) FROM t;
```

関連情報

ビット処理演算子 [31 ページ]
GET_BIT 関数 [ビット配列] [372 ページ]
SET_BIT 関数 [ビット配列] [522 ページ]
sa_get_bits システムプロシージャ [1510 ページ]

1.3.2.196 SIGN 関数 [数値]

指定された数値の符号 (正または負) を返します。

構文

```
SIGN( numeric-expression )
```

パラメータ

numeric-expression

符号を返す数値。`numeric-expression` の型は、INTEGER、DOUBLE、または NUMERIC です。

戻り値

SMALLINT

備考

負の数を指定すると、SIGN 関数は -1 を返します。

0 を指定すると、SIGN 関数は 0 を返します。

正の数を指定すると、SIGN 関数は 1 を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 -1 を返します。

```
SELECT SIGN( -550 );
```

1.3.2.197 SIMILAR 関数 [文字列]

2 つの文字列の類似性を示す数を返します。

構文

```
SIMILAR( string-expression-1, string-expression-2 )
```

パラメータ

string-expression-1

比較する最初の文字列。

string-expression-2

比較する 2 番目の文字列。

戻り値

SMALLINT

備考

SIMILAR 関数は、2 つの文字列の類似性を表す 0 ～ 100 の整数を返します。結果は、2 つの文字列の文字が一致している割合として解釈できます。値が 100 の場合は、2 つの文字列は同じです。

この関数を使用して、名前 (顧客名など) のリストを修正できます。顧客がわずかに異なる名前で重複して登録されている場合があります。SIMILAR 関数を使用して、類似する顧客名を見つけることができます。このことを行うには、customer テーブルをそのテーブル自体にジョインし、類似性が 90 パーセントより大きく、かつ 100 パーセント未満のすべての顧客のレポートを作成します。

SIMILAR 関数で実行される計算は、単に文字数が一致する場合よりも複雑になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は値 75 を返します。2 つの値が 75 % 前後であることを示します。

```
SELECT SIMILAR( 'toast', 'coast' );
```

関連情報

文字列関数 [213 ページ]

1.3.2.198 SIN 関数 [数値]

数のサインを返します。

構文

```
SIN( numeric-expression )
```

パラメータ

numeric-expression

角度 (ラジアン)。

戻り値

DOUBLE

備考

SIN 関数は、引数のサインを返します。この引数はラジアン単位で表現される角度です。SIN 関数と ASIN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、0.52 の SIN 値を返します。

```
SELECT SIN( 0.52 );
```

関連情報

ASIN 関数 [数値] [240 ページ]
COS 関数 [数値] [287 ページ]
COT 関数 [数値] [288 ページ]
TAN 関数 [数値] [560 ページ]

1.3.2.199 SOAP_HEADER 関数 [SOAP]

SOAP ヘッダエントリまたは SOAP 要求のヘッダエントリの属性値を返します。

構文

```
SOAP_HEADER( header-key [, index, header-attribute ] )
```

パラメータ

header-key

VARCHAR パラメータには、特定の SOAP ヘッダエントリで最上位層にある XML 要素の XML ローカル名を指定します。

index

このオプションの INTEGER パラメータは、同じ名前の SOAP ヘッダフィールドでも違いを見つけます。同じ名前になる状況は、複数のヘッダエントリが同じ localname を持つ最上位 XML 要素がある場合に発生します。通常、このような要素には一意の番号空間があります。

header-attribute

オプションの VARCHAR パラメータは、次に示すヘッダエントリ要素内にある任意の属性ノードを指定できます。

@namespace

特定のヘッダエントリの名前空間にアクセスするときに使用する特殊な SQL Anywhere 属性。

mustUnderstand

ヘッダエントリが必須かオプションかを処理する受信者に示す SOAP 1.1 のヘッダエントリ属性。

encodingStyle

エンコーディングスタイルを示す SOAP 1.1 ヘッダエントリ属性。この属性にはアクセスできますが、SQL Anywhere の内部では使用されません。

actor

受信者の URL を指定することで、ヘッダエントリの受信者を示す SOAP 1.1 ヘッダエントリ属性。

戻り値

LONG VARCHAR

備考

この関数は、単一のパラメータ header-key を使用してヘッダエントリを返すときに使用します。ヘッダエントリは、SOAP ヘッダに含まれている要素と、その要素に含まれるすべての部分要素を XML 文字列で表現したものです。

この関数は、オプションの index と header-attribute パラメータを指定して、ヘッダエントリ属性を抽出するときにも使用できます。

この関数は、指定された SOAP ヘッダフィールドの値を返します。SOAP サービスから呼び出されていない場合は NULL を返します。Web サービスを介して SOAP 要求を処理する場合に使用します。

指定した header-key のヘッダが存在しない場合、戻り値は NULL です。

標準

ANSI/ISO SQL 標準

標準になし。

例

HTTP Web サービスによって呼び出されるストアドプロシージャ内で使用された場合、次の例は SOAP 要求ヘッダにあるすべてのキーを処理します。認証キーを処理するときに、キーの値も取得します。

```
BEGIN
  DECLARE hd_key LONG VARCHAR;
  DECLARE hd_entry LONG VARCHAR;
header_loop:
  LOOP
    SET hd_key = NEXT_SOAP_HEADER( hd_key );
    IF hd_key IS NULL THEN
      -- no more header entries
      LEAVE header_loop;
    END IF;
    IF hd_key = 'Authentication' THEN
      SET hd_entry = SOAP_HEADER( hd_key );
    END IF;
  END LOOP header_loop;
END;
```

関連情報

Web サービス関数 [212 ページ]
NEXT_SOAP_HEADER 関数 [SOAP] [457 ページ]
sa_set_soap_header システムプロシージャ [1625 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.3.2.200 SORTKEY 関数 [文字列]

ソートキー値を生成します。つまり、代替照合規則に基づいて文字列をソートする場合に使用できる値を生成します。

構文

```
SORTKEY( string-expression
[, { collation-id
| collation-name[ ( collation-tailoring-string) ] } ]
)
```

パラメータ

string-expression

文字列式には、データベース側文字セットでエンコードされた文字のみを指定できます。

`string-expression` が空の文字列の場合、SORTKEY 関数は長さ 0 のバイナリ値を返します。`string-expression` が NULL の場合、SORTKEY 関数は NULL 値を返します。空の文字列のソート順の値は、データベースカラムの NULL 文字列の値とは異なります。

SORTKEY 関数が処理できる文字列の最大長は 254 バイトです。これより長い部分は、無視されます。

collation-name

使用するソート順の名前を指定する文字列または文字変数。また、alias char_collation または equivalently, db_collation を指定して、データベースが CHAR の照合に使用するソートキーを生成するときに指定することもできます。同様に、エイリアス nchar_collation を指定して、データベースで使用している NCHAR の照合順でソートキーを生成できます。

collation-id

使用するソート順の ID 番号を指定する変数、定数 (整数)、または文字列。`collation-id` を指定しない場合のデフォルトは、デフォルト Unicode マルチ言語です。

collation-tailoring-string

必要に応じて、文字列のソートや比較を詳細に制御することために、照合の調整オプション (`collation-tailoring-string`) を指定することもできます。これらのオプションは、「キーワード=値」のペアの形式で、カッコで囲んで指定して、その後ろに照合名を記述します。たとえば、`'UCA(locale=es;case=LowerFirst;accent=respect)'` のように記述します。これらのオプションの組み合わせを指定する構文は、CREATE DATABASE 文の COLLATION 句を定義する構文と同じです。

> **注記**
>
> UCA 照合を指定すると、照合の適合化のすべてのオプションがサポートされます。その他の照合の場合、大文字小文字の区別の適合化のみがサポートされます。

戻り値

BINARY

備考

SORTKEY 関数が生成する値を使用して、事前定義済みのソート順の動作に基づいて結果を順序付けることができます。これにより、データベース照合では使用できない文字ソート順の動作を操作できます。SORTKEY 関数から保持される値はバイナリ値で、入力文字列のエンコードされたソート順情報が含まれています。たとえば、SORTKEY 関数から返された値を、ソース文字列と一緒にカラムに格納できます。必要な順序で文字データを取り出すには、SORTKEY 関数の実行結果が格納されているカラムの SELECT 文に ORDER BY 句を指定するだけです。

SORTKEY 関数は、特定のソート順の基準セットに対して返された値が、varbinary データ型で実行されるバイナリ比較で使用できることを保証します。

クエリのソートキーを作成する作業には負荷がかかります。使用頻度の高いソートキーの代替手段として、ソートキーの値を保存する計算カラムを作成し、クエリの ORDER BY 句を使用してカラムを参照する方法を検討します。

SORTKEY 関数の入力は、各入力文字に対して最大 6 バイトのソート順情報を生成できます。SORTKEY 関数の出力は VARBINARY 型で、最大長は 1024 バイトです。

ソートキーの生成中に照合に UCA を指定すると、デフォルトでは、照合の適合化はアクセント記号および大文字と小文字が区別されます。たとえば、UCA が自身によって指定されると、適用されるデフォルトの適合化は `'UCA(case=UpperFirst;accent=Respect;punct=Primary)'` に等しくなります。

SORTKEY に対する 2 番目のパラメータで別の照合を指定すると、その設定内容によって、デフォルトの設定内容が上書きされます。たとえば、次の 2 つの文は同じです。

```
SELECT SORTKEY( 'abc', 'UCA(accent=Ignore)' );
SELECT SORTKEY( 'abc', 'UCA(case=UpperFirst;accent=Ignore;punct=Primary)' );
```

UCA 以外の照合を指定しても、デフォルトでは、照合の適合化はアクセント記号および大文字と小文字が区別されます。ただし、UCA 以外の照合の場合、照合の適合化を使用して上書きできるのは、大文字と小文字の区別のみです。例:

```
SELECT SORTKEY( 'abc', '1252LATIN1(case=Respect)' );
```

適合化オプションを指定しないで作成されたデータベースでは (たとえば、`dbinit -c -zn uca -dba DBA,passwd mydb.db`)、次の 2 つの句では異なるソート順が生成されることがあります。これは、SORTKEY 関数に対してデータベースの照合名が指定されている場合でも同様です。

```
ORDER BY string-expression
ORDER BY SORTKEY( string-expression, database-collation-name )
```

このような現象が起こるのは、データベースの作成に使用されたデフォルトの適合化設定と、SORTKEY 関数のデフォルトの適合化設定が異なるためです。SORTKEY でもデータベース照合と同じ動作が実行されるようにするには、データベース照合の設定に一致する調整構文を `collation-tailoring-string` に指定するか、`collation-name` に specify db_collation を指定します。例:

```
SORTKEY( expression, 'db_collation' )
```

注記

SQL Anywhere のバージョンに応じて、異なるソートキー値が生成されます。このため、あるバージョンの SQL Anywhere で作成されたソートキー値を別のバージョンの SQL Anywhere で作成されたデータベースで使用すると、ソートに関する問題が発生する場合があります。この問題が発生した場合は、ソートキー値を再生成してください。

アンロード／再ロードを使用してデータベースをアップグレードする場合も、ソートキー値を再生成してください。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Employees テーブルに問い合わせ、すべての従業員の FirstName と Surname を返します。結果は、dict 照合 (Latin-1、英語、フランス語、ドイツ語辞書) を使用して、Surname カラムのソートキー値でソートされます。

```
SELECT Surname, GivenName FROM GROUPO.Employees ORDER BY SORTKEY( Surname,
'dict' );
```

次の例は、UCA 照合と適合化オプションを使用して、abc のソートキー値を返します。

```
SELECT SORTKEY( 'abc', 'UCA(locale=es;case=LowerFirst;accent=respect)' );
```

関連情報

文字列関数 [213 ページ]
COMPARE 関数 [文字列] [273 ページ]
CREATE DATABASE 文 [781 ページ]

1.3.2.201 SOUNDEX 関数 [文字列]

文字列の発音を表す数を返します。

構文

```
SOUNDEX( string-expression )
```

パラメータ

string-expression

評価される文字列。

戻り値

SMALLINT

備考

文字列の SOUNDEX 関数の値は、先頭の文字とそれに続く H、Y、W 以外の 3 つの子音がベースになっています。文字列の最初の文字である場合を除き、`string-expression` の母音は無視されます。同じ文字が 2 つ続く場合は 1 文字としてカウントされます。たとえば、"apples" という単語は、文字 A、P、L、S がベースになります。

マルチバイト文字は、SOUNDEX 関数では無視されます。

完全とは言えませんが、SOUNDEX 関数は通常、類似発音で同じ文字で始まる語句に対して同じ数を返します。

SOUNDEX 関数は、英単語の処理に最も優れています。他の言語の処理は英語の場合よりも劣ります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、それぞれの名前の発音を表す 2 つの同じ数 3827 を返します。

```
SELECT SOUNDEX( 'Smith' ), SOUNDEX( 'Smythe' );
```

関連情報

文字列関数 [213 ページ]

1.3.2.202 SPACE 関数 [文字列]

指定した数のスペースを返します。

構文

```
SPACE( integer-expression )
```

パラメータ

integer-expression

返すスペースの数。

戻り値

LONG VARCHAR

備考

`integer-expression` が正数でない場合は、NULL 文字列を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、10 個のスペースを含む文字列を返します。

```
SELECT SPACE( 10 );
```

関連情報

文字列関数 [213 ページ]

1.3.2.203 SQLDIALECT 関数 [その他]

文の SQL ダイアレクトを示す Watcom SQL または Transact-SQL のどちらかを返します。

構文

```
SQLDIALECT( sql-statement-string )
```

パラメータ

sql-statement-string

関数がダイアレクトの判断に使用する SQL 文。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、文字列 Transact-SQL を返します。

```
SELECT
   SQLDIALECT( 'SELECT EmployeeName = Surname FROM GROUPO.Employees' )
FROM SYS.DUMMY;
```

関連情報

TRANSACTSQL 関数 [その他] [569 ページ]
WATCOMSQL 関数 [その他] [591 ページ]

1.3.2.204 SQLFLAGGER 関数 [その他]

ANSI/ISO SQL 標準など、指定した規格に対する、指定した SQL 文の準拠性を返します。

構文

```
SQLFLAGGER( sql-standard-string, sql-statement-string )
```

パラメータ

sql-standard-string

準拠をテストする規格レベル。使用できる値は、sql_flagger_error_level データベースオプションと同じです。

SQL:2008/Core

コア SQL/2008 構文に対する準拠性をテストします。

SQL:2008/Package

上級レベルの SQL/2008 構文に対する準拠性をテストします。

SQL:2003/Core

コア SQL/2003 構文に対する準拠性をテストします。

SQL:2003/Package

上級レベルの SQL/2003 構文に対する準拠性をテストします。

SQL:1999/Core

コア SQL/1999 構文に対する準拠性をテストします。

SQL:1999/Package

上級レベルの SQL/1999 構文に対する準拠性をテストします。

SQL:1992/Entry

初級レベルの SQL/1992 構文に対する準拠性をテストします。

SQL:1992/Intermediate

中級レベルの SQL/1992 構文に対する準拠性をテストします。

SQL:1992/Full

上級レベルの SQL/1992 構文に対する準拠性をテストします。

UltraLite

Ultra Light に対する準拠性をテストします。

sql-statement-string

準拠性をチェックする SQL 文。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、使用できない拡張機能が検出されたときに返されるメッセージの例を示します。

```
SELECT SQLFLAGGER(
   'SQL:2003/Package', 'SELECT top 1 dummy_col FROM sys.dummy ORDER BY
dummy_col' );
```

この文は、メッセージ `'0AW03 Disallowed language extension detected in syntax near 'top' on line 1'` を返します。

次の文は、使用できない拡張機能を含んでいないため、`'00000'` を返します。

```
SELECT SQLFLAGGER( 'SQL:2003/Package', 'SELECT dummy_col FROM sys.dummy' );
```

関連情報

sa_ansi_standard_packages システムプロシージャ [1448 ページ]

1.3.2.205 SQRT 関数 [数値]

数の平方根を返します。

構文

```
SQRT( numeric-expression )
```

パラメータ

numeric-expression

平方根が計算される数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

標準

ANSI/ISO SQL 標準

SQRT 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。

例

次の文は、値 3 を返します。

```
SELECT SQRT( 9 );
```

1.3.2.206 STACK_TRACE 関数 [その他]

現在の文のスタックトレースに関する情報を返します。

構文

```
sa_stack_trace(
[ stack_frames
[, detail_level
[, connection_id ] ] ]
)
```

パラメータ

stack_frames

プロシージャ、外部文、またはその両方のどれを含めるかを制御します。

'procedure'

最も外側にある文ではなく、プロシージャを返します。これはデフォルトの動作です。

'caller'

最も外側にある文 (クライアントから受け取った文) のみ返します。

'procedure+caller' または **'caller+procedure'**

すべての文を返します。

detail_level

返されたデータに含める詳細のレベルを制御します。

'stack'

プロシージャ名と行番号を含めます。これはデフォルトの動作です。

'stack+sql' または **'sql+stack'**

プロシージャ名と行番号に加えて、各レベルで実行される文の SQL テキストを含めます。

connection_id

connection_id オプションを使用して、指定された ID に返される結果をフィルタします。

戻り値

現在の文のスタックトレースを表す LONG VARCHAR。

備考

結果には、改行文字 (¥n) で区切られたテキストの行が含まれます。返される値の各行には、修飾プロシージャ名またはバッチタイプが含まれ、文の行数が続きます。返される値の最後の行の末尾は、改行文字ではありません。スタックトレースの最初の行は、関数が呼び出された行を表します。複合文がプロシージャ、ファンクション、トリガ、イベントのいずれの一部でもない場合は、プロシージャ名ではなく、バッチのタイプ (watcom_batch または tsql_batch) が返されます。

この関数は、プロシージャの SYSPROCEDURE システムテーブルの proc_defn カラムで見つかった行番号を返します。これらの行番号は、プロシージャの作成に使用されるソース定義の行番号とは異なる可能性があります。

この関数は、sa_stack_trace システムプロシージャと同じ情報を返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、プロシージャコールスタックトレースを示しています。

```
CREATE OR REPLACE PROCEDURE proc3()
BEGIN
   DECLARE v INTEGER;
   SET v = 1;
   SELECT * FROM sa_split_list( STACK_TRACE('caller+procedure', 'stack+sql'),
'¥n' );
END;
CREATE OR REPLACE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;
CREATE OR REPLACE PROCEDURE proc1()
BEGIN
    CALL proc2();
END;
CALL proc1();
```

結果:

```
line_num row_value
-------------------------------------------------------------------------------
-----------------------------------------------
          1   "DBA"."proc3" : 5 : select
sa_split_list.line_num,sa_split_list.row_value from
sa_split_list(STACK_TRACE('caller+procedure','stack
+sql'),'¥x0A')

          2   "DBA"."proc2" : 3 : call
proc3()


          3   "DBA"."proc1" : 3 : call
proc2()


          4    call proc1(
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]
SYSPROCEDURE システムビュー [1824 ページ]

1.3.2.207 STDDEV 関数 [集合]

STDDEV_SAMP のエイリアスです。

関連情報

STDDEV_SAMP 関数 [集合] [544 ページ]

1.3.2.208 STDDEV_POP 関数 [集合]

数値式からなる母集団の標準偏差を DOUBLE として計算します。

構文

式

```
STDDEV_POP( numeric-expression )
```

Window 関数

```
STDDEV_POP( numeric-expression ) OVER ( window-spec )
```

```
window-spec : see the Remarks section below
```

パラメータ

numeric-expression

ローセットで母集団ベースの標準偏差を計算する対象の式。通常、式はカラム名です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

母集団ベースの標準偏差は、次の式によって計算されます。

$s = [(1/N) * SUM(x_i - mean(x))^2]^{½}$

この標準偏差には、numeric-expression が NULL 値のローは含まれません。グループにローが含まれていない場合は、NULL を返します。

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

STDDEV_POP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。Window 関数として使用する場合、STDDEV_POP はオプションの ANSI/ISO SQL 基本機能 T611、「Elementary OLAP operations」の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、「Full value expressions」の一部です。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、「Extended set function support」もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set function」がサポートされていません。このソフトウェアでは、STDDEV_POP 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT YEAR( ShipDate ) AS Year,
    QUARTER( ShipDate ) AS Quarter,
    AVG( Quantity ) AS Average,
    STDDEV_POP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	14.2794...
2000	2	27.050847	15.0270...
...	...	...	...

関連情報

集合関数 [200 ページ]

WINDOW 句 [1409 ページ]

1.3.2.209 STDDEV_SAMP 関数 [集合]

数値式からなるサンプルの標準偏差を DOUBLE として計算します。

構文

式

```
STDDEV_SAMP( numeric-expression )
```

Window 関数

```
STDDEV_SAMP( numeric-expression ) OVER ( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

numeric-expression

ローセットでサンプルベースの標準偏差を計算する対象の式です。通常、式はカラム名です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準偏差は、次の式によって計算されます。これは、正規分布を前提としています。

$s = [(1/(N - 1)) * SUM(x_I - mean(x))^2]^{½}$

この標準偏差には、`numeric-expression` が NULL 値のローは含まれません。グループに 0 か 1 のローが含まれている場合は、NULL を返します。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

STDDEV_SAMP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。Window 関数として使用する場合、STDDEV_SAMP はオプションの SQL 基本機能 T611、「Elementary OLAP operations」の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、「Full value expressions」の一部です。このソフトウェアでは、ANSI/ISO SQL 標準言語機能 F441、「Extended set function support」もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set function」がサポートされていません。このソフトウェアでは、STDDEV_SAMP 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることもできません。

例

次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT YEAR( ShipDate ) AS Year,
    QUARTER( ShipDate ) AS Quarter,
    AVG( Quantity ) AS Average,
    STDDEV_SAMP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	14.3218...
2000	2	27.050847	15.0696...
...	...	...	...

関連情報

集合関数 [200 ページ]
WINDOW 句 [1409 ページ]
AVG 関数 [集合] [244 ページ]

1.3.2.210 STR 関数 [文字列]

指定した数に相当する文字列を返します。

構文

```
STR( numeric-expression [, length [, decimal ] ] )
```

パラメータ

numeric-expression

-1E126 と 1E127 との間の任意の概数 (浮動小数点、実数、または倍精度)。

length

返される文字数 (小数点、小数点の左右のすべての桁、ブランクを含む)。デフォルトは 10 です。

decimal

返される小数点以下の桁数。デフォルトは 0 です。

戻り値

VARCHAR

備考

数の整数部分が指定した長さに合わない場合は、指定した長さの文字列がすべてアスタリスクで埋められて返されます。たとえば、次の文は *** を返します。

```
SELECT STR( 1234.56, 3 );
```

注記

サポートされる最大長は 128 です。1 ～ 128 の範囲にない長さでは結果が NULL になります。

標準

ANSI/ISO SQL 標準

標準の機能。

例

次の文は、6 つのスペースの後に 1235 が続く、合計で 10 の文字からなる文字列を返します。

```
SELECT STR( 1234.56 );
```

次の文は、1234.6 を返します。

```
SELECT STR( 1234.56, 6, 1 );
```

関連情報

文字列関数 [213 ページ]

1.3.2.211 STRING 関数 [文字列]

1 つ以上の文字列を連結して 1 つの長い文字列にします。

構文

```
STRING( string-expression [, ... ] )
```

パラメータ

string-expression

評価される文字列。

引数を 1 つだけ指定する場合は、1 つの式に変換されます。複数の引数を指定する場合は、連結されて 1 つの文字列になります。

戻り値

LONG VARCHAR
LONG NVARCHAR
LONG BINARY

備考

数値または日付をパラメータとして指定した場合は、文字列に変換されてから連結されます。また、1 つの式を唯一のパラメータとして指定すると、STRING 関数を使用して、その式を文字列に変換できます。

すべてのパラメータが NULL の場合、STRING は NULL を返します。NULL でないパラメータが存在すると、NULL パラメータはすべて空の文字列として処理されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 `testing123` を返します。

```
SELECT STRING( 'testing', NULL, 123 );
```

関連情報

文字列関数 [213 ページ]

1.3.2.212 STRTOUUID 関数 [文字列]

文字列の値をユニークな識別子 (UUID または GUID) の値に変換します。

構文

```
STRTOUUID( string-expression )
```

パラメータ

string-expression

`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` 形式の文字列。

戻り値

UNIQUEIDENTIFIER

備考

xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx の形式の文字列 (x は 16 進の桁) を、一意な識別子の値に変換します。

この関数は、UUID 値をデータベースに挿入するときに役立ちます。

文字列が有効な UUID 文字列でない場合は、conversion_error オプションが OFF に設定されていないかぎり変換エラーが返されます。このオプションが OFF の場合は NULL が返されます。この関数は NCHAR の入力または出力をサポートしています。波括弧は、string-expression の最初と最後の文字として使用できます。バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義されており、UUID 値のバイナリ表現と文字列表現の間の変換には STRTOUUID 関数と UUIDTOSTR 関数が必要でした。バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブデータ型に変更されています。データ型の変換は、データベースサーバが必要に応じて実行します。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。

Ultra Light: バージョン 9.0.2 より前に作成されたデータベースでは、UUID 値のバイナリ表現と文字列表現を変換するために STRTOUUID 関数と UUIDTOSTR 関数が必要でした。バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブデータ型に変更されています。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は同等であり、いずれも結果 0x6c2b64a93c6f47dc901536b9ed49fec2 を返します。

```
SELECT STRTOUUID( '6c2b64a9-3c6f-47dc-9015-36b9ed49fec2' );
SELECT STRTOUUID( '{6c2b64a9-3c6f-47dc-9015-36b9ed49fec2}' );
```

関連情報

文字列関数 [213 ページ]
UNIQUEIDENTIFIER データ型 [175 ページ]

UUIDTOSTR 関数 [文字列] [584 ページ]
NEWID 関数 [その他] [447 ページ]

1.3.2.213 STUFF 関数 [文字列]

1 つの文字列から複数の文字を削除して、別の文字列に置き換えます。

構文

```
STUFF( string-expression-1, start, length, string-expression-2 )
```

パラメータ

string-expression-1

STUFF 関数によって変更される文字列。

start

削除を開始する文字の位置。文字列の先頭文字の位置を 1 とします。

length

削除する文字数。

string-expression-2

挿入する文字列。STUFF 関数を使用して文字列の一部を削除するには、NULL の置換文字列を使用します。

戻り値

入力式のデータ型に応じて、LONG BINARY、LONG VARCHAR、または LONG NVARCHAR。

備考

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 chocolate pie を返します。

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' );
```

関連情報

文字列関数 [213 ページ]
INSERTSTR 関数 [文字列] [405 ページ]

1.3.2.214 SUBSTRING 関数 [文字列]

文字列の部分文字列を返します。

構文

```
{ SUBSTRING | SUBSTR }( string-expression, start
[, length ] )
```

パラメータ

string-expression

部分文字列が返される文字列。

start

文字単位で指定した、返される部分文字列の開始位置。

length

文字単位で指定した、返される部分文字列の長さ。`length` を指定すると、指定した長さの部分文字列に制限されます。

戻り値

LONG BINARY
LONG VARCHAR
LONG NVARCHAR
Ultra Light では LONG BINARY および LONG VARCHAR が返されます。

備考

文字列の末尾の文字を取得するには、RIGHT 関数を使用します。

`string-expression` が binary データ型の場合、SUBSTRING 関数は BYTE_SUBSTR のように動作します。

この関数は NCHAR の入力または出力をサポートしています。入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。この関数の動作は、ansi_substring データベースオプションの設定によって変わります。ansi_substring オプションを On (デフォルト) に設定した場合、SUBSTRING 関数は ANSI/ISO SQL 標準と同じ動作をします。動作を次に示します。

ansi_substring オプション設定	start 値	length 値
On	文字列の先頭文字の位置が 1 になります。負またはゼロの開始オフセットは、文字列の左側が文字以外で埋められたように扱われます。	正の `length` は、部分文字列が開始位置の右側から `length` 文字で終わることを示します。 負の `length` はエラーを返します。
Off	文字列の先頭文字の位置が 1 になります。負の開始位置を指定する場合は、文字列の最初からの文字数ではなく、文字列の最後からの文字数を指定します。 `start` が 0 で length が負でない場合は、1 の start 値が使用されます。start が 0 で `length` が負の場合は、-1 の start 値が使用されます。	正の `length` は、部分文字列が開始位置の右側から `length` 文字で終わることを示します。 負の `length` は、開始位置から最大 `length` 文字左側の文字を返します。

Ultra Light: 入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。Ultra Light では、データベースに ansi_substring オプションがありませんが、デフォルトでは、SUBSTR 関数は ansi_substring が ON に設定されているように動作します。関数の動作は ANSI/ISO SQL 標準の動作に対応しています。

Start 値

文字列の先頭文字の位置が 1 になります。負またはゼロの開始オフセットは、文字列の左側が文字以外で埋められたように扱われます。

Length 値

正の `length` は、部分文字列が開始位置の右側から `length` 文字で終わることを示します。

負の `length` はエラーを返します。

0 の `length` は、空の文字列を返します。

標準

ANSI/ISO SQL 標準

コア機能。ただし、ANSI/ISO SQL 標準の実装内容はこのソフトウェアの実装内容とはわずかに異なります。標準ではキーワード FROM と FOR を使用して SUBSTRING を 3 つのパラメータで定義しますが、このソフトウェアではこれらのキーワードはいずれも必要ありません。

例

次の表は、SUBSTRING 関数から返される値を示します。

例	結果
SUBSTRING('front yard', 1, 4)	fron
SUBSTRING('back yard', 6, 4)	yard
SUBSTR('abcdefgh', 0, -2)	SQL Anywhere ansi_substring オプションが ON の場合、エラーを返します
SUBSTR('abcdefgh', -2, 2)	SQL Anywhere ansi_substring オプションが ON の場合、空の文字列を返します

Ultra Light: 次の表は、SUBSTRING 関数から返される値を示します。

例	結果
SUBSTRING('front yard', 1, 4)	fron
SUBSTRING('back yard', 6, 4)	yard
SUBSTR('abcdefgh', 0, -2)	エラーを返します
SUBSTR('abcdefgh', -2, 2)	空の文字列を返します

関連情報

文字列関数 [213 ページ]
BYTE_SUBSTR 関数 [文字列] [261 ページ]
LEFT 関数 [文字列] [416 ページ]
RIGHT 関数 [文字列] [508 ページ]
CHARINDEX 関数 [文字列] [270 ページ]

1.3.2.215 SUM 関数 [集合]

ローグループごとに、指定された式の合計を返します。

構文

式

```
SUM( [ ALL | DISTINCT ] expression )
```

Window 関数

```
SUM( [ ALL ] expression ) OVER ( window-spec )
```

`window-spec`: 以下の備考部分を参照してください。

Ultra Light 構文:式

```
SUM( [ DISTINCT ] expression )
```

パラメータ

expression

合計される式の名前。通常はカラム名です。

[ALL] expression

合計される式の名前。通常はカラム名です。

DISTINCT expression

各グループの `expression` で一意の値の合計を計算します。

戻り値

- INTEGER
- DOUBLE
- NUMERIC

備考

指定された式が NULL のローは含まれません。

グループにローが含まれていない場合は、NULL を返します。

この関数ではオーバーフローエラーが発生することがあり、これにより、エラーが返される場合があります。オーバーフローエラーを回避するために、numeric-expression で CAST 関数を使用できます。

window-spec を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、window-spec の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

コア機能。Window 関数として使用する場合、SUM はオプションの ANSI/ISO SQL 言語機能 T611、"Basic OLAP operations" の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、"Full value expressions" の一部です。このソフトウェアでは、言語機能 F441、"Extended set function support" もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの機能 F442、"Mixed column references in set function" がサポートされていません。このソフトウェアでは、SUM 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の文は、値 3749146.740 を返します。

```
SELECT SUM( Salary )
FROM GROUPO.Employees;
```

関連情報

WINDOW 句 [1409 ページ]
COUNT 関数 [集合] [289 ページ]
AVG 関数 [集合] [244 ページ]

1.3.2.216 SUSER_ID 関数 [システム]

指定したユーザ名の数値のユーザ ID を返します。

構文

```
SUSER_ID( [ user-name ] )
```

パラメータ

user-name

検索しているユーザ ID のユーザ名。

戻り値

VARCHAR

備考

`user-name` を指定しない場合は、現在のユーザの ID が返されます。

この機能は、他のベンダーとの互換性を確保するために提供されています。まったく同じものである USER_ID 関数を使用することもできます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は 101 (GROUPO ユーザの ID) を返します。

```
SELECT SUSER_ID( 'GROUPO' );
```

関連情報

SUSER_NAME 関数 [システム] [557 ページ]
USER_ID 関数 [システム] [582 ページ]

1.3.2.217 SUSER_NAME 関数 [システム]

指定したユーザ ID のユーザ名を返します。

構文

```
SUSER_NAME( [ user-id ] )
```

パラメータ

user-id

検索しているユーザのユーザ ID。

戻り値

VARCHAR

備考

`user-id` を指定しない場合は、現在のユーザのユーザ名が返されます。

この機能は、他のベンダーとの互換性を確保するために提供されています。まったく同じものである USER_NAME 関数を使用することもできます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は GROUPO (ID 101 のユーザのユーザ名) を返します。

```
SELECT SUSER_NAME( 101 );
```

関連情報

SUSER_ID 関数 [システム] [555 ページ]
USER_NAME 関数 [システム] [583 ページ]

1.3.2.218 SWITCHOFFSET 関数 [日付と時刻]

元のタイムゾーンオフセットから指定のタイムゾーンオフセットに変換される TIMESTAMP WITH TIME ZONE 値を返します。

構文

```
SWITCHOFFSET( tmz-expression, time-zone-offset )
```

パラメータ

tmz-expression

変換される TIMESTAMP WITH TIME ZONE 値。

time-zone-offset

結果のタイムゾーンオフセット。協定世界時 (UTC: Coordinated Universal Time) の前または後の分数を表す整数、{ + | - } hh:nn 形式の文字列、またはズールータイムゾーンを表す Z のいずれかをこの値として使用できます。ズールータイムゾーンは、UTC と同じタイムゾーンです。

戻り値

TIMESTAMP WITH TIME ZONE

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、タイムゾーンオフセット値を -04:00 時から -07:00 時に変更します。戻り値は 2009-04-03 11:45:12.123-07:00 です。

```
SELECT CAST ( '2009-04-03 14:45:12.123-04:00' AS datetimeoffset ) AS EDT,
```

```
SWITCHOFFSET( EDT,'-07:00' ) AS PDT;
```

関連情報

TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
SYSDATETIMEOFFSET 関数 [日付と時刻] [559 ページ]

1.3.2.219 SYSDATETIMEOFFSET 関数 [日付と時刻]

システムクロックを使用するデータベースサーバの現在の日付、時刻、およびタイムゾーンオフセットを返します。

構文

```
SYSDATETIMEOFFSET ( )
```

戻り値

TIMESTAMP WITH TIME ZONE

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、データベースサーバの現在の日付と時刻およびタイムゾーンオフセットを返します。

```
SELECT SYSDATETIMEOFFSET ( );
```

次の例は、SYSDATETIMEOFFSET 値をクライアントコンピュータのタイムゾーンに変換します。

```
SELECT SWITCHOFFSET ( SYSDATETIMEOFFSET ( ),
CAST( connection_property ('TimeZoneAdjustment') AS INT ));
```

関連情報

TIMESTAMP WITH TIME ZONE データ型 [170 ページ]
SWITCHOFFSET 関数 [日付と時刻] [558 ページ]

1.3.2.220 TAN 関数 [数値]

数のタンジェントを返します。

構文

```
TAN( numeric-expression )
```

パラメータ

numeric-expression

角度 (ラジアン)。

戻り値

DOUBLE

備考

ATAN 関数と TAN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、0.52 の tan 値を返します。

```
SELECT TAN( 0.52 );
```

関連情報

COS 関数 [数値] [287 ページ]
SIN 関数 [数値] [527 ページ]

1.3.2.221 TEXTPTR 関数 [テキストとイメージ]

指定したカラムへの 16 バイトのバイナリポインタを返します。この機能は Transact-SQL との互換性のためにのみ提供されているものであり、使用しないことをお奨めします。

構文

```
TEXTPTR( column-name )
```

パラメータ

column-name

CHAR、NCHAR、または BINARY のデータが含まれるカラムの名前。

戻り値

BINARY

備考

この関数は、Transact-SQL との互換性を保つために実装されています。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の Embedded SQL の例では、TEXTPTR を使用して、MarketingInformation テーブルの ProductID 500 に関連付けられている Description カラムを見つけます。

```
EXEC SQL BEGIN DECLARE SECTION;
char            hostvar[100];
EXEC SQL END DECLARE SECTION;
EXEC SQL create variable txtptr binary(16);
EXEC SQL set txtptr =
    ( SELECT textptr(Description)
        FROM GROUPO.MarketingInformation
        WHERE ProductID = '500' );
EXEC SQL PREPARE S1 FROM
    'READTEXT GROUPO.MarketingInformation.Description txtptr 181 55';
EXEC SQL EXECUTE S1 INTO :hostvar;
printf( "hostvar: %s¥n", hostvar );
```

テキストポインタは、変数 txtptr に格納され、READTEXT 文のパラメータとして指定されます。READTEXT 文は、カラムオフセット 181 から開始して 55 バイトを返します。READTEXT は次の文字列を返します。

```
Lightweight 100% organically grown cotton construction.
```

1.3.2.222 TO_CHAR 関数 [文字列]

任意のサポートされている文字セットの文字データを、データベースの CHAR 文字セットに変換します。

構文

```
TO_CHAR( string-expression [, source-charset-name ] )
```

パラメータ

string-expression

変換される文字列。

source-charset-name

文字列の文字セット。

戻り値

LONG VARCHAR

備考

source-charset-name を指定すると、この関数は次と同等になります。

```
CAST( CSCONVERT( CAST( string-expression AS BINARY ),
 'db_charset', source-charset-name )
   AS CHAR );
```

source-charset-name を指定しないと、この関数は次と同等になります。

```
CAST( string-expression AS CHAR );
```

標準

ANSI/ISO SQL 標準

標準になし。

例

cp850 文字セットを含む BINARY 値の場合、次の文はデータを CHAR の文字セットとデータ型に変換します。

```
SELECT TO_CHAR( 'cp850_data', 'cp850' );
```

関連情報

CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]
CSCONVERT 関数 [文字列] [297 ページ]
NCHAR 関数 [文字列] [446 ページ]
TO_NCHAR 関数 [文字列] [564 ページ]
UNICODE 関数 [文字列] [578 ページ]
UNISTR 関数 [文字列] [579 ページ]

1.3.2.223 TO_NCHAR 関数 [文字列]

任意のサポートされている文字セットの文字データを、NCHAR 文字セットに変換します。

構文

```
TO_NCHAR( string-expression [, source-charset-name ] )
```

パラメータ

string-expression

変換される文字列。

source-charset-name

文字列の文字セット。

戻り値

LONG NVARCHAR

備考

source-charset-name を指定すると、この関数は次と同等になります。

```
CAST( CSCONVERT( CAST( string-expression AS BINARY ),
 'nchar_charset', source-charset-name )
 AS NCHAR );
```

source-charset-name を指定しないと、この関数は次と同等になります。

```
CAST( string-expression AS NCHAR );
```

標準

ANSI/ISO SQL 標準

標準になし。

例

cp850 文字セットを含む BINARY 値の場合、次の例はデータを NCHAR の文字セットとデータ型に変換します。

```
SELECT TO_NCHAR( 'cp850_data', 'cp850' );
```

関連情報

CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]
CSCONVERT 関数 [文字列] [297 ページ]
NCHAR 関数 [文字列] [446 ページ]
TO_CHAR 関数 [文字列] [562 ページ]
UNICODE 関数 [文字列] [578 ページ]
UNISTR 関数 [文字列] [579 ページ]

1.3.2.224 TODATETIMEOFFSET 関数 [日付と時刻]

指定のタイムゾーンオフセットを使用して、TIMESTAMP 値を TIME STAMP WITH TIME ZONE 値に変換します。

構文

```
TODATETIMEOFFSET( timestamp-expression, time-zone-offset )
```

パラメータ

timestamp-expression

変換される TIMESTAMP 式。

time-zone-offset

タイムゾーンオフセット。UTC の前または後の分数を表す INTEGER、{ + | - } hh:nn 形式の VARCHAR、またはズールータイムゾーンを表す文字列 "Z" のいずれかをこの値として使用できます。ズールータイムゾーンは、UTC と同じタイムゾーンです。

戻り値

TIMESTAMP WITH TIME ZONE

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、TIMESTAMP 値を TIMESTAMP WITH TIME ZONE 値に変換します。

```
SELECT CAST('2009-04-03 14:45:12.123' AS TIMESTAMP) AS orig,
        TODATETIMEOFFSET (orig,'+11:00');
```

関連情報

TIMESTAMP WITH TIME ZONE データ型 [170 ページ]

1.3.2.225 TODAY 関数 [日付と時刻]

現在の日付を DATE 値で返します。

構文

```
TODAY( [ * ] )
```

戻り値

DATE

備考

セマンティック上、TODAY(*) と TODAY() は同義です。TODAY は CURRENT DATE 特別値と同義です。

要求に含まれる TODAY 関数の各インスタンスは、多くても 1 回のみ評価されます。同じ要求に含まれる TODAY の複数のインスタンスは、同じ DATE 値を共有する場合もあれば、そうでない場合もあります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、システムクロックによる現在の日付を返します。

```
SELECT TODAY( * );
SELECT CURRENT DATE;
```

1.3.2.226 TRACEBACK 関数 [その他]

ストアドプロシージャ、トリガ、またはカスタム関数の実行中に発生した最後の例外 (エラー) のスタックで、文を返します。

構文

```
TRACEBACK( [ * ] )
```

戻り値

LONG VARCHAR

備考

返された呼び出しスタックは、オブジェクト名と行番号を表します。HIDDEN 定義を持つプロシージャからの文は、スタックトレースに <hidden> として記録されます。

TRACEBACK の文は、ストアドプロシージャの文のデータベースサーバ表現から生成され、プロシージャ定義のテキストとは正確に一致しない場合があります。

セマンティック上、TRACEBACK(*) と TRACEBACK() は同義です。

この関数は、プロシージャとトリガ (特に、Transact-SQL ダイアレクトで記述されている場合) のデバッグに役立ちます。

標準

ANSI/ISO SQL 標準

標準になし。

例

TRACEBACK 関数を使用する場合は、プロシージャの実行中にエラーが発生した後で次の文を実行します。

```
SELECT TRACEBACK( * );
```

TRACEBACK 関数を使用した後で、次と同様の出力が返されます。

```
"user1"."proc1" : 10 : set ret_val = in_val / (in_val - in_val)
"user2"."proc2" : 5 : <hidden>
"user3"."proc1" : 7 : call user2.proc2( 10 )
```

1.3.2.227 TRACED_PLAN 関数 [その他] (廃止予定)

診断トレース機能は廃止予定です。SQL Anywhere プロファイラを使用して、データベースの問題を診断します。この関数は、*SQL Central* でトレーシングデータを使用してクエリのグラフィカルなプランを生成するときに使用します。

構文

```
TRACED_PLAN( logging_session_id, query_id )
```

パラメータ

logging_session_id

この INTEGER パラメータと `query_id` を組み合わせると、プランを生成する sa_diagnostic_query のローを識別できます。

query_id

この INTEGER パラメータと `logging_session_id` を組み合わせると、プランを生成する sa_diagnostic_query のローを識別できます。

戻り値

LONG VARCHAR

備考

この関数は *SQL Central* から使用されます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

sa_diagnostic_query テーブル (廃止予定) [1422 ページ]

1.3.2.228 TRANSACTSQL 関数 [その他]

Transact-SQL に Watcom SQL 文を書き直します。

構文

```
TRANSACTSQL( sql-statement-string )
```

パラメータ

sql-statement-string

Transact-SQL 表現に書き換えられる SQL 文。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、文字列 'select EmployeeName=empl_name from GROUPO.Employees' を返します。

```
SELECT TRANSACTSQL( 'SELECT empl_name as EmployeeName FROM GROUPO.Employees' )
FROM SYS.DUMMY;
```

関連情報

SQLDIALECT 関数 [その他] [536 ページ]
WATCOMSQL 関数 [その他] [591 ページ]

1.3.2.229 TREAT 関数 [データ型変換]

ジオメトリ式の宣言されたタイプをサブタイプに変更します。この関数は、空間データで使用します。

構文

```
TREAT( geometry-expression AS subtype )
```

パラメータ

geometry-expression

変換される式。

subtype

geometry-expression の変換後のサブタイプ。

戻り値

要求されたデータ型によって異なります。

備考

TREAT 関数は、ジオメトリでのみ使用できます。

式の動的タイプがターゲットのデータ型のサブタイプでない場合は、エラーが返されます。また、CAST 関数を使用して、ジオメトリ式が宣言されたタイプを変更することもできます。ただし、CAST 関数では、サブタイプ階層の外側を変更できます。たとえば、CAST を使用して、ポイントをマルチポイントに変換することができます。このような変換を実行すると、式の動的タイプに予期しない変更が加えられることがあるため、スーパータイプからサブタイプに移動する場合は TREAT 関数を使用することをお奨めします。また、TREAT 関数は、CAST 関数と比較して、より効率的に実行されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文を実行して、テーブルを作成し、2 つの値をロードします。

```
DROP TABLE IF EXISTS treatExample;
CREATE TABLE treatExample( pk INT PRIMARY KEY, geo ST_Geometry );
INSERT INTO treatExample VALUES(0, NEW ST_Point(3,4) );
INSERT INTO treatExample VALUES(1, NEW ST_MultiPoint( new ST_Point( 5, 6 ) ) );
```

次のクエリはエラーを返します。

```
SELECT geo.ST_X() FROM treatExample T WHERE pk = 0;
```

次のクエリは成功します。

```
SELECT TREAT( geo AS ST_Point ).ST_X() FROM treatExample WHERE pk = 0;
```

次のクエリはエラーを返します。

```
SELECT TREAT( geo AS ST_Point ).ST_X() FROM treatExample T WHERE pk = 0;
```

次の例では、TREAT 文の代わりに CAST 文が使用されているため、成功します。

```
SELECT CAST( geo AS ST_Point ) FROM treatExample WHERE pk = 1;
```

関連情報

CAST 関数 [データ型変換] [264 ページ]

1.3.2.230 TRIM 関数 [文字列]

先行ブランクと後続ブランクを文字列から削除します。

構文

```
TRIM( string-expression )
```

パラメータ

string-expression

削除する文字列。

戻り値

VARCHAR
NVARCHAR
LONG VARCHAR
LONG NVARCHAR
Ultra Light の場合、VARCHAR または LONG VARCHAR が返されます。

備考

この関数は NCHAR の入力または出力をサポートしています。

Ultra Light では NCHAR の入力または出力がサポートされていません。

標準

ANSI/ISO SQL 標準

コア機能。

このソフトウェアでは、ANSI/ISO SQL 標準で定義されている追加のパラメータ `trim specification` および `trim character` がサポートされていません。ソフトウェアに用意されている TRIM の実装は、BOTH の TRIM 仕様に対応しています。

ANSI/ISO SQL 標準で定義される他の TRIM 仕様 (LEADING と TRAILING) の場合、それぞれ LTRIM 関数と RTRIM 関数で提供されます。

例

次の文は、先行ブランクや後続ブランクがすべて削除された値 chocolate を返します。

```
SELECT TRIM( '   chocolate   ' );
```

関連情報

文字列関数 [213 ページ]
LTRIM 関数 [文字列] [428 ページ]
RTRIM 関数 [文字列] [515 ページ]

1.3.2.231 TRIM_ARRAY 関数 [複合]

配列で指定した数の要素で構成される、暗黙的にバインドされた配列を返します。

構文

```
TRIM_ARRAY( array-expression, integer-expression )
```

パラメータ

array-expression

削除する配列。

integer-expression

生成される配列に含まれる要素の数。生成される配列では、array-expression に最初の integer-expression の数の要素が含まれます。

integer-expression が 0 の場合は、生成される配列が空になります。integer-expression が 0 未満の場合、または配列のカーディナリティよりも大きい場合は、エラーが生成されます。

戻り値

ARRAY

備考

TRIM_ARRAY の引数のいずれかが NULL の場合は、その結果も NULL となります。

標準

ANSI/ISO SQL 標準

機能 S404。

例

次の例は、TRIM_ARRAY 関数を使用して、4 つの要素が含まれる配列の最初の 2 つの要素で構成される配列を作成する方法を示します。

```
DECLARE UntrimmedArray ARRAY( 4 ) OF INT;
DECLARE TrimmedArray ARRAY( 2 ) OF INT;
SET UntrimmedArray = ARRAY( 1, 2, 3, 4 );
SET TrimmedArray = TRIM_ARRAY( UntrimmedArray, 2 );
```

関連情報

ARRAY_AGG 関数 [集合] [236 ページ]
ARRAY_MAX_CARDINALITY 関数 [複合] [238 ページ]
CARDINALITY 関数 [複合] [262 ページ]

1.3.2.232 TRUNCNUM 関数 [数値]

指定した桁数で小数点以下を切り捨てます。

構文

```
{ TRUNCNUM | TRUNCATE }( numeric-expression, integer-expression )
```

パラメータ

numeric-expression

トランケートされる数。この引数には、NUMERIC または DOUBLE データ型を使用できます。

integer-expression

正の整数は、丸めを行う小数点の右側の有効桁数を指定します。負の値は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値

NUMERIC または DOUBLE

備考

数字をトランケートする場合、TRUNCATE 関数ではなく TRUNCNUM を使用します。

TRUNCATE 関数の使用はおすすめしません。truncate という単語はキーワードなので、quoted_identifier オプションを OFF に設定するか、単語 TRUNCATE を引用符で囲む必要があるためです。

Ultra Light: いずれかのパラメータが NULL 値の場合、結果は NULL 値になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 600 を返します。

```
SELECT TRUNCNUM( 655, -2 );
```

次の文は、値 655.340 を返します。

```
SELECT TRUNCNUM( 655.348, 2 );
```

関連情報

ROUND 関数 [数値] [509 ページ]

1.3.2.233 TSEQUAL 関数 [システム] (旧式)

2 つの TIMESTAMP 値を比較し、これらが同じかどうかを返します。

構文

```
TSEQUAL ( timestamp-expression-1, timestamp-expression-2 )
```

パラメータ

timestamp-expression-1

TIMESTAMP 値。

timestamp-expression-2

TIMESTAMP 値。

戻り値

BIT

備考

TSEQUAL 関数は、WHERE 句でのみ使用できます。これは、UPDATE 文の一部として最も一般的に使用されます。

TSEQUAL 関数は通常の 2 つの TIMESTAMP 値の比較に使用できますが、TSEQUAL の目的は、2 つの特殊な Transact-SQL TIMESTAMP 値を比較することでローが他の接続によって変更されたかどうかを判断することにあります。

TSEQUAL を使用するシングルローの UPDATE 文で、`timestamp-expression-1` と `timestamp-expression-2` が等しく、このうちの一方が DEFAULT TIMESTAMP で宣言されたカラムを参照しており、ローが最後にフェッチされたときのカラム値をもう一方が参照している場合は、フェッチ後にローが変更されていないため、TSEQUAL は TRUE を返します。別のユーザがローを変更した場合は、そのタイプスタンプが変更されており、TSEQUAL 関数は FALSE を返します。TSEQUAL 関数がこの場合に FALSE を返すと、UPDATE は実行されません。アプリケーションでは、影響を受けるロー数を @@rowcount などを使用して調べることにより、ローが更新されたかどうかを判断できます。影響を受けるローがない場合は、別のユーザがローを変更したと想定でき、再フェッチが必要になります。

標準

ANSI/ISO SQL 標準

標準になし。

例

TIMESTAMP カラム Products.LastUpdated を作成し、ローが最後に更新された時刻のタイムスタンプを格納するとします。次の例は、TSEQUAL 関数を使用してローの値を変更します。最後にフェッチされた後でローに変更がない場合にのみ、更新が適用されます。

```
SELECT LastUpdated into old_ts_value
FROM GROUPO.Products
WHERE ID = '300';
```

```
UPDATE GROUPO.Products
SET Color = 'Yellow'
WHERE ID = '300'
AND TSEQUAL( LastUpdated, old_ts_value );
```

関連情報

TIMESTAMP 特別値 [106 ページ]
UPDATE 文 [1391 ページ]

1.3.2.234 UCASE 関数 [文字列]

文字列中のすべての文字を大文字に変換します。

構文

```
UCASE( string-expression )
```

パラメータ

string-expression

大文字に変換される文字列。

戻り値

NCHAR データで使用された場合は LONG NVARCHAR
データベース照合が UCA の場合、CHAR データで使用されるときは LONG VARCHAR

それ以外の場合、データ型は入力データ型と同じになります。
Ultra Light では、入力データ型と同じデータ型が返されます。

備考

この関数は UPPER 関数と同じです。

標準

ANSI/ISO SQL 標準

標準になし。UPPER 関数は、ANSI/ISO SQL 標準準拠です。

例

次の文を実行すると、値 CHOCOLATE が返ります。

```
SELECT UCASE( 'ChocoLate' );
```

関連情報

文字列関数 [213 ページ]
UPPER 関数 [文字列] [581 ページ]
LCASE 関数 [文字列] [415 ページ]

1.3.2.235 UNICODE 関数 [文字列]

文字列の最初の文字に Unicode のコードポイントを含む整数を返します。最初の文字が有効なエンコーディングではない場合は NULL を返します。

構文

```
UNICODE( nchar-string-expression )
```

パラメータ

nchar-string-expression

最初の文字が整数に変換される NCHAR 文字列。

戻り値

INT

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、整数 65536 を返します。

```
SELECT UNICODE(UNISTR( '¥u010000data' ));
```

関連情報

CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]
NCHAR 関数 [文字列] [446 ページ]
TO_CHAR 関数 [文字列] [562 ページ]
TO_NCHAR 関数 [文字列] [564 ページ]
UNISTR 関数 [文字列] [579 ページ]

1.3.2.236 UNISTR 関数 [文字列]

文字と Unicode エスケープシーケンスで構成される文字列を NCHAR 文字列に変換します。

構文

```
UNISTR( string-expression )
```

パラメータ

string-expression

変換される文字列。

戻り値

- NVARCHAR
- LONG NVARCHAR

備考

UNISTR 関数を使用すると、SQL 文で使用される CHAR 文字セットで表現できない Unicode 文字を使用できるようになります。たとえば、英語環境では UNISTR 関数を使用すると、漢字を使用できるようになります。

UNISTR 関数には N'' 定数と機能が似ています。ただし、UNISTR 関数は Unicode 文字と CHAR 文字セットの文字を使用できますが、N'' 定数は CHAR 文字セットの文字のみを使用できます。

`string-expression` には文字と Unicode のエスケープシーケンスが含まれます。Unicode のエスケープシーケンス形式は、¥uXXXX または ¥uXXXXXX です (各 X は 16 進数)。UNISTR 関数は、個々の文字と Unicode のエスケープシーケンスを対応する Unicode 文字に変換します。

6 桁の Unicode のエスケープシーケンスを使用する場合、値は Unicode のコードポイントの最大値である 10FFFF を超えません。¥u234567 などのシーケンスは 6 桁の Unicode エスケープシーケンスではありません。4 桁のシーケンス ¥u2345 の後に、文字 6 と 7 が続きます。

2 つの隣接する Unicode エスケープシーケンスが UTF-16 の代理ペアを構成する場合、出力時に 1 つの Unicode 文字に結合されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、文字列 Hello を返します。

```
SELECT UNISTR( 'Hel¥u006c¥u006F' );
```

次の例は、UTF-16 の代理ペア D800-DF02 を Unicode コードポイント 10302 に結合します。

```
SELECT UNISTR( '¥uD800¥uDF02' );
```

次の例は前の例と同等です。

```
SELECT UNISTR( '¥u010302' );
```

関連情報

文字列 [10 ページ]
CONNECTION_EXTENDED_PROPERTY 関数 [文字列] [279 ページ]
NCHAR 関数 [文字列] [446 ページ]
TO_CHAR 関数 [文字列] [562 ページ]
TO_NCHAR 関数 [文字列] [564 ページ]
UNICODE 関数 [文字列] [578 ページ]

1.3.2.237 UPPER 関数 [文字列]

文字列中のすべての文字を大文字に変換します。

構文

```
UPPER( string-expression )
```

パラメータ

string-expression

大文字に変換される文字列。

戻り値

NCHAR データで使用された場合は LONG NVARCHAR
データベース照合が UCA の場合、CHAR データで使用されるときは LONG VARCHAR
それ以外の場合、データ型は入力データ型と同じになります。
Ultra Light では、入力データ型と同じデータ型が返されます。

備考

この関数は UCASE 関数と同じです。

標準

ANSI/ISO SQL 標準

コア機能。

例

次の文を実行すると、値 CHOCOLATE が返ります。

```
SELECT UPPER( 'ChocoLate' );
```

関連情報

文字列関数 [213 ページ]
UCASE 関数 [文字列] [577 ページ]
LCASE 関数 [文字列] [415 ページ]
LOWER 関数 [文字列] [427 ページ]

1.3.2.238 USER_ID 関数 [システム]

指定したユーザ名の数値のユーザ ID を返します。

構文

```
USER_ID( [ user-name ] )
```

パラメータ

user-name

検索しているユーザ ID の数値のユーザ名。

戻り値

INTEGER

備考

`user-name` を指定しない場合は、現在のユーザ ID の数値が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザ名 GROUPO のユーザ ID の数値を返します。

```
SELECT USER_ID( 'GROUPO' );
```

関連情報

USER_NAME 関数 [システム] [583 ページ]
SUSER_ID 関数 [システム] [555 ページ]

1.3.2.239 USER_NAME 関数 [システム]

指定したユーザ ID のユーザ名を返します。

構文

```
USER_NAME( [ user-id ] )
```

パラメータ

user-id

検索しているユーザのユーザ ID。

戻り値

VARCHAR

備考

`user-id` を指定しない場合は、現在のユーザのユーザ名が返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザ ID 101 のユーザ名 GROUPO を返します。

```
SELECT USER_NAME( 101 );
```

関連情報

USER_ID 関数 [システム] [582 ページ]
SUSER_NAME 関数 [システム] [557 ページ]

1.3.2.240 UUIDTOSTR 関数 [文字列]

ユニークな識別子の値 (UUID または GUID) を文字列の値に変換します。

構文

```
UUIDTOSTR( uuid-expression )
```

パラメータ

uuid-expression

ユニークな識別子の値。

戻り値

VARCHAR

備考

一意な識別子を、xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx の形式の文字列値に変換します (x は 16 進の桁)。バイナリ値が有効な uniqueidentifier でない場合は、NULL を返します。

この関数は、UUID 値を表示するのに便利です。

注記

バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義されており、UUID 値のバイナリ表現と文字列表現の間の変換には STRTOUUID 関数と UUIDTOSTR 関数が必要でした。バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブデータ型に変更されています。データ型の変換は、データベースサーバが必要に応じて実行します。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。

Ultra Light: バージョン 9.0.2 より前に作成されたデータベースでは、UUID 値のバイナリ表現と文字列表現の間を変換するための STRTOUUID 関数と UUIDTOSTR 関数が必要です。バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブデータ型に変更されています。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、2 つのカラムを持つテーブル mytab を作成します。カラム pk は uniqueidentifier データ型で、カラム c1 は integer データ型です。それぞれに値 1 と値 2 を持つ 2 つのローをカラム c1 に挿入します。

```
CREATE TABLE mytab(
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    c1 INT );
```

```
INSERT INTO mytab( c1 ) values ( 1 );
INSERT INTO mytab( c1 ) values ( 2 );
```

次の SELECT 文を実行すると、新規に作成したテーブルのすべてのデータを返します。

```
SELECT * FROM mytab;
```

2 つのカラムと 2 つのローを持ったテーブルが表示されます。カラム pk に表示される値は、バイナリ値です。

ユニークな識別子の値を読みやすい形式に変換するには、次の文を実行します。

```
SELECT UUIDTOSTR(pk), c1 FROM mytab;
```

UUIDTOSTR 関数は、バージョン 9.0.2 以降で作成されたデータベースでは不要です。

関連情報

文字列関数 [213 ページ]
UNIQUEIDENTIFIER データ型 [175 ページ]
NEWID 関数 [その他] [447 ページ]
STRTOUUID 関数 [文字列] [548 ページ]

1.3.2.241 VAR_POP 関数 [集合]

数値式からなる母集団の統計上の平方偏差を DOUBLE として計算します。

構文

式

```
VAR_POP( numeric-expression )
```

Window 関数

```
VAR_POP( numeric-expression ) OVER ( window-spec )
```

```
window-spec: 次の「備考」の項を参照してください。
```

パラメータ

numeric-expression

ローセットで母集団ベースの平方偏差を計算する対象の式。通常、式はカラム名です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

`numeric-expression` (x) の母集団ベースの平方偏差 (s^2) は、次の式によって計算されます。

$s^2 = (1/N) * SUM(x_i - mean(x))^2$

この平方偏差には、`numeric-expression` が NULL 値のローは含まれません。グループにローが含まれていない場合は、NULL を返します。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

VAR_POP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。Window 関数として使用する場合、VAR_POP はオプションの SQL 基本機能 T611、「Elementary OLAP operations」の一部です。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、「Full value expressions」の一部です。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、「Extended set function support」もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの機能 F442、「Mixed column references in set function」がサポートされていません。このソフトウェアでは、VAR_POP 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT YEAR( ShipDate ) AS Year,
    QUARTER( ShipDate ) AS Quarter,
    AVG( Quantity ) AS Average,
    VAR_POP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	203.9021...
2000	2	27.050847	225.8109...
...	...	...	...

関連情報

集合関数 [200 ページ]
WINDOW 句 [1409 ページ]

1.3.2.242 VAR_SAMP 関数 [集合]

数値式からなるサンプルの統計上の平方偏差を DOUBLE として計算します。

構文

式

```
VAR_SAMP( numeric-expression )
```

Window 関数

```
VAR_SAMP( numeric-expression ) OVER ( window-spec )
```

`window-spec`: 次の「備考」の項を参照してください。

パラメータ

numeric-expression

ローセットでサンプルベースの平方偏差を計算する対象の式。通常、式はカラム名です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

`numeric-expression` (x) の平方偏差 (s^2) は、通常の分散と想定して、次の式によって計算されます。

$s^2 = (1/(N - 1)) * SUM(x_i - mean(x))^2$

この平方偏差には、`numeric-expression` が NULL 値のローは含まれません。グループに 0 か 1 のローが含まれている場合は、NULL を返します。

`window-spec` を使用してこの関数を指定した場合、SELECT 文で Window 関数として使用することを意味します。そのように、`window-spec` の要素は、関数構文の中 (インライン) に指定するか、または SELECT 文の WINDOW 句と組み合わせて指定できます。

標準

ANSI/ISO SQL 標準

VAR_SAMP 関数は、オプションの ANSI/ISO SQL 言語機能 T621、「Enhanced numeric functions」の一部です。Window 関数として使用する場合、VAR_SAMP はオプションの ANSI/ISO SQL 基本機能 T611、「Elementary OLAP operations」の一部です。VARIANCE 構文は標準にありません。

カラム参照ではない式に対して DISTINCT を指定する機能は、オプションの ANSI/ISO SQL 言語機能 F561、「Full value expressions」の一部です。このソフトウェアでは、ANSI/ISO SQL 言語機能 F441、「Extended set function support」もサポートされています。これにより、他のクエリブロックの式に対する外部参照など、カラム参照ではない任意の式を集合関数のオペランドで使用できます。

このソフトウェアでは、オプションの ANSI/ISO SQL 機能 F442、「Mixed column references in set function」がサポートされていません。このソフトウェアでは、VAR_SAMP 関数を含むクエリブロックからのカラム参照と外部参照の両方を、集合関数の引数に含めることはできません。

例

次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT YEAR( ShipDate ) AS Year,
    QUARTER( ShipDate ) AS Quarter,
    AVG( Quantity ) AS Average,
    VAR_SAMP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	205.1158...
2000	2	27.050847	227.0939...
...	...	...	...

関連情報

集合関数 [200 ページ]
VARIANCE 関数 [集合] [591 ページ]
WINDOW 句 [1409 ページ]

1.3.2.243 VAREXISTS 関数 [その他]

指定された名前のユーザ定義変数が存在する場合は、1 を返します。該当する変数が存在しない場合は、0 を返します。

構文

```
VAREXISTS( variable-name-string [, owner ] )
```

パラメータ

variable-name-string

テストする変数の文字列としての名前 ('myVariable' など)。

owner 変数の所有者の文字列としてのユーザ ID。`owner` は、所有しているデータベーススコープ変数でのみ指定できます。

戻り値

INT

標準

ANSI/ISO SQL 標準

標準になし。

例

次の IF 文は、start_time という変数が存在するかどうかを確認します。存在しない場合、データベースサーバはその名前の接続スコープ変数を作成し、その値を現在の時刻に設定します。

```
IF VAREXISTS( 'start_time' ) = 0 THEN
    CREATE VARIABLE start_time TIMESTAMP;
END IF;
```

```
SET start_time = CURRENT TIMESTAMP;
```

次の IF 文は、ユーザ ID jsmith が所有する run_time という名前のデータベーススコープ変数が存在するかどうかを確認します。存在しない場合、データベースサーバは変数を作成し、その値を現在の時刻に設定します。

```
IF VAREXISTS( 'run_time', 'jsmith' ) = 0 THEN
    CREATE DATABASE VARIABLE jsmith.run_time TIMESTAMP = CURRENT TIMESTAMP;
END IF;
```

関連情報

CREATE VARIABLE 文 [992 ページ]
DECLARE 文 [1010 ページ]
IF 文 [1156 ページ]

1.3.2.244 VARIANCE 関数 [集合]

VAR_SAMP のエイリアス。

関連情報

VAR_SAMP 関数 [集合] [588 ページ]

1.3.2.245 WATCOMSQL 関数 [その他]

Watcom SQL に Transact-SQL 文を書き直します。この関数は、既存の Adaptive Server Enterprise ストアドプロシージャを Watcom SQL 構文に変換するときに役立ちます。

構文

```
WATCOMSQL( sql-statement-string )
```

パラメータ

sql-statement-string

関数で Watcom SQL ダイアレクトに書き換える SQL 文。

戻り値

LONG VARCHAR

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、文字列 `'select Surname as last_name from GROUPO.Employees'` を返します。

```
SELECT WATCOMSQL( 'SELECT last_name = Surname FROM GROUPO.Employees' ) FROM
SYS.DUMMY;
```

関連情報

SQLDIALECT 関数 [その他] [536 ページ]
TRANSACTSQL 関数 [その他] [569 ページ]

1.3.2.246 WEEKS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の週の数を返します。

構文

0000-02-29 と TIMESTAMP 値の間の週数を返します。

```
WEEKS( timestamp-expression )
```

2 つの TIMESTAMP 値の間の週数を返します。

```
WEEKS( timestamp-expression, timestamp-expression )
```

週を TIMESTAMP 値に追加します。

```
WEEKS( timestamp-expression, integer-expression )
```

パラメータ

timestamp-expression

TIMESTAMP データ型の値の日付と時刻。

integer-expression

`timestamp-expression` に追加する週数。`integer-expression` が負の場合、適切な週数が `timestamp-expression` から減算されます。`integer-expression` を指定する場合は、`timestamp-expression` を DATE または TIMESTAMP として明示的にキャストしてください。

戻り値

2 つの TIMESTAMP 値を比較するときは INTEGER。

TIMESTAMP 値に週を追加する場合は TIMESTAMP。

備考

1 つの日付を指定すると、WEEKS 関数は、0000-02-29 からの週数を返します。

2 つの日付を指定すると、WEEKS 関数は、2 つの日付の間の週数を返します。WEEKS 関数は DATEDIFF 関数に似ていますが、2 つの日付間の週数を計算する場合に使用される方法はこれと同じではなく、返される結果が異なる場合があります。WEEKS の戻り値は、2 つの日付の間の日数を 7 で割り、余りを切り捨てて求められます。ただし、DATEDIFF では週の境目の数が使用されます。このため、2 つの関数から返される値が異なる場合があります。たとえば、最初の日付が金曜日で次の日付が翌月曜日の場合、WEEKS 関数は週数として 0 を返しますが、DATEDIFF 関数は 1 を返します。どちらかが優れているという問題ではないので、WEEKS と DATEDIFF との動作の違いをふまえて使用してください。

1 つの日付と整数を指定すると、WEEKS 関数は、`timestamp-expression` に、指定された整数の週数を加算します。この構文では、`timestamp-expression` を TIME、DATE、または TIMESTAMP データ型として明示的にキャストしてください。`timestamp-expression` が TIME 値の場合、現在の日付が使用されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、値 8 を返します。これは、2008-09-13 10:07:12 が、2008-07-13 06:07:12 の 8 週間後であることを示します。

```
SELECT WEEKS( '2008-07-13 06:07:12', '2008-09-13 10:07:12' );
```

次の文は、値 104792 を返します。これは、指定された日付が、0000-02-29 の 104792 週間後であることを示します。

```
SELECT WEEKS( '2008-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 2008-06-16 21:05:07.0 を返します。これは、返された日時が 2008-05-12 21:05:07 の 5 週間後であることを示します。

```
SELECT WEEKS( CAST( '2008-05-12 21:05:07' AS TIMESTAMP ), 5);
```

関連情報

DATEDIFF 関数 [日付と時刻] [305 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.3.2.247 WRITE_CLIENT_FILE 関数 [文字列]

クライアントコンピュータ上にファイルを作成して書き込みます。

構文

```
WRITE_CLIENT_FILE( filename, blob-expression [, 'A' ] )
```

パラメータ

filename

クライアントコンピュータ上のファイル名。ファイル名は、クライアントコンピュータ上で、クライアントアプリケーションの現在の作業ディレクトリとの相対パスとして解決されます。

blob-expression

クライアントコンピュータ上の `filename` に書き込まれるバイナリ文字列。

A

デフォルトでは、ファイルがすでに存在する場合は、そのファイルが上書きされます。既存のデータにデータを追加する場合は、'A' を指定します。ファイルがまだ存在しないときに 'A' を指定した場合でも、ファイルは作成されます。

戻り値

INT

備考

データベースサーバは、データベースの文字セットからクライアントの文字セットに `filename` を変換します。クライアントコンピュータ上では、`filename` がオペレーティングシステムの文字セットに変換されます。

データがバイナリ文字列であるため、データを特定の文字セットにしたり、圧縮や暗号化を行う場合は、データにこれらの処理を実行してから、WRITE_CLIENT_FILE 関数に送信する必要があります。

クライアントのソフトウェアライブラリによってファイルの書き込みが行われ、**Command Sequence** (**CmdSeq**) 通信プロトコルを使用してデータの転送が実行されます。クライアント側データの転送は、**Tabular Data Stream** (**TDS**) プロトコルではサポートされていません。

権限

クライアントコンピュータにあるファイルに書き込む場合

- WRITE CLIENT FILE システム権限が必要です。
- クライアントアプリケーションには、書き込みを行うコンピュータ上での書き込みパーミッションが必要です。
- allow_write_client_file データベースオプションが有効になっている必要があります。
- WRITE_CLIENT_FILE 機能が有効になっている必要があります。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

UNLOAD 文 [1376 ページ]
CSCONVERT 関数 [文字列] [297 ページ]
DECOMPRESS 関数 [文字列] [325 ページ]
DECRYPT 関数 [文字列] [327 ページ]

1.3.2.248 XMLAGG 関数 [集合]

XML 値のコレクションから XML 要素のフォレストを生成します。

構文

```
XMLAGG( expression [ ORDER BY order-by-expression ] )
```

パラメータ

expression

XML 値。データ型が XML でない場合、内容はエスケープされます。order-by-expression は、関数が返した要素を並べ替えます。

order-by-expression

この式の値に従って XML 要素を順序付けるために使用される式。

ORDER BY 句に定数が含まれている場合、それらの定数はオプティマイザによって解釈され、同義の ORDER BY 句に置き換えられます。たとえば、オプティマイザは ORDER BY 'a' を ORDER BY 式として解釈します。

クエリブロックに、有効な ORDER BY 句が指定された複数の集合関数が含まれているとき、それらの ORDER BY 句を単一の ORDER BY 句に論理的に結合できる場合は、そのクエリブロックを実行できます。たとえば、次の ORDER BY 句の場合は、

```
ORDER BY expression1, 'a', expression2
```

```
ORDER BY expression1, 'b', expression2, 'c', expression3
```

次の ORDER BY 句として結合されます。

```
ORDER BY expression1, expression2, expression3
```

戻り値

XML

備考

NULL である値はどれも結果から省かれます。すべての入力が NULL の場合、またはローがない場合、結果は NULL です。整形式の XML ドキュメントを要求する場合、生成された XML のルート要素が 1 つになるようにクエリを作成します。

XMLAGG を含むクエリを実行すると、BINARY、LONG BINARY、IMAGE、VARBINARY の各カラムのデータは、自動的に base64 エンコード形式で返されます。

標準

ANSI/ISO SQL 標準

XMLAGG は、オプションの ANSI/ISO SQL 言語機能 X034 の一部です。XMLAGG 関数のオプションの ORDER BY 句は、オプションの ANSI/ISO SQL 言語機能 X035 の一部です。

例

次の文は、各顧客の注文を示す XML ドキュメントを生成します。

```
SELECT XMLELEMENT( NAME "order",
                   XMLATTRIBUTES( ID AS order_id ),
                     ( SELECT XMLAGG(
                         XMLELEMENT(
                           NAME "Products",
                           XMLATTRIBUTES( ProductID, Quantity AS
"quantity_shipped" ) ) )
                        FROM SalesOrderItems soi
                        WHERE soi.ID = so.ID
                     )
                 ) AS products_ordered
FROM GROUPO.SalesOrders so
ORDER BY so.ID;
```

1.3.2.249 XMLCONCAT 関数 [文字列]

XML 要素のフォレストを生成します。

構文

```
XMLCONCAT( xml-value [, ... ] )
```

パラメータ

xml-value

連結された XML 値。

戻り値

XML

備考

XML 要素のフォレストを生成します。解析対象外の XML ドキュメントでは、フォレストは文書内の複数のルートノードを示します。NULL 値は、結果から省かれます。値がすべて NULL の場合は、NULL が返されます。XMLCONCAT 関数は、引数にプロログがあるかどうかをチェックしません。整形式の XML ドキュメントを要求する場合、1 つのルート要素が生成されるようにクエリを作成します。

データ型が XML ではない場合、要素内容は必ずエスケープされます。XMLCONCAT 関数を含むクエリを実行すると、BINARY、LONG BINARY、IMAGE、VARBINARY の各カラムのデータは、自動的に base64 エンコード形式で返されます。

標準

ANSI/ISO SQL 標準

XMLCONCAT は、オプションの ANSI/ISO SQL 言語機能 X020 の一部です。

例

次のクエリは、顧客ごとに <CustomerID>、<cust_fname>、<cust_lname> の各要素を生成します。

```
SELECT XMLCONCAT( XMLELEMENT ( NAME CustomerID, ID ),
                  XMLELEMENT( NAME cust_fname, GivenName ),
                  XMLELEMENT( NAME cust_lname, Surname )
                ) AS "Customer Information"
FROM GROUPO.Customers
WHERE ID < 120;
```

関連情報

文字列関数 [213 ページ]
XMLELEMENT 関数 [文字列] [599 ページ]
XMLFOREST 関数 [文字列] [601 ページ]

1.3.2.250 XMLELEMENT 関数 [文字列]

クエリ内の XML 要素を生成します。

構文

```
XMLELEMENT( { NAME element-name-expression } | string-expression
   [, XMLATTRIBUTES ( attribute-value-expression [ AS attribute-name ],... ) ]
   [, element-content-expression,... ]
)
```

パラメータ

element-name-expression

識別子。各ローに対して、識別子と同じ名前を持った XML 要素が生成されます。

attribute-value-expression

要素の属性。このオプションの引数を使用すると、生成された要素に属性値を指定できます。この引数は、属性の名前と内容を指定します。カラム名が attribute-value-expression の場合、属性名はデフォルトでカラム名になります。属性名は、attribute-nameargument. 引数を指定することにより変更できます。

attribute-value-expression の変数名を指定できます。

element-content-expression

要素の内容。任意の文字列式を指定できます。element-content-expression 引数は、無制限に指定でき、連結もできます。たとえば、次の SELECT 文は、値 <x>abcdef</x> を返します。

```
SELECT XMLELEMENT( NAME x, 'abc', 'def' );
```

戻り値

XML

備考

NULL の要素値と NULL の属性値は、結果から省かれます。要素名と属性名の大文字と小文字は、クエリから取得されます。

データ型が XML ではない場合、要素内容は必ずエスケープされます。無効な要素名と属性名も引用符で囲まれます。次の文を例にとります。

```
SELECT XMLELEMENT('H1', f_get_page_heading() );
```

関数 f_get_page_heading が RETURNS LONG VARCHAR または RETURNS VARCHAR(1000) と定義されている場合、結果は HTML でエンコーディングされます。

```
CREATE FUNCTION f_get_page_heading() RETURNS LONG VARCHAR
   BEGIN
      RETURN ('<B>My Heading</B>');
   END;
```

上記の SELECT 文は次の値を返します。

```
<H1>&lt;B&gt;My Heading&lt;/B&gt;</H1>
```

関数が RETURNS XML と宣言されている場合、上記の SELECT 文は次の値を返します。

```
<H1><B>My Heading</B></H1>
```

XMLELEMENT 関数をネストして階層を作成できます。同じレベルのドキュメント階層で異なる要素を返すには、XMLFOREST 関数を使用します。

XMLELEMENT 関数を含むクエリを実行すると、BINARY、LONG BINARY、IMAGE、VARBINARY の各カラムのデータは、自動的に base64 エンコード形式で返されます。

標準

ANSI/ISO SQL 標準

XMLELEMENT は、オプションの ANSI/ISO SQL 言語機能 X031 の一部です。NAME キーワードを省略し、第 1 引数に文字列式を使用することは、標準にはありません。このソフトウェアは、XMLELEMENT 関数でオプションの OPTION 句をサポートしていません。

例

次の例は、結果セットの各製品に対して <item_name> 要素を生成します。ここでは、製品名が要素の内容です。

```
SELECT ID, XMLELEMENT( NAME item_name, p.Name )
FROM Products p
WHERE ID > 400;
```

次の例は、<A HREF="http://www.sap.com/" TARGET="_top">SAP web site</A> を返します。

```
SELECT XMLELEMENT(
   'A',
   XMLATTRIBUTES( 'http://www.sap.com/'
      AS "HREF", '_top' AS "TARGET"),
   'SAP web site'
);
```

次の例は、<table><tbody><tr align="center" valign="top"><td>Cell 1 info</td><td>Cell 2 info</td></tr></tbody></table> を返します。

```
SELECT XMLELEMENT( name "table",
         XMLELEMENT( name "tbody",
           XMLELEMENT( name "tr",
             XMLATTRIBUTES('center' AS "align", 'top' AS "valign"),
```

```
            XMLELEMENT( name "td", 'Cell 1 info' ),
            XMLELEMENT( name "td", 'Cell 2 info' )
          )
        )
      );
```

次の例は、'<x>abcdef</x>','<custom_element>abcdef</custom_element>' を返します。

```
CREATE VARIABLE @my_element_name VARCHAR(200);
SET @my_element_name = 'custom_element';
SELECT XMLELEMENT( NAME x, 'abc', 'def' ),
   XMLELEMENT( @my_element_name,'abc', 'def' );
```

関連情報

文字列関数 [213 ページ]
XMLCONCAT 関数 [文字列] [597 ページ]
XMLFOREST 関数 [文字列] [601 ページ]

1.3.2.251 XMLFOREST 関数 [文字列]

XML 要素のフォレストを生成します。

構文

```
XMLFOREST( element-content-expression [ AS element-name ],... )
```

パラメータ

element-content-expression

文字列。指定された各 element-content-expression 引数に対して、要素が生成されます。element-content-expression の値が要素の内容になります。たとえば、この引数に従業員テーブルの EmployeeID カラムを指定すると、テーブルの各値に対して EmployeeID 値を含む <EmployeeID> 要素が生成されます。

要素に element-content-expression 以外の名前を割り当てる場合は、element-name 引数を指定します。それ以外の場合、要素名はデフォルトで element-content-expression 名になります。

戻り値

XML

備考

XML 要素のフォレストを生成します。解析対象外の XML ドキュメントでは、フォレストはドキュメント内の複数のルートノードを示します。XMLFOREST 関数の引数がすべて NULL の場合は、NULL 値が返されます。一部の値が NULL の場合、NULL 値は結果から省かれます。データ型が XML ではない場合、要素内容は必ず引用符で囲まれます。XMLFOREST 関数を使用して属性を指定することはできません。生成された要素に属性を指定する場合は、XMLELEMENT 関数を使用します。

データ型が XML でない場合、要素名はエスケープされます。

整形式の XML ドキュメントを要求する場合、1 つのルート要素が生成されるようにクエリを作成します。

XMLFOREST を含むクエリを実行すると、BINARY、LONG BINARY、IMAGE、VARBINARY の各カラムのデータは、自動的に base64 エンコード形式で返されます。

標準

ANSI/ISO SQL 標準

XMLFOREST は、オプションの ANSI/ISO SQL 言語機能 X032 の一部です。このソフトウェアは、XMLFOREST 関数でオプションの XMLNAMESPACES 句または OPTION 句をサポートしていません。

例

次の文は、各従業員の姓と名前に対して XML 要素を生成します。

```
SELECT EmployeeID,
       XMLFOREST( GivenName, Surname )
       AS "Employee Name"
FROM Employees;
```

関連情報

文字列関数 [213 ページ]
XMLELEMENT 関数 [文字列] [599 ページ]
XMLCONCAT 関数 [文字列] [597 ページ]

1.3.2.252 XMLGEN 関数 [文字列]

XQuery コンストラクタに基づいて XML 値を生成します。

構文

```
XMLGEN( xquery-constructor, content-expression [ AS variable-name ],... )
```

パラメータ

xquery-constructor

XQuery コンストラクタ。XQuery コンストラクタは、XQuery 言語で定義された項目です。XQuery 式に基づいて XML 要素を構成するための構文を提供します。`xquery-constructor` 引数は、1 つ以上の変数参照を持つ整形式の XML ドキュメントにしてください。変数参照は中かっこで囲まれます。プレフィクス $ が必要で、前後に空白スペースは不要です。例:

```
SELECT XMLGEN( '<a>{$x}</a>', 1 AS x );
```

content-expression

変数。複数の `content-expression` 引数を指定できます。オプションの `variable-name` 引数は、変数の名前を付ける際に使用されます。例:

```
SELECT XMLGEN( '<emp EmployeeID="{$EmployeeID}"><StartDate>{$x}</StartDate></
emp>',
               EmployeeID, StartDate
               AS x )
FROM GROUPO.Employees;
```

戻り値

XML

備考

XQuery 仕様で定義されている計算コンストラクタは、XMLGEN 関数でサポートされません。

XMLGEN 関数を含むクエリを実行すると、BINARY、LONG BINARY、IMAGE、VARBINARY の各カラムのデータは、自動的に base64 エンコード形式で返されます。

データ型が XML ではない場合、要素内容は必ずエスケープされます。無効な XML 要素名と属性名もエスケープされます。

標準

ANSI/ISO SQL 標準

標準になし。XMLGEN 関数の機能は、ANSI SQL XMLDOCUMENT 関数に似ています。

例

次の例は、各従業員の <emp> 要素、<Surname> 要素、<GivenName> 要素、<StartDate> 要素を生成します。

```
SELECT XMLGEN( '<emp EmployeeID="{$EmployeeID}">
```

```
                   <Surname>="{$Surname}"</Surname>
                   <GivenName>="{$GivenName}"</GivenName>
                   <StartDate>="{$StartDate}"</StartDate>
                </emp>',
                EmployeeID,
                Surname,
                GivenName,
                StartDate
              ) AS employee_list
FROM GROUPO.Employees;
```

関連情報

文字列関数 [213 ページ]

1.3.2.253 YEAR 関数 [日付と時刻]

TIMESTAMP 引数の年部分を返します。

構文

```
YEAR( timestamp-expression)
```

パラメータ

timestamp-expression

TIMESTAMP 値。

戻り値

SMALLINT

備考

指定された TIMESTAMP 値の年部分が SMALLINT として返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、値 2001 を返します。

```
SELECT YEAR( '2001-09-12' );
```

1.3.2.254 YEARS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の年数を返します。

構文

0000 年と TIMESTAMP 値の間の年数を返します。

```
YEARS( timestamp-expression )
```

2 つの TIMESTAMP 値の間の年数を返します。

```
YEARS( timestamp-expression, timestamp-expression )
```

年を TIMESTAMP 値に追加します。

```
YEARS( timestamp-expression, integer-expression )
```

パラメータ

timestamp-expression

TIMESTAMP データ型の値の日付と時刻。

integer-expression

`timestamp-expression` に (SMALLINT 値として) 追加される年数。`integer-expression` が負の場合、適切な年数が `timestamp-expression` から減算されます。`integer-expression` を指定する場合は、`timestamp-expression` を DATE、TIME、または TIMESTAMP 値として明示的にキャストしてください。`timestamp-expression` が TIME の場合、現在の年が使用されます。

戻り値

2 つの TIMESTAMP 値を比較するときは SMALLINT。

TIMESTAMP 値に年を追加する場合は TIMESTAMP。

備考

YEARS の値は、2 つの日付の間に年の最初の日がいくつあるかをカウントして求められます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文はどちらも -4 を返します。

```
SELECT YEARS( '1998-07-13 06:07:12',
              '1994-03-13 08:07:13' );
```

```
SELECT DATEDIFF( year,
   '1998-07-13 06:07:12',
   '1994-03-13 08:07:13' );
```

次の文は、1998 を返します。

```
SELECT YEARS( '1998-07-13 06:07:12' )
SELECT DATEPART( year, '1998-07-13 06:07:12' );
```

次の文は、指定した日付の 300 年後を返します。

```
SELECT YEARS( CAST( '1998-07-13 06:07:12' AS TIMESTAMP ), 300 )
```

```
SELECT DATEADD( year, 300, '1998-07-13 06:07:12' );
```

関連情報

DATEDIFF 関数 [日付と時刻] [305 ページ]
DATEADD 関数 [日付と時刻] [304 ページ]
CAST 関数 [データ型変換] [264 ページ]

1.3.2.255 YMD 関数 [日付と時刻]

指定した年、月、日に相当する日付値を返します。引数は -32768 ～ 32767 の INTEGER です。

構文

```
YMD( smallint-expression1, smallint-expression2, smallint-expression3 )
```

パラメータ

smallint-expression1

年

smallint-expression2

月の数。月が 1 ～ 12 の範囲外である場合は、年が相応に調整されます。

smallint-expression3

日の数。任意の整数を指定でき、それに応じて日付が調整されます。

戻り値

DATE

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文を実行すると、値 1998/06/12 が返ります。

```
SELECT YMD( 1998, 06, 12 );
```

値が通常の範囲から外れている場合、それに応じて日付が調整されます。たとえば、次の文は DATE 値 2000-03-01 を返します。

```
SELECT YMD( 1999, 15, 1 );
```

1.4 SQL 文

SQL 文のマニュアルでは、いくつかの表記規則が使用されています。

このセクションの内容:

1.4.1 一般的な SQL 構文要素

多くの SQL 構文で使われる言語要素について説明します。

column-name

カラム名を表す識別子。

condition

TRUE、FALSE、または UNKNOWN の評価を行う式。

connection-name

アクティブな接続の名前を表す文字列。

data-type

記憶データ型。

expression

式。構文に含まれる式の一般的な例を挙げると、カラム名があります。

filename

ファイル名を指定した文字列。

hostvar

先頭にコロンがあるホスト変数として宣言される C 言語変数の 1 つ。

materialized-view-name

マテリアライズドビュー名を表す識別子。

number

任意の順序に並んだ数字。小数点以下の位があったり、負の記号を付けたりできます。また、数字の後に E と指数を付けることもできます。例:

```
42
```

```
-4.038
.001
3.4e10
1e-10
```

owner

データベースオブジェクトの所有者であるユーザ ID を表す識別子。

query-block

クエリブロックは、単純なクエリ式、または ORDER BY 句を使用したクエリ式です。

query-expression

クエリ式は、SELECT、UNION、INTERSECT、または EXCEPT ブロック (つまり ORDER BY、WITH、FOR、FOR XML、または OPTION 句を含まない文) で構成できます。これらのブロックを組み合わせて構成することも可能です。

role-name

外部キーの役割名を表す識別子。概念データベースモデルで、ある視点からの関係を説明する動詞または句を指します。各関係は 2 つのロールを使用して表すことができます。"contains (A は B を含む)" や "is a member of (B は A のメンバー)" などのロールがあります。

savepoint-name

セーブポイント名を表す識別子。

search-condition

TRUE、FALSE、または UNKNOWN の評価を行う条件。

special-value

特別値。

statement-label

ループまたは複合文のラベルを表す識別子。

statement-list

それぞれがセミコロンで終わる SQL 文のリスト。

string-expression

文字列に解決される式。

table-list

テーブル名のリスト。相関名が含まれることもあります。

table-name

テーブル名を表す識別子。

userid

ユーザ名を表す識別子。

variable-name

変数名を表す識別子。

window-name

ウィンドウ名を表す識別子。ウィンドウ定義に関する構文に使用されます (たとえば、WINDOW 句、RANK などの Window 関数)。

関連情報

SQL 変数 [118 ページ]
文字列 [10 ページ]
特別値 [83 ページ]
真理値探索条件 [80 ページ]
SQL 文の式 [32 ページ]
SQL データ型 [124 ページ]
探索条件 [53 ページ]
FROM 句 [1112 ページ]
識別子 [6 ページ]

1.4.2 SQL 構文の表記規則

SQL 構文の説明に使用する表記規則を以下に示します。

キーワード

SQL キーワードはすべて次の例に示す SQL 文 ALTER TABLE のように大文字で表記します。

```
ALTER TABLE [ owner.]table-name
```

プレースホルダ (変数値または変数とも呼ばれる)

適切な識別子または式で置き換えられる項目は、次の例に示す owner や table-name のように斜体で表記します。

```
ALTER TABLE [ owner.]table-name
```

プレースホルダが複数の語の場合、それらの語はパラメータと区別するためにアンダースコアではなくハイフンで区切られます (table-name など)。

パラメータ

関数などのオブジェクトのパラメータは、ソフトウェアに設定されます。パラメータを指定する場合は、ソフトウェアで使用すると表現と一致させる必要があります。パラメータが複数の語の場合、それらの語はアンダースコアで区切られます。構文マニュアルにあるパラメータは、太字で表示されます (**table_name** など)。

句の順序

オプション句の順序が SQL 文の構文において重要な場合、それらの句はすでに記述するべき順序で構文にリストされています。ベストプラクティスとして、また、文の特定の句のヘルプを見つけるのに役立てるため、マニュアルに記述されている句の順序に従ってください。

```
CREATE SYNCHRONIZATION SUBSCRIPTION [ subscription-name ]
TO publication-name
[ FOR ml-username, ... ]
...
```

オプション部分

文のオプション部分は角カッコで囲みます。例:

```
RELEASE SAVEPOINT [ savepoint-name ]
```

角カッコは、savepoint-name への値の指定がオプションであることを示します。角カッコは入力しないでください。

また、キーワード部分を角カッコで囲む場合もあります。たとえば、次の構文は COMMIT TRAN または COMMIT TRANSACTION を使用できることを示します。

```
COMMIT TRAN[SACTION] ...
```

同様に、次の構文は COMMIT または COMMIT WORK を使用できることを示します。

```
COMMIT [ WORK ]
```

構文の一部の繰り返し表現

繰り返し可能な項目は、省略記号 (ピリオド 3 つ) と、任意の繰り返しを示す適切なカッコを付けて表されます。次のスタイルは、両方が構文マニュアルで使用され、意味的に同等です。ただし、複雑な構文での読みやすさを向上させるため、一方のスタイルを他方のスタイルよりも優先して使用できます。

```
ADD column-definition [ column-constraint, ... ]
```

```
ADD column-definition [ column-constraint [,...] ]
```

値のセット

項目リストから 1 つだけ選択する場合は、項目間をパイプ記号で区切り、リスト全体をカッコで囲みます。

```
{ ON | OFF }
[ ASC | DESC ]
```

最初の例では、ON と OFF のどちらかの値を 1 つを選択する必要があります。2 番目の例では、ASC と DESC のどちらか 1 つを選択しても、選択しなくてもかまいません。通常、太字で表示されていない場合や入力が必要な文字列の一部でない場合は、カッコは入力しないでください。

選択肢

オプションの中の 1 つを必ず選択する場合は、選択肢を中カッコで囲みます。

```
[ QUOTES { ON | OFF } ]
```

この場合、ON または OFF のどちらかを必ず選択します。角カッコと中カッコは入力しないでください。

カッコ

中カッコ ({ }) は、値を必ず 1 つ選択するセットを示し、明示的にしていされている場合を除き、入力することができません。角カッコ ([ ]) は、そのカッコ内のコンテンツがオプションであることを示します。角カッコを入力することはできません。丸カッコ (( )) は構文の一部で、入力が必要です。

1.4.3 文の適応性インジケータ

一部の文には、タイトルの後ろに角カッコで囲まれたインジケータが付き、文が使用される場所を示します。

このインジケータは以下のとおりです。

[ESQL]

Embedded SQL で使用される文。

[Interactive SQL]

Interactive SQL 専用の文。

[SP]

ストアドプロシージャ、トリガ、またはバッチで使用される文。

[T-SQL]

Adaptive Server Enterprise との互換性のために実装されている文。Transact-SQL フォーマット以外のストアドプロシージャで使用できないことがあります。また、Transact-SQL の互換性が問題にならないかぎり、ANSI/ISO SQL 標準に近い代わりの文が推奨される場合もあります。

[外部呼び出し]

外部ファンクションおよび外部プロシージャの呼び出しで使用される文。

[Mobile Link]

Mobile Link クライアント専用の文。

[SQL Remote]

SQL Remote 専用の文。

[Web サービス]

Web サービスクライアント専用の文。

カッコが 2 つある場合、文はどちらの環境でも使用できます。たとえば、[ESQL][SP] は Embedded SQL でもストアドプロシージャでも使用できる文であることを意味します。

1.4.4 アルファベット順 SQL 文リスト

このソフトウェアでは、多数の SQL 文がサポートされています。

このセクションの内容:

ALTER EXTERNAL ENVIRONMENT 文 [647 ページ]
Java、PHP、Perl などの外部環境のロケーションを指定します。

ALTER FUNCTION 文 [649 ページ]
関数を変更します。

ALTER INDEX 文 [652 ページ]
インデックス、プライマリキー、または外部キーの名前を変更したり、インデックスのクラスタ化された内容を変更したりします。

ALTER LDAP SERVER 文 [654 ページ]
LDAP サーバ設定オブジェクトを変更します。

ALTER LOGIN POLICY 文 [657 ページ]
既存のログインポリシーを変更します。

ALTER MATERIALIZED VIEW 文 [659 ページ]
マテリアライズドビューを変更します。

ALTER MIRROR SERVER 文 [662 ページ]
ミラーサーバの属性を変更します。

ALTER ODATA PRODUCER 文 [666 ページ]
OData プロデューサを変更します。

ALTER PROCEDURE 文 [669 ページ]
プロシージャを変更します。

ALTER PUBLICATION 文 [Mobile Link] [SQL Remote] [672 ページ]
パブリケーションを変更します。Mobile Link では、パブリケーションが SQL Anywhere リモートデータベース内の同期データを識別します。SQL Remote では、統合データベース内とリモートデータベース内の両方のレプリケートされたデータがパブリケーションによって識別されます。

ALTER REMOTE MESSAGE TYPE 文 [SQL Remote] [674 ページ]
特定のメッセージシステム、または作成したメッセージタイプに対する、パブリッシャのメッセージシステム、またはパブリッシャのアドレスを変更します。

ALTER ROLE 文 [675 ページ]
互換ロールをユーザ定義ロールに移行してから、その互換ロールを削除します。

ALTER SEQUENCE 文 [677 ページ]
シーケンスを変更します。

ALTER SERVER 文 [679 ページ]
リモートサーバの属性を変更します。

ALTER SERVICE 文 [HTTP Web サービス] [683 ページ]
既存の HTTP Web サービスを変更します。

ALTER SERVICE 文 [SOAP Web サービス] [688 ページ]
既存の HTTP を介した SOAP または DISH サービスを変更します。

ALTER SPATIAL REFERENCE SYSTEM 文 [694 ページ]
既存の空間参照系の設定を変更します。空間参照系を変更する前に、注意事項について「備考」の項を参照してください。

ALTER STATISTICS 文 [698 ページ]
テーブルの 1 つのカラムまたは複数のカラムに関する統計情報を自動的に更新するかどうかを制御します。

ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link] [700 ページ]
SQL Anywhere 同期プロファイルを変更します。同期プロファイルは、同期の制御に使用できる同期オプションの名前付きコレクションです。

ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [701 ページ]
SQL Anywhere リモートデータベースで同期サブスクリプションのプロパティを変更します。

ALTER SYNCHRONIZATION USER 文 [Mobile Link] [704 ページ]
SQL Anywhere リモートデータベースで Mobile Link ユーザのプロパティを変更します。

ALTER TABLE 文 [705 ページ]
テーブルの定義を変更したり、従属ビューを無効にしたりします。

ALTER TEXT CONFIGURATION 文 [720 ページ]
テキスト設定オブジェクトを変更します。

ALTER TEXT INDEX 文 [724 ページ]
テキストインデックスの定義を変更します。

ALTER TIME ZONE 文 [726 ページ]
タイムゾーンオブジェクトを削除します。

ALTER TRACE EVENT SESSION 文 [728 ページ]
トレースイベントのセッションへの追加およびセッションからの削除、ターゲットのセッションへの追加およびセッションからの削除、またはトレースセッションの開始および停止を実行します。

ALTER TRIGGER 文 [730 ページ]
トリガの定義を修正したものに置き換えます。新しいトリガの定義全体を ALTER TRIGGER 文にインクルードします。この文は、SAP IQ のカタログストアテーブルのみに適用されます。

ALTER USER 文 [731 ページ]
ユーザ設定を変更します。

ALTER VIEW 文 [734 ページ]
ビュー定義を修正したものに置き換えます。

ATTACH TRACING 文 (廃止) [737 ページ]
診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。ATTACH TRACING 文は、診断トレースセッションを開始します (診断テーブルへの診断情報の送信を開始します)。

BACKUP DATABASE 文 [739 ページ]
データベースとトランザクションログをバックアップします。

BEGIN 文 [745 ページ]
複合文を指定します。

BEGIN 文 [TSQL] [748 ページ]
複合文を指定します。

BEGIN PARALLEL WORK 文 [750 ページ]
複数の論理プロセッサを使用することで、コンピュータ上で CREATE INDEX または LOAD TABLE 文のリストを実行するときの時間を短縮します。

BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
ユーザ定義のトランザクションを開始します。

BEGIN SNAPSHOT 文 [754 ページ]
スナップショットアイソレーショントランザクションで使用するスナップショットを指定の期間に開始します。

BREAK 文 [T-SQL] [755 ページ]
複合文またはループから出ます。

CALL 文 [756 ページ]
プロシージャを呼び出します。

CASE 文 [760 ページ]
複数の状況に基づいて実行パスを選択します。

CASE 文 [T-SQL] [763 ページ]
複数の状況に基づいて実行パスを選択します。

CHECKPOINT 文 [765 ページ]
データベースにチェックポイントを設定します。

CLEAR 文 [Interactive SQL] [766 ページ]
Interactive SQL で開いているすべての結果セットを閉じます。

CLOSE 文 [ESQL] [SP] [767 ページ]
カーソルを閉じます。

COMMENT 文 [769 ページ]
データベースオブジェクトに対するコメントをシステムテーブルに格納します。

COMMIT 文 [771 ページ]
データベースを永続的に変更したり、ユーザ定義のトランザクションを終了したりします。

CONFIGURE 文 [Interactive SQL] [773 ページ]
Interactive SQL のオプションウィンドウを開きます。

CONNECT 文 [ESQL] [Interactive SQL] [774 ページ]
データベースへの接続を確立します。

CONTINUE 文 [778 ページ]
ループを再開します。

CREATE CERTIFICATE 文 [779 ページ]
指定されたファイルまたは文字列からデータベース内の証明書を追加したり置き換えます。証明書を作成するには、証明書作成ユーティリティ (createcert) を使用します。

CREATE DATABASE 文 [781 ページ]
データベースを作成します。

CREATE DBSPACE 文 [789 ページ]
新しいデータベース領域を定義し、関連するデータベースファイルを作成します。

CREATE DECRYPTED DATABASE 文 [790 ページ]
すべてのトランザクションログおよび DB 領域を含む既存のデータベースの、復号化されたコピーを作成します。

CREATE DECRYPTED FILE 文 [792 ページ]
強力な暗号化が施されているデータベースの復号化されたコピーを作成します。この文を使用して、トランザクションログ、トランザクションログミラー、および DB 領域の復号化されたコピーを作成できます。

CREATE DOMAIN 文 [794 ページ]
データベース内にドメインを作成します。

CREATE ENCRYPTED DATABASE 文 [798 ページ]
すべてのトランザクションログおよび DB 領域を含む既存のデータベースの暗号化されたコピーを作成するか、テーブル暗号化が有効にされた既存のデータベースのコピーを作成します。

CREATE ENCRYPTED FILE 文 [801 ページ]

CREATE ENCRYPTED DATABASE 文を使えない場合に、強力に暗号化されたデータベースファイルのコピーを作成します。また、トランザクションログファイル、トランザクションログミラーファイル、DB 領域ファイルの暗号化されたコピーも作成します。

CREATE EVENT 文 [805 ページ]

イベントとイベントに関連付けて定義済みアクションを自動化するイベントハンドラを定義し、スケジュールされたアクションを定義します。

CREATE EXISTING TABLE 文 [812 ページ]

リモートサーバ上の既存のオブジェクトを表す新しいプロキシテーブルを作成します。

CREATE EXTERNLOGIN 文 [816 ページ]

リモートサーバとの通信に使用される代替ログイン名とパスワードを割り当てます。

CREATE FUNCTION 文 [外部呼び出し] [818 ページ]

ネイティブ関数または外部関数へのインタフェースを作成します。

CREATE FUNCTION 文 [Web サービス] [826 ページ]

HTTP 要求または SOAP over HTTP 要求を行う Web クライアント関数を作成します。

CREATE FUNCTION 文 [837 ページ]

データベースにユーザ定義 SQL 関数を作成します。

CREATE INDEX 文 [842 ページ]

指定したテーブル、またはマテリアライズドビューにインデックスを作成します。

CREATE LDAP SERVER 文 [847 ページ]

LDAP サーバ設定オブジェクトを作成します。

CREATE LOCAL TEMPORARY TABLE 文 [850 ページ]

プロシージャが完了した後も、明示的に削除されるか接続が終了するまで保持されるローカルテンポラリテーブルをプロシージャ内で作成します。

CREATE LOGIN POLICY 文 [852 ページ]

ログインポリシーを作成します。

CREATE MATERIALIZED VIEW 文 [854 ページ]

マテリアライズドビューを作成します。

CREATE MESSAGE 文 [T-SQL] [856 ページ]

メッセージ番号/メッセージ文字列のペアを作成します。このメッセージ番号は PRINT 文と RAISERROR 文で使用できます。

CREATE MIRROR SERVER 文 [858 ページ]

データベースミラーリングまたは読み込み専用のスケールアウトに使用されているミラーサーバを作成するか、置き換えます。

CREATE MUTEX 文 [863 ページ]

ファイルやプロシージャなどのリソースをロックするために使用できるミューテックス (ロック) を作成または置換します。

CREATE ODATA PRODUCER 文 [865 ページ]

OData プロデューサを作成します。

CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]

ネイティブプロシージャまたは外部プロシージャへのインタフェースを作成します。

CREATE PROCEDURE 文 [Web サービス] [878 ページ]
HTTP サーバへの HTTP 要求または SOAP 要求を作成するユーザ定義の Web クライアントプロシージャを作成します。

CREATE PROCEDURE 文 [T-SQL] [888 ページ]
Adaptive Server Enterprise 互換の方法で、データベース内にプロシージャを作成します。

CREATE PROCEDURE 文 [891 ページ]
データベースにユーザ定義 SQL プロシージャを作成します。

CREATE PUBLICATION 文 [Mobile Link] [SQL Remote] [899 ページ]
パブリケーションを作成します。Mobile Link では、パブリケーションが SQL Anywhere リモートデータベース内の同期データを識別します。SQL Remote では、統合データベース内とリモートデータベース内の両方のレプリケートされたデータがパブリケーションによって識別されます。

CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote] [903 ページ]
データベースからの出力メッセージのメッセージリンクとリターンアドレスを識別します。

CREATE ROLE 文 [905 ページ]
ロールの作成または置換、ユーザ拡張ロールの作成、またはロール管理者の変更を行います。

CREATE SCHEMA 文 [907 ページ]
データベースユーザのテーブルとビューのコレクションを作成します。

CREATE SEMAPHORE 文 [909 ページ]
セマフォを作成または置換し、カウンタの初期値を設定します。セマフォとはロックメカニズムであり、カウンタを使用して、外部ライブラリやプロシージャなどのリソースの可用性を伝達および制御します。

CREATE SEQUENCE 文 [911 ページ]
複数のテーブル間でユニークな値となるプライマリキー値、およびテーブルのデフォルト値を生成するために使用できるシーケンスを作成します。

CREATE SERVER 文 [914 ページ]
リモートサーバまたはディレクトリアクセスサーバの作成

CREATE SERVICE 文 [HTTP Web サービス] [921 ページ]
新しい HTTP Web サービスを作成します。

CREATE SERVICE 文 [SOAP Web サービス] [927 ページ]
新しい HTTP または DISH を介した SOAP サービスを作成します。

CREATE SPATIAL REFERENCE SYSTEM 文 [933 ページ]
空間参照系を作成するか、置き換えます。

CREATE SPATIAL UNIT OF MEASURE 文 [940 ページ]
空間測定単位を作成するか、置き換えます。

CREATE STATISTICS 文 [942 ページ]
オプティマイザが使用するカラムの統計情報を再作成し、ISYSCOLSTAT システムテーブルに格納します。

CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]
パブリケーションに対してユーザのサブスクリプションを作成します。

CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link] [946 ページ]
SQL Anywhere 同期プロファイルを作成します。

CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [948 ページ]

SQL Anywhere リモートデータベースで、Mobile Link ユーザとパブリケーションとの間のサブスクリプションを作成します。

CREATE SYNCHRONIZATION USER 文 [Mobile Link] [950 ページ]
SQL Anywhere リモートデータベースで Mobile Link ユーザを作成します。

CREATE TABLE 文 [952 ページ]
データベースに新しいテーブルを作成し、オプションでリモートサーバ上にテーブルを作成します。

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
データベースが停止するまで持続するユーザトレースイベントを作成します。

CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ユーザトレースイベントセッションを作成します。

CREATE TEXT CONFIGURATION 文 [975 ページ]
テキストインデックスの構築と更新に使用するテキスト設定オブジェクトを作成します。

CREATE TEXT INDEX 文 [976 ページ]
テキストインデックスを作成します。

CREATE TIME ZONE 文 [979 ページ]
データベースがサーバのタイムゾーンとは異なるタイムゾーンをシミュレートする場合に使用する、タイムゾーンオブジェクトを作成します。

CREATE TRIGGER 文 [982 ページ]
テーブル内にトリガを作成します。

CREATE TRIGGER 文 [T-SQL] [989 ページ]
Adaptive Server Enterprise 互換の方法で、データベース内に新しいトリガを作成します。

CREATE USER 文 [990 ページ]
データベースユーザまたはグループを作成します。

CREATE VARIABLE 文 [992 ページ]
接続スコープ変数またはデータベーススコープ変数を作成します。

CREATE VIEW 文 [995 ページ]
データベース上にビューを作成します。

DEALLOCATE DESCRIPTOR 文 [ESQL] [999 ページ]
SQL 記述子領域に関連付けられているメモリを解放します。

DEALLOCATE 文 [1000 ページ]
この文は SQL Anywhere では機能しないので、無視されます。これは Adaptive Server Enterprise と Microsoft SQL Server との互換性のために用意されています。この文の詳細については、Adaptive Server Enterprise または Microsoft SQL Server のマニュアルを参照してください。

宣言セクション [ESQL] [1000 ページ]
Embedded SQL プログラムでホスト変数を宣言します。ホスト変数を使って、データベースとデータを交換します。

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
カーソルを宣言します。

DECLARE LOCAL TEMPORARY TABLE 文 [1007 ページ]
ローカルテンポラリテーブルを宣言します。

DECLARE 文 [1010 ページ]
複合文 (BEGIN...END) 内で SQL 変数 (接続スコープ) または例外を宣言します。

DELETE 文 (位置付け) [ESQL] [SP] [1014 ページ]
カーソルの現在位置のデータを削除します。

DELETE 文 [1016 ページ]
データベースからローを削除します。

DESCRIBE 文 [ESQL] [1019 ページ]
データベースから取り出したデータを格納するために必要なホスト変数、またはデータベースにデータを渡すために必要なホスト変数に関する情報を取得します。

DESCRIBE 文 [Interactive SQL] [1023 ページ]
指定されたデータベースオブジェクトに関する情報を返します。

DETACH TRACING 文 (廃止) [1027 ページ]
診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。DETACH TRACING セッションによって、診断トレースセッションは終了します。

DISCONNECT 文 [ESQL] [Interactive SQL] [1028 ページ]
データベースへの接続を切断します。

DROP CERTIFICATE 文 [1029 ページ]
データベースから証明書を削除します。

DROP CONNECTION 文 [1030 ページ]
データベースへのユーザの接続を削除します。

DROP DATABASE 文 [1032 ページ]
データベースに関連付けられているすべてのデータベースファイルを削除します。

DROP DATATYPE 文 [1033 ページ]
データベースからデータ型を削除します。

DROP DBSPACE 文 [1034 ページ]
データベースから DB 領域を削除します。

DROP DOMAIN 文 [1035 ページ]
データベースからドメイン (データ型) を削除します。

DROP EVENT 文 [1036 ページ]
データベースからイベントを削除します。

DROP EXTERNLOGIN 文 [1038 ページ]
データベースから外部ログインを削除します。

DROP FUNCTION 文 [1039 ページ]
データベースから関数を削除します。

DROP INDEX 文 [1040 ページ]
データベースからインデックスを削除します。

DROP LDAP SERVER 文 [1042 ページ]
LDAP サーバ設定オブジェクトを削除します。

DROP LOGIN POLICY 文 [1043 ページ]
ログインポリシーを削除します。

DROP MATERIALIZED VIEW 文 [1045 ページ]
データベースからマテリアライズドビューを削除します。

DROP MESSAGE 文 [1046 ページ]
データベースからメッセージを削除します。

DROP MIRROR SERVER 文 [1047 ページ]
ミラーサーバを削除します。

DROP MUTEX 文 [1048 ページ]
指定されたミューテックスを削除します。

DROP ODATA PRODUCER 文 [1050 ページ]
OData プロデューサを削除します。

DROP PROCEDURE 文 [1051 ページ]
データベースからプロシージャを削除します。

DROP PUBLICATION 文 [Mobile Link] [SQL Remote] [1052 ページ]
パブリケーションを削除します。

DROP REMOTE MESSAGE TYPE 文 [SQL Remote] [1053 ページ]
データベースからメッセージタイプの定義を削除します。

DROP REMOTE CONNECTION 文 [1055 ページ]
リモートサーバへのリモーデータアクセスの接続を切断します。

DROP ROLE 文 [1056 ページ]
データベースからロールを削除するか、ユーザ拡張ロールを変換して通常のユーザに戻します。

DROP SEMAPHORE 文 [1059 ページ]
セマフォを削除します。

DROP SEQUENCE 文 [1060 ページ]
シーケンスを削除します。

DROP SERVER 文 [1061 ページ]
カタログからリモートサーバを削除します。

DROP SERVICE 文 [1063 ページ]
Web サービスを削除します。

DROP SPATIAL REFERENCE SYSTEM 文 [1064 ページ]
空間参照系を削除します。

DROP SPATIAL UNIT OF MEASURE 文 [1065 ページ]
空間測定単位を削除します。

DROP STATEMENT 文 [ESQL] [1066 ページ]
文のリソースを解放します。

DROP STATISTICS 文 [1067 ページ]
指定したカラムのすべての統計情報を消去します。

DROP SUBSCRIPTION 文 [SQL Remote] [1069 ページ]
パブリケーションからユーザのサブスクリプションを削除します。

DROP SYNCHRONIZATION PROFILE 文 [Mobile Link] [1070 ページ]
SQL Anywhere 同期プロファイルを削除します。

DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [1071 ページ]
リモートデータベース内の同期サブスクリプションを削除します。

DROP SYNCHRONIZATION USER 文 [Mobile Link] [1073 ページ]
SQL Anywhere リモートデータベースから 1 人以上の同期ユーザを削除します。

DROP TABLE 文 [1074 ページ]
データベースからテーブルを削除します。

DROP TEXT CONFIGURATION 文 [1076 ページ]
テキスト設定オブジェクトを削除します。

DROP TEXT INDEX 文 [1077 ページ]
データベースからテキストインデックスを削除します。

DROP TIME ZONE 文 [1078 ページ]
データベースからタイムゾーンを削除します。

DROP TRACE EVENT 文 [1079 ページ]
ユーザ定義トレースイベントを削除します。

DROP TRACE EVENT SESSION 文 [1081 ページ]
トレースイベントセッションを削除します。

DROP TRIGGER 文 [1082 ページ]
データベースからトリガを削除します。

DROP USER 文 [1084 ページ]
ユーザを削除します。

DROP VARIABLE 文 [1085 ページ]
SQL 変数を削除します。

DROP VIEW 文 [1087 ページ]
データベースからビューを削除します。

EXCEPT 文 [1088 ページ]
2 つのクエリブロックの集合差を返します。

EXECUTE IMMEDIATE 文 [SP] [1090 ページ]
動的に作成された文をプロシージャの中から実行できるようにします。

EXECUTE 文 [ESQL] [1093 ページ]
準備された SQL 文を実行します。

EXECUTE 文 [T-SQL] [1096 ページ]
CALL 文の代わりとなる Adaptive Server Enterprise と互換性のあるプロシージャを呼び出して、準備された SQL 文を Transact-SQL で実行します。

EXIT 文 [Interactive SQL] [1098 ページ]
Interactive SQL を終了します。

EXPLAIN 文 [ESQL] [1099 ページ]
特定のカーソルに使用されている最適化方法のテキスト仕様を取得します。

FETCH 文 [ESQL] [SP] [1101 ページ]
カーソルを特定のローに配置 (再配置) し、そのローの式の値を、ストアドプロシージャまたはアプリケーション内からアクセスできる変数にコピーします。

FOR 文 [1106 ページ]
文リストの実行をカーソル内のローごとに 1 回繰り返します。

FORWARD TO 文 [1110 ページ]
ネイティブ構文の SQL 文をリモートサーバに送信します。

FROM 句 [1112 ページ]
DELETE、SELECT、または UPDATE 文に必要なデータベーステーブルまたはビューを指定します。SELECT 文内で使用される場合、FROM 句は MERGE 文または INSERT 文でも使用できます。

GET DATA 文 [ESQL] [1124 ページ]
カーソルの現在のローの 1 つのカラムに対する文字列またはバイナリデータを取得します。

GET DESCRIPTOR 文 [ESQL] [1126 ページ]
記述子領域内の変数に関する情報を取り出すか、その値を取り出します。

GET OPTION 文 [ESQL] [1128 ページ]
オプションの現在の設定を取得します。この文の代わりに、CONNECTION_PROPERTY 関数を使用することをお奨めします。

GOTO 文 [1129 ページ]
ラベルの付いた文に制御を分岐します。

GRANT 文 [1131 ページ]
システム権限とオブジェクトレベル権限をユーザとロールに付与します。

GRANT ROLE 文 [1136 ページ]
ユーザおよびロールにロールを付与します。

GRANT CONNECT 文 [1139 ページ]
新しいユーザを作成し、ユーザが自分のパスワードを変更する場合にも使用できます。ただし、ユーザを作成する場合は、GRANT CONNECT 文ではなく CREATE USER 文を使用することをおすすめします。

GRANT CONSOLIDATE 文 [SQL Remote] [1141 ページ]
SQL Remote 階層において、現在のデータベースのすぐ上にあり、現在のデータベースからメッセージを受信するデータベースを識別します。

GRANT CREATE 文 [1143 ページ]
指定された DB 領域にデータベースオブジェクトを作成できるユーザ権限を付与します。

GRANT EXECUTE 文 [1144 ページ]
プロシージャまたはユーザ定義ファンクションを実行するユーザ権限を付与します。

GRANT INTEGRATED LOGIN 文 [1145 ページ]
プロシージャまたはユーザ定義ファンクションを実行するユーザ権限を付与します。

GRANT KERBEROS LOGIN 文 [1146 ページ]
1 つ以上の Kerberos プリンシパルから既存のデータベースユーザ ID にマッピングする、Kerberos の認証済みログインを作成します。

GRANT PUBLISH 文 [SQL Remote] [1147 ページ]
ユーザ ID にパブリッシャ権限を付与します。パブリッシャ権限を付与するには、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

GRANT REMOTE 文 [SQL Remote] [1149 ページ]
SQL Remote 階層において、現在のデータベースのすぐ下にあり、現在のデータベースからメッセージを受信するデータベースを識別します。これらはリモートユーザといいます。

GRANT USAGE ON SEQUENCE 文 [1151 ページ]
指定されたシーケンスを使用する権限を付与します。

GROUP BY 句 [1152 ページ]
カラム、エイリアス名、関数を SELECT 文の一部としてグループ化します。

HELP 文 [Interactive SQL] [1155 ページ]
Interactive SQL 環境のヘルプを提供します。

IF 文 [1156 ページ]
SQL 文の条件付き実行を制御します。

IF 文 [T-SQL] [1158 ページ]
Watcom SQL IF 文の代わりに、SQL 文の条件付き実行を制御します。

INCLUDE 文 [ESQL] [1159 ページ]
SQL プリプロセッサによってスキャンされるソースプログラムにファイルをインクルードします。

INPUT 文 [Interactive SQL] [1160 ページ]
外部ファイル、キーボード、ODBC データソース、またはシェイプファイルからデータベーステーブルにデータをインポートします。

INSERT 文 [1167 ページ]
データベースの任意の場所から、1 つのローあるいは選択されたローをテーブルへ挿入します。

INSTALL EXTERNAL OBJECT 文 [1173 ページ]
外部環境で実行できるオブジェクトをインストールします。

INSTALL JAVA 文 [1175 ページ]
Java クラスをデータベース内で使用できるようにします。

INTERSECT 文 [1177 ページ]
2 つ以上のクエリの結果セット間の共通部分を計算します。

LEAVE 文 [1179 ページ]
複合文またはループから出ます。

LOAD STATISTICS 文 [1181 ページ]
この文は内部でのみ使用され、dbunload ユーティリティが使用して古いデータベースからカラム統計を ISYSCOLSTAT システムテーブルにアンロードします。

LOAD TABLE 文 [1182 ページ]
外部ファイルからデータベーステーブルにバルクデータをインポートします。

LOCK FEATURE 文 [1197 ページ]
他の同時接続でデータベースサーバの機能が使用されないようにします。

LOCK MUTEX 文 [1199 ページ]
事前定義されたミューテックスを使用して、ファイルやシステムプロシージャなどのリソースをロックします。

LOCK TABLE 文 [1201 ページ]
同時に実行されている他のトランザクションがテーブルにアクセスしたり、テーブルを修正したりすることを防止します。

LOOP 文 [1203 ページ]
文リストの実行を繰り返します。

MERGE 文 [1204 ページ]
テーブル、ビュー、プロシージャの結果をテーブルまたはビューにマージします。

MESSAGE 文 [1211 ページ]

メッセージを表示します。

NOTIFY SEMAPHORE 文 [1215 ページ]
セフォマに関連付けられたカウンタを増分します。

NOTIFY TRACE EVENT 文 [1217 ページ]
ユーザ定義トレースイベントをトレースセッションに記録します。

OPEN 文 [ESQL] [SP] [1218 ページ]
事前に宣言したカーソルを開き、データベースの情報にアクセスします。

OUTPUT 文 [Interactive SQL] [1222 ページ]
現在のクエリ結果をファイルに出力します。

PARAMETERS 文 [Interactive SQL] [1228 ページ]
Interactive SQL スクリプトファイルにパラメータを指定します。

PASSTHROUGH 文 [SQL Remote] [1230 ページ]
SQL Remote 管理のパススルーモードを起動または停止します。

PIVOT 句 [1232 ページ]
SELECT 文の FROM 句 (FROM `pivoted-derived-table`) のテーブル式を、行列変換された派生テーブルに変換します。行列変換された派生テーブルでは、テーブル式のカラムから取得したローの値を複数のカラムに回転し、必要に応じて結果セットに含まれるカラムを集計するための容易な方法を提供します。

PREPARE 文 [ESQL] [1238 ページ]
後で実行する文を準備するか、カーソルを定義します。

PREPARE TO COMMIT 文 [1241 ページ]
COMMIT を正常に実行できるかどうかをチェックします。

PRINT 文 [T-SQL] [1243 ページ]
クライアントにメッセージを返すか、データベースサーバメッセージウィンドウにメッセージを表示します。

PUT 文 [ESQL] [1244 ページ]
指定したカーソルにローを挿入します。

RAISERROR 文 [1246 ページ]
エラー信号を送り、クライアントにメッセージを送信します。

READ 文 [Interactive SQL] [1248 ページ]
ファイルから Interactive SQL 文を読み込む。

READTEXT 文 [T-SQL] [1250 ページ]
指定したオフセットから開始して、指定したバイト数のテキスト値とイメージ値をデータベースから読み込みます。この機能は Transact-SQL との互換性のためにのみ提供されているものであり、使用しないことをお奨めします。

REFRESH MATERIALIZED VIEW 文 [1252 ページ]
クエリ定義を実行することで、マテリアライズドビューのデータを初期化または更新します。

REFRESH TEXT INDEX 文 [1255 ページ]
テキストインデックスをリフレッシュします。

REFRESH TRACING LEVEL 文 (廃止予定) [1258 ページ]
診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。REFRESH TRACING LEVEL 文は、トレーシングセッションが進行中に、sa_diagnostic_tracing_level テーブルからトレーシングレベルをリロードします。

RELEASE MUTEX 文 [1259 ページ]

指定した接続スコープミューテックスが現在の接続によってロックされている場合に、その接続スコープミューテックスを解除します。

RELEASE SAVEPOINT 文 [1261 ページ]
現在のトランザクション内のセーブポイントを解放します。

REMOTE RESET 文 [SQL Remote] [1262 ページ]
カスタムデータベース抽出プロシージャで、単一トランザクションでの 1 リモートユーザのすべてのサブスクリプションを起動します。

REMOVE EXTERNAL OBJECT 文 [1263 ページ]
データベースから外部オブジェクトを削除します。

REMOVE JAVA 文 [1264 ページ]
データベースからクラスまたは JAR ファイルを削除します。

REORGANIZE TABLE 文 [1266 ページ]
データベースへの連続アクセスという要件があるために、データベース全体の再構築ができない場合に、テーブルの断片化を解除します。

RESIGNAL 文 [SP] [1268 ページ]
例外条件を送り返します。

RESTORE DATABASE 文 [1269 ページ]
バックアップされたデータベースをアーカイブからリストアします。

RESUME 文 [1271 ページ]
結果セットを返すカーソルの実行を再開します。

RETURN 文 [1272 ページ]
関数、プロシージャ、またはバッチを無条件で終了し、オプションで値を返します。

REVOKE CONSOLIDATE 文 [SQL Remote] [1274 ページ]
このデータベースからの SQL Remote メッセージを統合データベースが受信するのを中止させます。

REVOKE PUBLISH 文 [SQL Remote] [1276 ページ]
名前を指定したユーザ ID を現在のパブリッシャとして識別するのを中止します。パブリッシャ権限を取り消すには、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

REVOKE REMOTE 文 [SQL Remote] [1277 ページ]
このデータベースからの SQL Remote メッセージをユーザが受信できないようにします。

REVOKE 文 [1278 ページ]
ユーザとロールのシステム権限とオブジェクトレベル権限を取り消します。

REVOKE ROLE 文 [1281 ページ]
ユーザやロールからロールや権限を取り消します。

ROLLBACK 文 [1284 ページ]
トランザクションを終了し、最後に行った COMMIT または ROLLBACK 以降の変更を元に戻します。

ROLLBACK TO SAVEPOINT 文 [1285 ページ]
SAVEPOINT の後に加えられた変更を取り消します。

ROLLBACK TRANSACTION 文 [T-SQL] [1286 ページ]
SAVE TRANSACTION の後に加えられた変更を取り消します。

ROLLBACK TRIGGER 文 [1287 ページ]
トリガによって加えられた変更を取り消します。

SAVE TRANSACTION 文 [T-SQL] [1288 ページ]
現在のトランザクション内でセーブポイントを確立します。

SAVEPOINT 文 [1290 ページ]
現在のトランザクション内でセーブポイントを確立します。

SELECT 文 [1291 ページ]
データベースから情報を取得します。

SET CONNECTION 文 [Interactive SQL] [ESQL] [1302 ページ]
アクティブなデータベース接続を変更します。

SET DESCRIPTOR 文 [ESQL] [1303 ページ]
SQL 記述子領域内の変数を記述したり、データを記述子領域に格納したりします。

SET MIRROR OPTION 文 [1305 ページ]
データベースミラーリングと読み込み専用のスケールアウトの設定を制御するオプションの値を変更します。

SET OPTION 文 [1310 ページ]
データベースオプションと接続オプションの値を変更します。

SET OPTION 文 [Interactive SQL] [1314 ページ]
Interactive SQL オプションの値を変更します。

SET REMOTE OPTION 文 [SQL Remote] [1316 ページ]
SQL Remote メッセージリンクのメッセージ制御パラメータを設定します。

SET SQLCA 文 [ESQL] [1323 ページ]
デフォルトのグローバル `sqlca` 以外の SQLCA を使用するように、SQL プリプロセッサに通知します。

SET 文 [1325 ページ]
SQL 変数に値を代入します。

SET 文 [T-SQL] [1327 ページ]
Adaptive Server Enterprise との互換性を保つ方法で現在の接続にデータベースオプションを設定します。

SETUSER 文 [1329 ページ]
認証された別のユーザ ID の使用をユーザに許可します (同一化)。

SIGNAL 文 [SP] [1332 ページ]
例外条件を通知します。

START DATABASE 文 [1333 ページ]
データベースを現在のデータベースサーバで起動します。

START SERVER 文 [Interactive SQL] [1336 ページ]
データベースサーバを起動します。

START EXTERNAL ENVIRONMENT 文 [1338 ページ]
外部環境を開始します。

START JAVA 文 [1339 ページ]
Java VM を起動します。

START LOGGING 文 [Interactive SQL] [1340 ページ]
実行された SQL 文およびメッセージのログファイルへのロギングを開始します。

START SUBSCRIPTION 文 [SQL Remote] [1341 ページ]
パブリケーションに対するユーザのサブスクリプションを開始します。

START SYNCHRONIZATION DELETE 文 [Mobile Link] [1343 ページ]
Mobile Link 同期で行われた削除のロギングを再起動します。

START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link] [1345 ページ]
Mobile Link 同期スキーマの変更を開始します。

STOP DATABASE 文 [1347 ページ]
現在のデータベースサーバ上のデータベースを停止します。

STOP EXTERNAL ENVIRONMENT 文 [1348 ページ]
外部環境を停止します。

STOP JAVA 文 [1350 ページ]
Java VM を停止します。

STOP LOGGING 文 [Interactive SQL] [1351 ページ]
現在のセッションで実行される SQL 文およびメッセージのロギングを停止します。

STOP SERVER 文 [1352 ページ]
データベースサーバを停止します。

STOP SUBSCRIPTION 文 [SQL Remote] [1353 ページ]
パブリケーションに対するユーザのサブスクリプションを停止します。

STOP SYNCHRONIZATION DELETE 文 [Mobile Link] [1355 ページ]
Mobile Link 同期で行われた削除のロギングを一時的に停止します。

STOP SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link] [1357 ページ]
Mobile Link の同期スキーマ変更を停止します。

SYNCHRONIZE 文 [Mobile Link] [1358 ページ]
SQL Anywhere データベースを Mobile Link サーバと同期させます。文自体に同期オプションを指定できます。

SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote] [1362 ページ]
パブリケーションに対するユーザのサブスクリプションを同期します。

SYSTEM 文 [Interactive SQL] [1364 ページ]
Interactive SQL から実行ファイルを起動します。

TRIGGER EVENT 文 [1365 ページ]
指定したイベントをトリガします。イベントは、イベントトリガに対して定義したものでも、スケジュールされたイベントでもかまいません。

TRUNCATE 文 [1366 ページ]
テーブル定義を削除しないで、テーブルからすべてのローを削除します。

TRUNCATE TEXT INDEX 文 [1369 ページ]
MANUAL または AUTO REFRESH テキストインデックスのデータを削除します。

TRY 文 [1370 ページ]
複合文のエラー処理を実装します (TRY ブロックでエラーが発生した場合、CATCH ブロックに囲まれている別の文のグループに制御を渡します)。

UNION 文 [1373 ページ]
2 つ以上の SELECT 文またはクエリ式の結果を結合します。

UNLOAD 文 [1376 ページ]
データソースからファイルにデータをアンロードします。

UNPIVOT 句 [1382 ページ]
FROM 句 (FROM `unpivoted-derived-table expression`) のテーブル式の準拠型カラムを、派生テーブルのローに列行変換します。UNPIVOT はデータの正規化に使用されます。たとえば、テーブルの複数のカラムに格納された類似のデータを持つ場合、それらを 1 つのカラムに返したい場合があります。

UPDATE (位置付け) 文 [ESQL] [SP] [1386 ページ]
カーソルの現在位置のデータを変更します。

UPDATE 文 [SQL Remote] [1388 ページ]
データベース内のデータを変更します。

UPDATE 文 [1391 ページ]
データベーステーブルにあるローを変更します。

VALIDATE LDAP SERVER 文 [1398 ページ]
LDAP サーバ設定オブジェクトを検証します。

VALIDATE 文 [1400 ページ]
現在のデータベース、または現在のデータベース内にある単一または複数のテーブル、マテリアライズドビュー、またはインデックスを検証します。

WAITFOR 文 [1403 ページ]
指定された時間の間、または指定の時間になるまで現在の接続処理を遅らせます。

WAITFOR SEMAPHORE 文 [1405 ページ]
セフォマに関連付けられたカウンタを減分します。

WHENEVER 文 [ESQL] [1407 ページ]
Embedded SQL プログラム内でのエラー処理を指定します。

WHILE 文 [T-SQL] [1408 ページ]
文または複合文を繰り返し実行します。

WINDOW 句 [1409 ページ]
SELECT 文の AVG や RANK などの Window 関数を使用するウィンドウのすべてまたは一部を定義します。

WRITETEXT 文 [T-SQL] [1412 ページ]
CHAR、NCHAR、または BINARY カラムのログなしの更新を許可します。この機能は Transact-SQL との互換性のためにのみ提供されているものであり、使用しないことをお奨めします。

1.4.4.1 ALLOCATE DESCRIPTOR 文 [ESQL]

SQL 記述子領域 (SQLDA) に使用する領域を割り付けます。

構文

```
ALLOCATE DESCRIPTOR descriptor-name
[ WITH MAX { integer | hostvar } ]
```

```
descriptor-name : identifier
```

パラメータ

WITH MAX 句

記述子領域の変数の数を指定できます。デフォルトのサイズは 1 です。さらに、fill_sqlda を呼び出して実際のデータ項目に使用する領域を割り付けてから、フェッチを行ったり、記述子領域内のデータにアクセスする文を実行します。

備考

記述子領域 (SQLDA) に使用する領域を割り付けます。C コードの中で次の宣言を行ってから、この文を使用します。

```
struct sqlda * descriptor_name
```

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

ALLOCATE DESCRIPTOR は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部です。

例

次のサンプルプログラムは、ALLOCATE DESCRIPTOR 文の使用例です。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
EXEC SQL INCLUDE SQLCA;
#include "sqldef.h"
EXEC SQL BEGIN DECLARE SECTION;
int         x;
short       type;
int         numcols;
char        string[100];
a_SQL_statement_number  stmt = 0;
EXEC SQL END DECLARE SECTION;
int main(int argc, char * argv[]){
```

```
    struct sqlda *      sqlda1;
    if( !db_init( &sqlca ) ) {
       return 1;
    }
    db_string_connect( &sqlca,
    "UID=DBA;PWD=passwd;DBF=d:¥¥DB Files¥¥sample.db");
    EXEC SQL ALLOCATE DESCRIPTOR sqlda1 WITH MAX 25;
    EXEC SQL PREPARE :stmt FROM
       'SELECT * FROM Employees';
    EXEC SQL DECLARE curs CURSOR FOR :stmt;
    EXEC SQL OPEN curs;
    EXEC SQL DESCRIBE :stmt into sqlda1;
    EXEC SQL GET DESCRIPTOR sqlda1 :numcols=COUNT;
    // how many columns?
    if( numcols > 25 ) {
       // reallocate if necessary
       EXEC SQL DEALLOCATE DESCRIPTOR sqlda1;
       EXEC SQL ALLOCATE DESCRIPTOR sqlda1
          WITH MAX :numcols;
       EXEC SQL DESCRIBE :stmt into sqlda1;
    }
    type = DT_STRING; // change the type to string
    EXEC SQL SET DESCRIPTOR sqlda1 VALUE 2 TYPE = :type;
    fill_sqlda( sqlda1 );
    // allocate space for the variables
    EXEC SQL FETCH ABSOLUTE 1 curs
       USING DESCRIPTOR sqlda1;
    EXEC SQL GET DESCRIPTOR sqlda1
       VALUE 2 :string = DATA;
    printf("name = %s", string );
    EXEC SQL DEALLOCATE DESCRIPTOR sqlda1;
    EXEC SQL CLOSE curs;
    EXEC SQL DROP STATEMENT :stmt;
    db_string_disconnect( &sqlca, "" );
    db_fini( &sqlca );
    return 0;
}
```

関連情報

DEALLOCATE DESCRIPTOR 文 [ESQL] [999 ページ]

1.4.4.2 ALTER DATABASE 文

データベースのアップグレード、データベースの jConnect サポートのオン/オフの切り替え、データベースの調整、トランザクションログファイル名とトランザクションログミラーファイル名の変更、またはミラーサーバ対するデータベースの所有権の取得の強制を実行します。

構文

システムのアップグレードまたはオブジェクトのリストアの実行

```
ALTER DATABASE UPGRADE
[ PROCEDURE ON ]
[ JCONNECT { ON | OFF } ]
```

```
[ RESTART { ON | OFF } ]
[ SYSTEM PROCEDURE AS DEFINER { ON | OFF } ]
```

ユーザ定義のアップグレードの実行

```
ALTER DATABASE UPGRADE
SCRIPT FILE sql_script_path
[ RESTART ON | OFF ]
```

調整の実行

```
ALTER DATABASE {
 CALIBRATE [ SERVER ]
 | CALIBRATE DBSPACE dbspace-name
 | CALIBRATE DBSPACE TEMPORARY
 | CALIBRATE GROUP READ
 | CALIBRATE PARALLEL READ
 | RESTORE DEFAULT CALIBRATION
}
```

トランザクションログ名とトランザクションログミラー名の変更

```
ALTER DATABASE dbfile
ALTER [ TRANSACTION ] LOG
{ ON [ log-name ] [ MIRROR mirror-name ] | OFF }
[ KEY key ]
```

データベースの所有権の変更

```
ALTER DATABASE
{ dbname FORCE START
| SET PARTNER FAILOVER }
```

チェックサムの停止

```
ALTER DATABASE SET CHECKSUM OFF
```

安定した状態へのキャッシュウォーミング

```
ALTER DATABASE
{ SAVE CACHE
 | RESTORE CACHE
 | DROP CACHE }
```

パラメータ

dbfile

データベースファイル。変数名も指定できます。

PROCEDURE clause

データベースに含まれるすべての dbo 所有プロシージャと SYS 所有プロシージャを削除して再作成します。

JCONNECT clause

jConnect JDBC ドライバからシステムカタログ情報にアクセスできるようにするには、JCONNECT ON を指定します。この句によって jConnect をサポートするシステムオブジェクトがインストールされます。jConnect システムオブジェクトを

除外するには、JCONNECT OFF を指定します。その場合でも、システム情報にアクセスしないかぎり、JDBC を使用できます。JCONNECT はデフォルトで ON です。

RESTART clause

RESTART はデフォルトで ON です。RESTART ON が指定されており、AutoStop (ASTOP) 接続パラメータが No に設定されている場合、アップグレード後にデータベースが再起動します。それ以外の場合、アップグレード後にデータベースが停止します。

SYSTEM PROCEDURE AS DEFINER { ON | OFF } clause

SYSTEM PROCEDURE AS DEFINER 句は、権限付き作業を実行する 16.0 より前のシステムプロシージャを、invoker または definer (所有者) の権限で実行するかどうかを指定します。ON の場合、これらのシステムプロシージャは definer (所有者) で実行されます。OFF の場合、これらのシステムプロシージャは invoker の権限で実行されます。

この句が指定されていない場合、アップグレードされたデータベースの現在の動作を維持するのがデフォルトとなります。16.0 より前のバージョンのデータベースをアップグレードしている場合、プロシージャは定義として実行されます。

この設定は、ユーザ定義プロシージャまたはバージョン 16.0 以降に導入されたプロシージャには影響しません。

SCRIPT FILE clause

この句を使用して、ユーザ定義のアップグレードスクリプトファイルの場所を指定します。`sql_script_path` は、実行する DML 文と DDL 文が含まれる `.sql` スクリプトファイルの場所と名前です。

スクリプトの実行に失敗し、RESTART ON (デフォルト) が指定されている場合、データベースはアップグレードの前に自動的に実行されたチェックポイントまでロールバックされ、再起動されます。RESTART OFF が指定されている場合、データベースはチェックポイントを行わずに停止し、次のデータベースの開始時に前回のチェックポイントまでロールバックされます。

CALIBRATE [SERVER] clause

テンポラリ DB 領域以外のすべての DB 領域を調整します。この句は、CALIBRATE PARALLEL READ によって実行される処理も実行します。

CALIBRATE DBSPACE clause

指定した DB 領域を調整します。

CALIBRATE DBSPACE TEMPORARY clause

テンポラリ DB 領域を調整します。

CALIBRATE GROUP READ clause

テンポラリ DB 領域でグループ読み込みの調整を実行します。テンポラリ DB 領域に大きなワークテーブルを書き込み、さまざまなグループ読み込みサイズを使用してファイルの読み込み時間を設定します。テンポラリテーブルに追加する領域が接続の制限を超えた場合、またはキャッシュサイズが足りず、最大メモリサイズで調整できない場合、調整が失敗してエラーメッセージが返されます。

CALIBRATE PARALLEL READ clause

すべての DB 領域ファイルについてデバイスの並列 I/O 機能を調整します。CALIBRATE [SERVER] 句もこの調整を実行します。

RESTORE DEFAULT CALIBRATION clause

一般的なハードウェアと構成設定に基づく組み込みのデフォルト値に、ディスク転送時間 (DTT: Disk Transfer Time) をリストアします。

ALTER [TRANSACTION] LOG clause

トランザクションログファイルまたはトランザクションログミラーファイルの名前を変更します。MIRROR `mirror-name` を指定しないと、新しいトランザクションログのファイル名が設定されます。データベースがトランザクションログを現在使っ

ていない場合、データベースは設定されたファイル名を使って起動します。すでにトランザクションログを使っている場合、データベースはトランザクションログとして新しいファイル名を使って起動します。

MIRROR `mirror-name` を指定する、新しいトランザクションログミラーのファイル名として設定されます。データベースがトランザクションログミラーを現在使っていない場合、データベースはこの設定された名前を使って起動します。すでにトランザクションログミラーを使っている場合、データベースはトランザクションログミラーとして新しいファイル名を使って起動します。

この句を使用して、トランザクションログまたはトランザクションログミラーをオフにすることもできます。たとえば、ALTER DATABASE ALTER LOG OFF のように指定します。

KEY clause

トランザクションログまたはトランザクションログミラーに使用する暗号化キーを指定します。この暗号化キーには文字列または変数名を使用できます。強力に暗号化されたデータベースで ALTER [TRANSACTION] LOG 句を使用するときは、暗号化キーを指定する必要があります。

dbname FORCE START clause

現在ミラーサーバとして動作しているデータベースサーバに、データベースの所有権の取得を強制します。

> ⚠ 警告
>
> FORCE START 句を使用する場合、ミラーサーバに存在しないトランザクションがプライマリサーバにあると、結果としてトランザクションが失われることがあります。
>
> プライマリを再起動し、SET PARTNER FAILOVER 句を指定して ALTER DATABASE を実行することで、トランザクションを失わないで失敗を強制することをおすすめします。FORCE START 句は、プライマリを再起動できない場合にのみ、最後の手段として使用してください。

この句は、プロシージャまたはイベントから実行できます。また、ミラーサーバ上のユーティリティデータベースに接続しているときに実行する必要があります。

SET PARTNER FAILOVER clause

サーバを停止せずに、プライマリサーバからミラーサーバへのデータベースミラーリングのフェイルオーバーを起動します。この文は、プライマリサーバ上のデータベースに接続している間に実行する必要があり、プロシージャ内またはイベント内から実行できます。この文が実行された場合

1. データベースサーバが、その文を実行している接続を含む、データベースに対するすべての接続を終了します。
2. データベースが停止し、ミラーロールで再起動します。
3. 文にプロシージャまたはイベントが含まれている場合、後続の他の文が実行されない場合があります。

SET CHECKSUM OFF clause

データベースのグローバルチェックサムを無効にします。デフォルトでは、新しいデータベースでグローバルチェックサムが有効になりますが、バージョン 11 以前のデータベースではグローバルチェックサムは有効になりません。

リムーバブルドライブなどのストレージデバイス上で実行されているデータベースでは、データベースファイルの破損を速やかに検出できるように、この句の設定に関係なく、データベースサーバによって常に書き込みチェックサムが有効になります。また、検証アクティビティの実行時に重要なページのチェックサムも計算されます。

グローバルチェックサムが有効になっていないデータベースでは、-wc オプションを使用して書き込みチェックサムを有効にすることができます。

SAVE CACHE clause

パフォーマンスを向上させため、必要に応じてデータベースをこの状態にリストアできるよう、現在のキャッシュコンテンツを記録します。

この文は、トランザクションログに記録されておらず、高可用性および読み込み専用スケールアウト設定に関連するデータベースを含む読み込み専用データベースで実行できません。

文の実行後、SYSDBSPACE システムビューの prefetch_pages カラムが現在キャッシュにあるページごとに 1 ビット更新されます。使用されるビットマップは、sp_db_cache_contents システムプロシージャから返されるものと同じです。

RESTORE CACHE clause

現在のデータベースを ALTER DATABASE SAVE CACHE 文が実行されたときの状態にリストアすることで、パフォーマンスが向上します。

この文は、トランザクションログには記録されません。

この文は、SYSDBSPACE システムビューの prefetch_pages カラムによって識別されるページを読み込みます。文の結果は、SYSDBSPACE システムビューの prefetch_column に格納されているビットマップとともに各 DB 領域の sp_read_db_pages システムプロシージャを呼び出した結果と同じです。

DROP CACHE clause

保存されているキャッシュページをクリアします。

この句は、SYSDBSPACE システムビューの saved_cache_pages カラムに格納されているページをクリアし、カラム値を NULL に設定します。ALTER DATABASE DROP CACHE 文の実行後、ALTER DATABASE SAVE CACHE 文を最初に実行していない場合は後続の ALTER DATABASE RESTORE CACHE 文からエラーが返されます。

備考

構文 - コンポーネントのアップグレードまたはオブジェクトのリストア

ALTER DATABASE UPGRADE 文は、データベースをアップグレードまたは更新するためのアップグレードユーティリティ (dbupgrad) の代わりとして使用します。デフォルトで、データベースが停止して、アップグレード後に再起動されます。アップグレード中にトランザクションログがアーカイブされ、データベースが停止または再起動する前に、新しいトランザクションログが作成されます。

ALTER DATABASE UPGRADE 文の実行後、データベースをシャットダウンし、トランザクションログをアーカイブする必要があります。

一般に、マイナーバージョンでのデータベースの変更はデータベースオプションの追加やシステムテーブルとシステムプロシージャの細かい変更にかぎられます。ALTER DATABASE UPGRADE 文は、システムテーブルを最新バージョンにアップグレードし、新規データベースオプションを追加します。必要に応じて、すべてのシステムプロシージャを削除し、再作成します。PROCEDURE ON 句を指定すると、システムプロシージャを強制的に再構築できます。

ALTER DATABASE UPGRADE 文を現在ミラーされているデータベースで実行すると、エラーメッセージが返されます。

ALTER DATABASE UPGRADE 文を使用して、設定やシステムオブジェクトを元のインストール状態にリストアすることもできます。

データベースファイルの物理的な再編成を必要とする機能は、ALTER DATABASE UPGRADE 文を実行して使用可能にすることはできません。そのような機能には、インデックスの拡張やデータの格納に関する変更が含まれています。これらの拡張機能を利用するには、データベースのアンロードと再ロードを行ってください。

データベースのアップグレードを行う前に、データベースをバックアップしてください。

jConnect JDBC ドライバを使用してシステムカタログ情報にアクセスするには、JCONNECT ON (デフォルト値) を指定します。jConnect システムオブジェクトを実行する場合は、JCONNECT OFF を指定します。JCONNECT OFF を設定し

ても、データベースから jConnect サポートが削除されることはありません。その場合でも、システムカタログ情報にアクセスしないかぎり、JDBC を使用できます。続けて jConnect の新しいバージョンをダウンロードする場合、ALTER DATABASE UPGRADE JCONNECT ON 文を (再) 実行して、データベースのバージョンをアップグレードできます。

構文 - ユーザ定義のアップグレードの実行

sql_script_path が有効でない場合は、エラーが返され、データベースは停止も再起動もしません。

スクリプトはデータベースサーバコンピュータに格納し、そこから実行する必要があります。

スクリプトの実行に成功すると、データベースの実行が継続されます。

構文 - 調整の実行

オプティマイザが使用する I/O コストモデルを再調整することもできます。この操作により、ディスク転送時間 (DTT: Disk Transfer Time) モデルが更新されます。DTT は、コストモデルで使用されるディスク I/O の数学的モデルです。I/O コストモデルを再調整するときは、データベースサーバを他の用途に使用できません。また、コンピュータ上の他のすべてのアクティビティがアイドル状態になっている必要があります。データベースサーバの再調整は高負荷のオペレーションであり、完了までに長時間かかることがあります。デフォルトのままにします。

CALIBRATE PARALLEL READ 句を使用する場合、10000 ページ未満の DB 領域ファイルでは並列の調整は実行されません。調整操作中にデータベースサーバが自動的にすべてのアクティビティを中断した場合でも、同じコンピュータに大量のリソースを消費するプロセスがなければ、並列の調整は実行されます。調整後、IOParallelism 拡張データベースプロパティを使用して、DB 領域ファイルに使用できる並列 I/O 操作の最大推定数を取得できます。

同じようなハードウェアインストールが大量にある場合に、繰り返し行われる時間のかかる再調整作業を省くには、sa_unload_cost_model システムプロシージャを使用してアンロードしてから、sa_load_cost_model システムプロシージャを使用して別のデータベースに適用 (ロード) して調整を再使用できます。

構文 - トランザクションログ名とトランザクションログミラー名の変更

ALTER DATABASE 文を使用して、データベースファイルに関連付けられているトランザクションログとトランザクションログミラーの名前を変更します。データベースの実行中にこれらの変更を行わないでください。これらの変更は、トランザクションログ (dblog) ユーティリティで行う変更と同じです。この文は、-gu オプションの設定に応じて、ユーティリティデータベースまたは別のデータベースと接続しているときに実行できます。

暗号化されたデータベースのトランザクションログまたはトランザクションログミラーを変更する場合は、キーを指定してください。データベースが監査を実行中にトランザクションログの使用を停止することはできません。監査をオフにすると、トランザクションログの使用を停止できます。この構文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

BACKUP DATABASE 文を使用して、実行中のデータベースのトランザクションログの名前を変更します。例:

```
BACKUP DATABASE DIRECTORY 'directory-name'
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME;
```

構文 - データベースの所有権の変更

ALTER DATABASE...FORCE START は、プライマリサーバではなくミラーサーバから実行する必要があります。

構文 - チェックサムの停止

この句は、データベースのチェックサムを無効にする場合にのみ使用できます。

構文 - 安定した状態へのキャッシュウォーミング

安定した状態のキャッシュコンテンツを記録し、必要に応じてこの情報をリストアするには、ALTER DATABASE 文を使用します。

注記

変数名を受け付けるパラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

次のリストで指定しないかぎり、データベースの更新には ALTER DATABASE システム権限のみが必要です。

- コンポーネントのアップグレードまたはオブジェクトのリストア: データベースをアップグレードするには、ALTER DATABASE システム権限が必要です。また、このデータベースに他の接続がないことが必要です。
- ユーザ定義のアップグレードの実行:ALTER DATABASE 文の実行には ALTER DATABASE システム権限のみが必要ですが、アップグレードを実行しているユーザは、.sql ファイルで指定したアクションを実行するために必要なすべての権限が必要です。適切な権限がない場合、データベースアップグレードは失敗し、データベースは ALTER DATABASE 文の実行前の状態にロールバックされます。
- 調整の実行: SERVER OPERATOR システム権限と、トランザクションログのあるディレクトリのファイルパーミッションを持っており、データベースが実行中である必要があります。
  ALTER DATABASE dbfile ALTER TRANSACTION LOG 文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。
- データベースの所有権の変更: SERVER OPERATOR システム権限が必要です。
  ALTER DATABASE dbname FORCE START 文を実行するのに必要な権限は、-gd データベースサーバオプションを使用して変更できます。
- チェックサムの停止: ALTER DATABASE システム権限が必要です。
- 安定した状態へのキャッシュウォーミング: SERVER OPERATOR システム権限が必要です。

関連する動作

オートコミット

ALTER DATABASE UPGRADE 文の実行では、アップグレード中にトランザクションログがアーカイブされ、データベースがアップグレード後に再起動されるときに新しいトランザクションログが作成されます。また、デフォルトで、データベースが停止して、アップグレード後に再起動されます。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

ALTER DATABASE 文は、Adaptive Server Enterprise でサポートされています。ただし、Adaptive Server Enterprise でサポートされる文の句は、SQL Anywhere でサポートされる句とは切り離されています。

例

1. 次の文は、jConnect サポートを無効にします。

```
ALTER DATABASE UPGRADE JCONNECT OFF;
```

2. 次の文は、*demo.db* に関連するトランザクションログファイル名を `mynewdemo.log` に設定します。

```
ALTER DATABASE 'demo.db¥¥demo.db'
   ALTER LOG ON 'mynewdemo.log';
```

3. 次の文は、架空の `myUpgrade.sql` スクリプトファイルを実行することでユーザ定義のアップグレードを実行します。アップグレードに成功すると、データベースの実行が継続されます。アップグレードに失敗すると、データベースはアップグレードの前に自動的に実行されたチェックポイントまでロールバックされ、再起動されます。

```
ALTER DATABASE UPGRADE SCRIPT FILE 'C:¥¥Users¥¥Public¥¥Documents¥
¥myUpgrade.sql'
   RESTART ON;
```

4. 次の文は、データベース `asatest.db` の変数を作成します。

```
CREATE VARIABLE @db1 LONG VARCHAR ;
SET @db1 = 'asatest' ;
```

   1. 次の文は、データベース `asatest.db` の変更に変数 @db1 を使用します。

```
ALTER DATABASE @db1
ALTER TRANSACTION LOG ON 'vis_tmp2.log'
MIRROR 'vis_tmp2.mlg' ;
```

5. 次の例は、データベースが安定した状態で実行されているときにキャッシュのコンテンツを保存し、必要に応じてキャッシュをその状態にリストアする方法を示します。
   1. 次の文を実行し、DB 領域を作成します。

```
CREATE DBSPACE mydbs
AS 'C:¥¥mydb¥¥mydbs.db';
```

   2. 新しい DB 領域にテーブルを作成し、次の文を実行してデータを追加します。

```
CREATE TABLE mytable( col1 INT, col2 CHAR(128) ) IN mydbs;
```

```
INSERT INTO mytable
    SELECT column_id, column_name
    FROM sys.syscolumn;
```

```
COMMIT;
```

   3. 次の文を実行し、新しい DB 領域の ID を特定します。

```
SELECT * FROM SYS.SYSDBSPACE WHERE dbspace_name = 'mydbs';
```

      新しい DB 領域の ID を書き留め、saved_cache_pages カラムが NULL であることに注目します。

4. 次の文を実行することで、新しい DB 領域のどのページが現在キャッシュにあるかを確認します。

```
SELECT * FROM dbo.sp_db_cache_contents( );
```

上の文は、データベース内のすべての DB 領域の情報を返します。結果を新しい mydbs DB 領域に制限するには、次の文を実行します。ここで、mydbs-ID は SYS から取得された新しい mydbs DB 領域の ID です。SYSDBSPACE:

```
SELECT * FROM dbo.sp_db_cache_contents( mydbs-ID );
```

5. 次の文を実行することで、データベースの現在の安定した状態を保存します。

```
ALTER DATABASE SAVE CACHE;
```

6. その後、他の多くのトランザクションが実行され、キャッシュに大幅な変更が加えられた後、保存された安定した状態に戻すことができます。すべての DB 領域の安定した状態のページをすべてキャッシュに読み込むか、特定の DB 領域の安定した状態のページをすべてキャッシュに読み込むこともできます。
すべての DB 領域の安定した状態のページをすべてキャッシュに読み込むには、次の文を実行します。

```
ALTER DATABASE RESTORE CACHE;
```

ある DB 領域 (新しい DB 領域 mydbs など) の安定した状態のページをすべて読み込む場合は、次のような一連の文を実行します。

```
CREATE VARIABLE cache_pages LONG VARBIT;
```

```
SELECT saved_cache_pages
    INTO cache_pages
    FROM SYS.SYSDBSPACE
    WHERE dbspace_name='mydbs';
```

```
CALL dbo.sp_read_db_pages( mydbs-ID, cache_pages );
```

関連情報

CREATE DATABASE 文 [781 ページ]
CREATE STATISTICS 文 [942 ページ]
BACKUP DATABASE 文 [739 ページ]
DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]
DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]

1.4.4.3 ALTER DBSPACE 文

DB 領域またはトランザクションログ用に領域を事前に割り付けたり、DB 領域ファイルの名前変更時や移動時にカタログを更新したりします。

構文

```
ALTER DBSPACE { dbspace-name | TRANSLOG | TEMPORARY }
 { ADD number [ add-unit ]
   | RENAME filename }
```

```
add-unit :
PAGES
| KB
| MB
| GB
| TB
```

パラメータ

TRANSLOG 句

この特別な DB 領域名 TRANSLOG を指定して、トランザクションログにディスク領域を事前に割り付けます。事前に割り付けておくと、トランザクションログが急速に大きくなることが予測される場合に、パフォーマンスを改善できます。たとえば、ビットマップのような多量のバイナリラージオブジェクト (BLOB) を処理する場合、この機能を使用できます。

構文 ALTER DBSPACE `dbspace-name` TRANSLOG RENAME `filename` はサポートされていません。

TEMPORARY 句

特別な DB 領域名 TEMPORARY を指定して、テンポラリ DB 領域にスペースを追加します。テンポラリ DB 領域にスペースが追加されるとすぐに、追加のスペースは対応するテンポラリファイルで実体化されます。データベースのテンポラリ DB 領域に領域を事前に割り付けると、大きなワークテーブルを使用する複雑なクエリを実行する場合、パフォーマンスが向上します。

ADD 句

ALTER DBSPACE 文に ADD 句を指定して、DB 領域にディスク領域を事前に割り付けます。ページ、キロバイト (KB)、メガバイト (MB)、ギガバイト (GB)、またはテラバイト (TB) 単位でサイズを指定して、対応するデータベースファイルを拡張します。単位を指定しない場合、デフォルトは PAGES です。データベースのページサイズはデータベースの作成時に決定されます。

領域が事前に割り付けられていない場合、データベースファイルは、領域が必要になったとき、ページサイズが 2 KB、4 KB、8 KB の場合は一度に約 256 KB 拡張され、その他のページサイズの場合は約 32 ページ拡張されます。領域を事前に割り付けると、多量のデータをロードする場合のパフォーマンスを改善でき、ファイルシステム内でデータベースファイルの断片化を防ぐことができます。

この句を使用して、事前定義の DB 領域 (system、temporary、temp、translog、translogmirror) のいずれかに領域を追加できます。

RENAME 句

メインファイル以外のデータベースファイルを別のファイル名に変更したり、別のディレクトリまたはデバイスに移動したりする場合は、RENAME 句を指定した ALTER DBSPACE 文を使用すると、データベース起動時にデータベースサーバに確実に新しいファイルを検索させることができます。filename パラメータには、文字列リテラルまたは変数を指定できます。

名前の変更は、次のように有効になります。

- 文を実行する前に DB 領域がすでに開いている場合 (つまり、実際のファイルの名前はまだ変更していない場合)、継続してアクセスすることはできますが、カタログに格納されている名前は更新されます。データベースが停止した後、ファイルの名前を変更して、RENAME 句で指定したのと同じ名前に変更する必要があります。そうしないと、カタログにある DB 領域の名前とファイル名が一致せず、データベースを次に起動するときに、データベースサーバが DB 領域を開くことができなくなります。
- 文を実行したときに DB 領域が開いていない場合、データベースサーバは、カタログを更新し、その後で DB 領域を開くことを試行します。DB 領域を開くことができたら、アクセス可能になっています。DB 領域を開くことができない場合でも、エラーは返されません。
  DB 領域が開いているかどうかを確認するには、次に示す文を実行します。結果が NULL である場合は、DB 領域は開いていません。

  ```
  SELECT DB_EXTENDED_PROPERTY('FileSize','dbspace-name');
  ```

  メイン DB 領域の system に RENAME 句を指定した ALTER DBSPACE を使用しても効果はありません。RENAME 句はトランザクションログファイルの名前の変更には対応していません。BACKUP DATABASE 文を使用して、実行中のデータベースのトランザクションログの名前を変更できます。例:

  ```
  BACKUP DATABASE DIRECTORY 'directory-name'
  TRANSACTION LOG ONLY
  TRANSACTION LOG RENAME;
  ```

備考

それぞれのデータベースは 1 つまたは複数のファイルの中に保持されます。DB 領域は、各データベースファイルに関連付けられた論理名を持つ追加ファイルであり、メインデータベースファイル単独では保持できないデータを格納するために使用されます。ALTER DBSPACE は、メインデータベース (ルートファイルとも呼ばれます) または追加の DB 領域を修正します。データベースの DB 領域名は、SYSDBSPACE ビューに保持されます。メインデータベースファイルの DB 領域名は system です。

マルチファイルデータベースを起動すると、起動ラインまたは ODBC データソースの記述が、データベースサーバにメインデータベースファイルの場所を知らせます。メインデータベースファイルは、システムテーブルを保持しており、SQL Anywhere は、このシステムテーブルを調べて他の DB 領域のロケーションを検索します。次に、データベースサーバはこれらのシステムテーブルで他の DB 領域の場所を検索し、各 DB 領域を開きます。default_dbspace オプションを設定して新規テーブルを作成する DB 領域を指定できます。

権限

MANAGE ANY DBSPACE システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、system の DB 領域サイズを 200 ページ増やします。

```
ALTER DBSPACE system
ADD 200;
```

次の例は、system の DB 領域サイズを 400 MB 増やします。

```
ALTER DBSPACE system
ADD 400 MB;
```

次の例は、架空の system_2 の DB 領域に関連するファイル名を変更します。

```
ALTER DBSPACE system_2
RENAME 'e:¥db¥dbspace2.db';
```

関連情報

CREATE DBSPACE 文 [789 ページ]
BACKUP DATABASE 文 [739 ページ]
SYSDBSPACE システムビュー [1798 ページ]

1.4.4.4 ALTER DOMAIN 文

ユーザ定義のドメインまたはデータ型の名前を変更します。

構文

```
ALTER { DOMAIN | DATATYPE } user-type
RENAME new-name
```

備考

この文を実行すると、ユーザ定義のドメインまたはデータ型の名前が ISYSUSERTYPE システムテーブル内で更新されます。

注記

テーブルスキーマ内のドメイン名参照は自動的にアップデートされますが、古いユーザ定義ドメインまたはデータ型を参照するプロシージャ、トリガ、ビュー、またはイベントはすべて、新しい名前を参照するよう手動でアップデートする必要があります。

権限

そのドメインの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- そのドメインに対する ALTER 権限
- ALTER DATATYPE システム権限
- ALTER ANY OBJECT システム権限

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。ALTER DOMAIN 文は、オプションの ANSI/ISO SQL 言語機能 P711 の一部です。ただし、標準では、ALTER DOMAIN では、変更された DEFAULT または CHECK 制約句を既存のドメインに対して指定できます。ソフトウェアでは、このいずれの操作もサポートされていません。機能 F711 では、ドメイン名の変更はサポートされていません。

例

次の例は、架空の Address ドメインの名前を MailingAddress に変更します。

```
ALTER DOMAIN Address RENAME MailingAddress;
```

関連情報

ドメイン [184 ページ]

SYSUSERTYPE システムビュー [1858 ページ]
CREATE DOMAIN 文 [794 ページ]

1.4.4.5 ALTER EVENT 文

イベントの定義、またはイベントに関連付けて定義済みアクションを自動化するイベントハンドラの定義を変更したり、スケジュールされたアクションの定義を変更したりします。この文を使用して、イベントハンドラの定義を隠すこともできます。

構文

イベントの変更

```
ALTER EVENT event-name
[ AT { CONSOLIDATED | REMOTE | ALL } ]
[ FOR { PRIMARY | ALL} ]
[ { DELETE TYPE
   | TYPE event-type
   | WHERE { trigger-condition | NULL }
   | { ADD | ALTER | DELETE } SCHEDULE schedule-spec } ]
[ ENABLE | DISABLE ]
[ [ ALTER ] HANDLER compound-statement | DELETE HANDLER ]
```

```
event-type :
BackupEnd
| Connect
| ConnectFailed
| DatabaseStart
| DBDiskSpace
| Deadlock
| "Disconnect"
| GlobalAutoincrement
| GrowDB
| GrowLog
| GrowTemp
| LogDiskSpace
| RAISERROR
| ServerIdle
| TempDiskSpace
```

```
trigger-condition :
event_condition( condition-name ) { = | < | > | != | <= | >= } value | @variable-
name
```

```
schedule-spec :
[ schedule-name ]
   { START TIME start-time | BETWEEN start-time AND end-time }
   [ EVERY period { HOURS | MINUTES | SECONDS } ]
   [ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]
   [ START DATE start-date ]
```

```
event-name | schedule-name :  identifier
```

```
day-of-week :  string
```

```
value | period | day-of-month :  integer
```

```
start-time | end-time : time
```

```
start-date : date
```

イベントハンドラの定義を非表示にする

```
ALTER EVENT event-name SET HIDDEN
```

パラメータ

ALTER EVENT 句

イベントは、所有者を持ちません。所有者 (たとえば、owner.event-name) を指定すると、所有者部分は無視されます。

AT 句

この句は、イベントを処理するデータベースに関する指定を変更するときに使用します。

FOR 句

この句をデータベースミラーリングまたは読み込み専用スケールアウトシステムで使用して、イベントが処理されるデータベースを制限します。

DELETE TYPE 句

この句は、イベントとイベントタイプの関連付けを解除するときに使用します。

ADD | ALTER | DELETE SCHEDULE 句

この句は、スケジュールの定義を変更するときに使用します。1 つの ALTER EVENT 文で 1 つのスケジュールしか変更できません。

WHERE 句

この句は、イベント発生のトリガ条件を変更するときに使用します。WHERE NULL オプションは条件を削除します。

event_condition 値の変数名を指定できます。

START TIME 句

この句は、イベントの開始時刻とオプションで終了時刻を指定するときに使用します。start-time パラメータと end-time パラメータは文字列です (たとえば、'12:34:56')。式は使用できません (たとえば、NOW())。

start-time の変数名を指定できます。start-time は、NULL 文字列変数の場合は無視されます。

BETWEEN...AND 句 start-time および end-time の変数名を指定できます。start-time または end-time は、NULL 文字列変数の場合は無視されます。

EVERY 句 period の変数名を指定できます。period が NULL 整数変数の場合、EVERY 句は無視されます。

START DATE 句

この句は、イベントの開始日を指定するときに使用します。start-date パラメータは、文字列です。式は使用できません (たとえば、TODAY())。start-date は、NULL 文字列変数の場合は無視されます。

start-date の変数名を指定できます。

SET HIDDEN 句

この句は、イベントハンドラの定義を隠すときに使用します。SET HIDDEN 句を指定すると、ISYSEVENT システムテーブルの action カラムに格納されているイベントハンドラの定義が永続的に難読化されます。

備考

この文によって、CREATE EVENT で作成されたイベント定義を変更することができます。この文は以下の目的に使用できます。

- イベントハンドラの定義を非表示にする
- 開発段階でトリガ条件やスケジュールを指定せずにイベントハンドラを定義、テストし、イベントハンドラの完成後に ALTER EVENT を使って実行条件を追加する

イベントは所有されていません。ただし、イベントが作成されると、ユーザ名がイベントに関連付けられます (ユーザ名を CREATE EVENT 文で明示的に指定するか、作成者として暗黙的に指定)。イベントはユーザ名の権限で実行されます。ALTER EVENT 文は、イベントに関連付けられたユーザ名の変更には使用できません (ユーザ名を指定すると無視されます)。代わりに、イベントを削除して再作成し、新しいユーザ名を指定する必要があります。

イベントを変更する必要がある場合は、ALTER EVENT...DISABLE 文を実行すると実行中のイベントを無効にできます。*SQL Central* でイベントを無効にするには、イベントを右クリックし、有効化オプションをクリアします。イベントを無効にしても、現在のイベントハンドラの実行は中断されません。イベントハンドラの実行は、完了するまで続行されます。完了したイベントハンドラは、再度有効に設定されるまで再開されません。定義は、変更して再度有効にすることができます。実行中のイベントを確認するには、次の文を実行します。

```
SELECT *
FROM dbo.sa_conn_info()
WHERE CONNECTION_PROPERTY( 'EventName',Number ) = 'event-name';
```

`event-name` の前の `owner` の指定は無視されます。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

ALTER ANY OBJECT または MANAGE ANY EVENT のどちらかのシステム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

1. 次の例は、BETWEEN 句の変数を使用してイベントを作成して変更します。
   1. 次の文は、開始時刻、終了時刻、時間間隔の 3 つの変数を作成します。

      ```
      CREATE VARIABLE @st1 LONG VARCHAR
      CREATE VARIABLE @et1 LONG VARCHAR
      CREATE VARIABLE @int1 INTEGER
      SET @st1 = ' 8:00AM '
      SET @et1 = ' 6:00PM '
      SET @int1 = 1;
      ```

   2. 次の文は、データベースのトランザクションログをバックアップするイベントを作成し、作成された変数を BETWEEN 句に使用します。

      ```
      CREATE EVENT HourlyLogBackup
         SCHEDULE hourly_log_backup
         BETWEEN @st1 AND @et1
         EVERY @int1 HOURS ON
            ( 'Monday' , 'Tuesday' , 'Wednesday' , 'Thursday' , 'Friday' )
            HANDLER
               BEGIN
                  BACKUP DATABASE DIRECTORY 'C:¥¥database¥¥backup'
                  TRANSACTION LOG ONLY
                  TRANSACTION LOG RENAME
               END;
      ```

   3. 次の文は、変数 @st1 を 'Now' にリセットします。

      ```
      SET @st1 = ' Now ';
      ```

   4. 次の文は、開始時間を 'Now' に変更してイベントを変更します。

      ```
      ALTER EVENT HourlyLogBackup
            SCHEDULE hourly_log_backup
            BETWEEN @st1 AND @et1
            EVERY @int1 HOURS ON
               ( 'Monday' , 'Tuesday' , 'Wednesday' , 'Thursday' , 'Friday' )
               HANDLER
                  BEGIN
                     BACKUP DATABASE DIRECTORY 'C:¥¥database¥backup'
                        TRANSACTION LOG ONLY
                        TRANSACTION LOG RENAME
                  END;
      ```

2. 次の例は、増分バックアップを実行するイベントを作成し、START TIME 句、EVERY 句、および START DATE 句の変数を使用してイベントを変更します。

1. 次の文は、毎日午前 1 時に増分バックアップを実行するイベントを作成します。

```
CREATE EVENT IncrementalBackup
SCHEDULE
 START TIME '1:00 AM' EVERY 24 HOURS
   HANDLER
      BEGIN
       BACKUP DATABASE DIRECTORY 'c:¥¥backup'
       TRANSACTION LOG ONLY
       TRANSACTION LOG RENAME MATCH
      END;
```

3. 次の文は、開始時刻、開始日付、時間間隔の 3 つの変数を作成します。

```
CREATE VARIABLE @st2 LONG VARCHAR
CREATE VARIABLE @sd2 LONG VARCHAR
CREATE VARIABLE @int2 INTEGER
SET @st2 = ' 3:00AM '
SET @sd2 = '2013-01-01'
SET @int2 = 24;
```

4. 次の例は、イベント IncrementalBackup を変更し、START TIME 句、EVERY 句、および START DATE 句の変数を使用します。

```
ALTER EVENT IncrementalBackup
   SCHEDULE
      START TIME @st2 EVERY @int2 HOURS
      START DATE @sd2
         HANDLER
            BEGIN
               BACKUP DATABASE DIRECTORY 'c:¥¥backup'
               TRANSACTION LOG ONLY
               TRANSACTION LOG RENAME MATCH
            END;
```

関連情報

SYSEVENT システムビュー [1800 ページ]
BEGIN 文 [745 ページ]
CREATE EVENT 文 [805 ページ]

1.4.4.6 ALTER EXTERNAL ENVIRONMENT 文

Java、PHP、Perl などの外部環境のロケーションを指定します。

構文

```
ALTER EXTERNAL ENVIRONMENT environment-name
LOCATION location-string
```

```
environment-name :
C_ESQL32
```

| *C_ESQL64*
| *C_ODBC32*
| *C_ODBC64*
| *CLR*
| *DBMLSYNC*
| *JAVA*
| *JS*
| *PERL*
| *PHP*

パラメータ

environment-name

`environment-name` は、変更する外部環境を指定するときに使用します。

location-string

`location-string` は、外部環境の実行ファイル/バイナリファイルがあるデータベースサーバコンピュータ上の場所を指定します。この句には、実行プログラム/バイナリの名前を指定します。このパスは、完全に修飾されたパスまたは相対パスのどちらでもかまいません。相対パスの場合、実行プログラム/バイナリは、サーバが検出できるロケーションに存在する必要があります。

備考

通常、データベースサーバが標準検索手法 (ソフトウェアのインストール場所、システム PATH などの検出) を使用して外部環境モジュールを検出できない場合に、ALTER EXTERNAL ENVIRONMENT 文を使用します。

CLR: LOCATION 文字列は CLR 外部環境サポートモジュールの場所およびバージョンの両方の識別に使用されます。たとえば、`dbextclr[VER_MAJOR]_v4.5` は .NET 4.x のサポートが必要なことを示します。この例ではファイルパスが指定されていないため、データベースサーバは標準的な検索手法を使用してモジュールを検出します。`dbextclr[VER_MAJOR]` モジュールは .NET 2 および 3.5 をサポートします。`dbextclr[VER_MAJOR]_v4.5` モジュールは .NET 4.x をサポートします。ソフトウェアの新しいバージョンに移植できるように、LOCATION 文字列には現在のソフトウェアリリースバージョン番号ではなく *[VER_MAJOR]* を使用してください。データベースサーバによって [VER_MAJOR] が適切なバージョン番号に置き換えられます。

DBMLSYNC: データベースサーバソフトウェアの複数バージョンが同一システムにインストールされている場合や、dbmlsync 実行プログラムがデータベースサーバと同じディレクトリに配置されていないために PATH 環境変数が見つからない場合は、ALTER EXTERNAL ENVIRONMENT コマンドを使用して dbmlsync 実行プログラムの場所を指定してください。

権限

MANAGE ANY EXTERNAL ENVIRONMENT システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Perl を外部環境として使用するときに使用する Perl 実行ファイルを指定します。

```
ALTER EXTERNAL ENVIRONMENT PERL
LOCATION 'c:¥¥Perl64¥¥bin¥¥perl.exe';
```

関連情報

START EXTERNAL ENVIRONMENT 文 [1338 ページ]
STOP EXTERNAL ENVIRONMENT 文 [1348 ページ]
INSTALL EXTERNAL OBJECT 文 [1173 ページ]
REMOVE EXTERNAL OBJECT 文 [1263 ページ]
SYSEXTERNENV システムビュー [1802 ページ]
SYSEXTERNENVOBJECT システムビュー [1804 ページ]

1.4.4.7 ALTER FUNCTION 文

関数を変更します。

構文

関数定義の変更

```
ALTER FUNCTION [ owner.]function-name function-definition
```

```
function-definition: CREATE FUNCTION 文を参照
```

関数定義の難読化

```
ALTER FUNCTION [ owner.]function-name
SET HIDDEN
```

関数の再コンパイル

```
ALTER FUNCTION [ owner.]function-name
RECOMPILE
```

備考

新しい関数全体を ALTER FUNCTION 文にインクルードします。

関数定義の変更

ALTER FUNCTION 文の構文は、最初の 1 語を除き、CREATE FUNCTION 文の構文とまったく同じです。

ALTER FUNCTION では、関数の既存の権限は変更されません。ただし、DROP FUNCTION に続けて CREATE FUNCTION を実行した場合は、実行権限が再割り当てされます。

関数定義の難読化

SET HIDDEN を使用して、関連する関数定義を難読化し、解読できないようにします。この関数はアンロードして、他のデータベースに再ロードできます。

SET HIDDEN を使用すると、デバッガを使用したデバッグでも、プロシージャプロファイリングによっても、関数定義は表示されません。

> **注記**
>
> この設定は、元に戻せません。元の関数定義をデータベースの外部に保持してください。

関数の再コンパイル

RECOMPILE 構文を使用して、ユーザ定義 SQL 関数を再コンパイルします。関数を再コンパイルすると、カタログに格納された定義が再解析され、構文が検証されます。保持されている関数のソースは、再コンパイルしても変わりません。関数を再コンパイルすると、SET HIDDEN 句で難読化された定義は、難読化された状態が維持され、解読できません。

> **注記**
>
> 変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。
>
> - 変数が存在しない
> - 変数の内容が NULL
> - 変数がパラメータで許可されている長さを超えている
> - 変数のデータ型がパラメータで要求されているものと一致していない

権限

その関数の所有者であるか、または次のいずれかの権限を持っていることが必要です。

- ALTER ANY PROCEDURE システム権限
- ALTER ANY OBJECT システム権限

関数を外部関数にするには、CREATE EXTERNAL REFERENCE システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

ALTER FUNCTION は、オプションの ANSI/ISO SQL 言語機能 F381 です。ただし、SQL 標準では、SQL Persistent Stored Module (PSM) 関数定義を再定義するために ALTER FUNCTION を使用することはできません。ANSI/ISO SQL 標準では、SET HIDDEN または RECOMPILE はサポートされていません。

例

1. この例では、MyFunction が作成され、変更されます。SET HIDDEN 句は、関数を難読化し、解読できないようにします。この例を実行するには、変更される前に関数が作成されるため、CREATE PROCEDURE システム権限も必要です。

```
CREATE FUNCTION MyFunction(
   firstname CHAR(30),
   lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
   DECLARE name CHAR(61);
   SET name = firstname || ' ' || lastname;
   RETURN (name);
ALTER FUNCTION MyFunction SET HIDDEN;
END;
```

2. 次の例は、NAMESPACE 句の変数を使用して関数を作成して変更します。
   1. 次の文は、NAMESPACE 句の変数を作成します。

```
CREATE VARIABLE @ns LONG VARCHAR ;
SET @ns = 'http://wsdl.domain.com/' ;
```

   2. 次の文は、NAMESPACE 句の変数を使用する FtoC という名前の関数を作成します。

```
CREATE FUNCTION FtoC ( IN temperature LONG VARCHAR )
RETURNS LONG VARCHAR
URL 'http://localhost:8082/FtoCService'
TYPE 'SOAP:DOC'
NAMESPACE @ns;
```

   3. 次の文は、FLOAT データ型を受け付けて返せるよう FtoC 関数を変更します。

```
ALTER FUNCTION FtoC ( IN temperature FLOAT )
RETURNS FLOAT
URL 'http://localhost:8082/FtoCService'
NAMESPACE @ns;
```

関連情報

CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
ALTER PROCEDURE 文 [669 ページ]
DROP FUNCTION 文 [1039 ページ]

1.4.4.8 ALTER INDEX 文

インデックス、プライマリキー、または外部キーの名前を変更したり、インデックスのクラスタ化された内容を変更したりします。

構文

```
ALTER { INDEX index-name
| [ INDEX ] FOREIGN KEY role-name
| [ INDEX ] PRIMARY KEY }
ON [ owner.]object-name { REBUILD | rename-clause | cluster-clause }
```

```
object-name : table-name | materialized-view-name
```

```
rename-clause : RENAME { AS | TO } new-index-name
```

```
cluster-clause : CLUSTERED | NONCLUSTERED
```

パラメータ

rename-clause

インデックス、プライマリキー、または外部キーの新しい名前を指定します。

外部キーまたはプライマリキーの基本となるインデックスの名前を変更しても、インデックスの対応する RI 制約の名前は変更されません。ただし、外部キーの役割名 (該当する場合) はインデックス名と同じであり、変更されます。必要に応じて ALTER TABLE 文を使用して、RI 制約名を変更します。

cluster-clause

インデックスを CLUSTERED と NONCLUSTERED のどちらに変更するかを指定します。特定のテーブル上で 1 つのインデックスのみ、クラスタ化できます。

REBUILD 句

この句は、インデックスを削除して再作成するのではなく、インデックスを再構築するときに使用します。

備考

ALTER INDEX 文は、次の 2 つのタスクを実行します。

- インデックス、プライマリキー、または外部キーの名前を変更するときに使用します。
- インデックスタイプをノンクラスタドからクラスタドに、またはその逆に変更するときに使用します。
  ALTER INDEX 文は、インデックスのクラスタ指定の変更には使用できますが、データの再編成はしません。また、特定のテーブルまたはマテリアライズドビュー上で 1 つのインデックスのみ、クラスタ化できます。

ローカルテンポラリテーブル上では ALTER INDEX を使用してインデックスを変更できません。この文を使用してインデックスを削除しようとすると、"インデックスが見つかりません。" エラーになります。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

権限

テーブルのインデックスを変更するには、テーブルの所有者であるか、次のいずれかの権限を持っている必要があります。

- そのテーブルに対する REFERENCES 権限
- ALTER ANY INDEX システム権限
- ALTER ANY OBJECT システム権限

マテリアライズドビューのインデックスを変更するには、マテリアライズドビューの所有者であるか、次のいずれかの権限を持っている必要があります。

- ALTER ANY INDEX システム権限
- ALTER ANY OBJECT システム権限

関連する動作

オートコミット。現在接続しているすべてのカーソルを閉じます。ALTER INDEX REBUILD が指定されている場合は、チェックポイントが実行されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、IX_product_name をクラスタドインデックスに変更します。

```
ALTER INDEX IX_product_name ON GROUPO.Products
```

```
CLUSTERED;
```

次の文は、Products テーブルのインデックス IX_product_name の名前を ixProductName に変更します。

```
ALTER INDEX IX_product_name ON GROUPO.Products
RENAME TO ixProductName;
```

関連情報

CREATE INDEX 文 [842 ページ]
ALTER TABLE 文 [705 ページ]

1.4.4.9 ALTER LDAP SERVER 文

LDAP サーバ設定オブジェクトを変更します。

構文

```
ALTER LDAP SERVER ldapua-server-name
[ ldapua-server-attribs ... ]
[ WITH { SUSPEND | ACTIVATE | REFRESH } ]
```

```
ldapua-server-attribs :
SEARCH DN search-dn-attributes ...
| AUTHENTICATION URL { 'url-string' | NULL }
| CONNECTION TIMEOUT timeout-value
| CONNECTION RETRIES retry-value
| TLS { ON | OFF }
```

```
search-dn-attributes :
URL { 'url-string' | NULL }
| ACCESS ACCOUNT { 'dn-string' | NULL }
| IDENTIFIED BY ( 'password' | NULL }
| IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
```

パラメータ

SEARCH DN 句

SEARCH DN 句のパラメータにはデフォルト値がありません。

URL

この句を使用して、ホストを (名前または IP アドレスで) 指定したり、ポート番号を指定したり、指定されたユーザ ID の LDAP 識別名 (DN) のルックアップを行うために実行する検索を指定します。`url-string` は正しい LDAP

URL 構文であることが検証されてから ISYSLDAPSERVER に格納されます。この文字列の最大サイズは 1024 バイトです。

`url-string` のフォーマットは、LDAP URL 標準に準拠している必要があります。LDAP 標準仕様 を参照してください。

ACCESS ACCOUNT

この句を使用して、データベースサーバによって LDAP サーバに接続するために使用される LDAP 識別名 (DN) を指定します。これは SQL Anywhere ユーザではなく、LDAP サーバへのログイン用に LDAP サーバで作成されたユーザです。このユーザは、SEARCH DN URL 句で指定される場所にあるユーザ ID によって DN を検索するために、LDAP サーバ内でパーミッションを得る必要があります。この文字列の最大サイズは 1024 バイトです。

IDENTIFIED BY

この句を使用して、ACCESS ACCOUNT で識別されるユーザに関連付けられたパスワードを指定します。最大サイズは 255 バイトで、NULL には設定できません。

IDENTIFIED BY ENCRYPTED

この句を使用して、暗号化形式で提供され、ディスクのいずれかの場所に保存されたバイナリ値である、ACCESS ACCOUNT で識別されるユーザに関連付けられたパスワードを指定します。バイナリの最大サイズは 289 バイトで、NULL には設定できません。IDENTIFIED BY ENCRYPTED により、パスワードを既知にせずに取得および使用できます。

AUTHENTICATION URL 句

この句を使用して、ホストを名前または IP アドレスで指定したり、ユーザ認証に使用する LDAP サーバのポート番号を指定します。以前の DN 検索およびユーザパスワードから取得したユーザの DN は、新しい接続を認証 URL にバインドするために使用されます。LDAP サーバへの正常な接続は、接続ユーザの ID の証明とみなされます。このパラメータにはデフォルト値はありません。

CONNECTION TIMEOUT 句

この句を使用して、DN の検索と認証の両方に対する LDAP サーバへの接続タイムアウトをミリ秒単位で指定します。デフォルト値は 10 秒です。

CONNECTION RETRIES 句

この句を使用して、DN の検索と認証の両方に対する LDAP サーバへの接続再試行回数をミリ秒単位で指定します。値の有効範囲は 1 ～ 60 です。デフォルトは 3 です。

TLS 句

この句を使用して、DN 検索と認証の両方について、LDAP サーバへの接続時における TLS プロトコルの使用を指定します。有効な値は ON と OFF です。デフォルトは OFF です。セキュア LDAP プロトコルを使用するには、URL の先頭に `ldap://` ではなく `ldaps://` を指定します。セキュア LDAP を使用するときは、TLS オプションを OFF に設定する必要があります。

WITH 句

WITH SUSPEND

LDAP サーバの通信状態を SUSPENDED (メンテナンスモード) に設定します。LDAP サーバへの接続は閉じられ、LDAP サーバによる認証は実行されなくなります。

WITH ACTIVATE

LDAP サーバをアクティブにしてすぐに使用できるようにします。これによって LDAP サーバの通信状態が READY に変わります。

WITH REFRESH

LDAP ユーザ認証を再初期化します。このコマンドは LDAP サーバの状態が SUSPENDED のときはその状態を変更しません。LDAP サーバが READY または ACTIVE 状態のときに WITH REFRESH を指定すると、LDAP サーバへの接続は閉じられます。その後、サーバオプションの値が ISYSLDAPSERVER システムテーブルから再読み込みされ、LDAP サーバへの新しい接続と、データベースサーバへの受信認証要求に適用されます。

備考

ALTER LDAP SERVER...WITH REFRESH は、多くの場合 LDAP サーバが ACTIVE または READY 状態のときに使用され、保持されているすべてのリソースを解放したり、サーバの外部のファイルに加えられた変更 (trusted_certificates_file データベースオプションによって指定されたファイルの内容に対する変更など) を再読み込みします。

その他の状態の場合、ALTER LDAP SERVER...WITH REFRESH は効果がありません。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE ANY LDAP SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、apps_primary という名前の架空の LDAP サーバを中断します。

```
ALTER LDAP SERVER apps_primary WITH SUSPEND;
```

次の例では、apps_primary という名前の LDAP サーバを、ホスト fairfax、ポート番号 1066 で認証を行うために異なる URL を使用するように変更し、接続リトライを 10 回に設定して、そのサーバをアクティブにします。

```
ALTER LDAP SERVER apps_primary
 AUTHENTICATION URL 'ldap://fairfax:1066/'
```

```
CONNECTION RETRIES 10
WITH ACTIVATE;
```

関連情報

SYSLDAPSERVER システムビュー [1814 ページ]
CREATE LDAP SERVER 文 [847 ページ]
DROP LDAP SERVER 文 [1042 ページ]
VALIDATE LDAP SERVER 文 [1398 ページ]

1.4.4.10 ALTER LOGIN POLICY 文

既存のログインポリシーを変更します。

構文

```
ALTER LOGIN POLICY policy-name policy-options
```

```
policy options :
policy-option [ policy-option ... ]
```

```
policy-option :
policy-option-name = policy-option-value
```

```
policy-option-value :
{ UNLIMITED
| DEFAULT
| legal-option-value }
```

パラメータ

policy-name

ログインポリシーの名前。ルートを指定してルートログインポリシーを修正します。

policy-option-name

ポリシーオプションの名前。

policy-option-value

ログインポリシーオプションに割り当てられている値。UNLIMITED を指定すると、制限は使用されません。DEFAULT を指定すると、デフォルトの制限が使用されます。

備考

ログインポリシーが変更されるとすぐに、すべてのユーザに変更が適用されます。

ポリシーオプションを指定しない場合は、ルートログインポリシーからこのログインポリシーの値が取得されます。新しいポリシーは、MAX_NON_DBA_CONNECTIONS および ROOT_AUTO_UNLOCK_TIME ポリシーオプションを継承しません。

すべての新しいデータベースには、ルートログインポリシーが含まれています。ルートログインポリシーの値を変更することはできますが、ポリシーは削除できません。ルートログインポリシーのデフォルト値の概要は上記の表に示されています。

権限

MANAGE ANY LOGIN POLICY システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、LOCKED および MAX_CONNECTIONS ポリシーオプションを変更することによって、架空の Test1 ログインポリシーを改変します。LOCKED の値は、このポリシーを割り当てられたユーザが新しい接続を確立できないことを示し、MAX_CONNECTIONS の値は、許容される同時接続数を制限します。

```
ALTER LOGIN POLICY Test1
LOCKED=ON
MAX_CONNECTIONS=5;
```

この例はルートログインポリシー LOCKED および MAX_CONNECTIONS ポリシーオプションを上書きします。

```
ALTER LOGIN POLICY root
LOCKED=ON
MAX_CONNECTIONS=5;
```

次の例は、架空の ldap_user_policy ログインポリシーに対してプライマリ LDAP サーバとセカンダリ LDAP サーバを設定し、データベースオプション login_mode に 'Standard' が含まれていても、標準認証にフェイルオーバーできる機能をオフにします。これによって、このログインポリシーのユーザを厳格に制御し、LDAP ユーザ認証のみを使用して認証できるようにします。大量のログイン接続が発生して LDAP サーバがすぐに応答したり認証できない場合、再試行回数とタイムアウト

を使い切ったユーザは、標準認証を使用するようにフェイルオーバーするのではなく、データベースサーバへの接続エラーが表示されます。

```
ALTER LOGIN POLICY ldap_user_policy
LDAP_PRIMARY_SERVER=ldapsrv1
LDAP_SECONDARY_SERVER=ldapsrv2
LDAP_FAILOVER_TO_STD=OFF;
```

次の例は、架空の application_user_policy ログインポリシーのタイムスタンプ値を現在の時刻にリセットします。このポリシーが割り当てられたユーザは、次のログイン試行時に自分の識別名 (DN) が検索され、ISYSUSER にキャッシュされている値は使用されません。この方式では、このポリシーに関連付けられているユーザの次回の認証時に、ISYSUSER に保持されている古い DN の値がパージされます。

```
ALTER LOGIN POLICY application_user_policy
LDAP_REFRESH_DN=NOW;
```

関連情報

ALTER USER 文 [731 ページ]
COMMENT 文 [769 ページ]
CREATE LOGIN POLICY 文 [852 ページ]
CREATE USER 文 [990 ページ]
DROP LOGIN POLICY 文 [1043 ページ]
DROP USER 文 [1084 ページ]

1.4.4.11 ALTER MATERIALIZED VIEW 文

マテリアライズドビューを変更します。

構文

```
ALTER MATERIALIZED VIEW [ owner.]materialized-view-name {
 SET HIDDEN
| { ENABLE | DISABLE }
| { ENABLE | DISABLE } USE IN OPTIMIZATION
| { ADD PCTFREE percent-free-space | DROP PCTFREE }
| [ NOT ] ENCRYPTED
| [ { IMMEDIATE | MANUAL } REFRESH ]
}
```

```
percent-free-space :integer
```

パラメータ

SET HIDDEN 句

SET HIDDEN 句は、マテリアライズドビューの定義を難読化するときに使用します。この設定は、元に戻せません。

ENABLE 句

ENABLE 句は、無効にされたマテリアライズドビューを有効にして、データベースサーバで使用できるようにするときに使用します。この句は、すでに有効になっているビューには影響ありません。この句を使用した後、ビューをリフレッシュして初期化し、ビューが無効にされたときに削除されたすべてのテキストインデックスを再作成してください。

DISABLE 句

DISABLE 句は、データベースサーバからビューを使用できないようにします。マテリアライズドビューを無効にすると、データベースサーバはビューのデータとインデックスを削除します。

{ ENABLE | DISABLE } USE IN OPTIMIZATION 句

この句は、オプティマイザでマテリアライズドビューを使用できるようにするかどうかを指定するときに使用します。DISABLE USE IN OPTIMIZATION を指定する場合、マテリアライズドビューを使用するのは、明示的にそのビューを参照するクエリを実行する場合のみです。デフォルトは ENABLE USE IN OPTIMIZATION です。

ADD PCTFREE 句

各ページに確保する空き領域の割合を指定します。空き領域は、データが更新されたときにローのサイズが増えた場合に使用されます。ページに空き領域がない場合は、ページのローのサイズが増えるたびに、ローを複数のページに分割することが必要になり、ローの断片化が発生します。また、パフォーマンス低下の可能性があります。

`percent-free-space` の値は 0 ～ 100 の整数です。値 0 を指定すると、各ページに空き領域を残しません。各ページは完全にパックされます。大きい値を指定すると、各ローが単独でページに挿入されます。PCTFREE が設定されない場合、または削除された場合、データベースのページサイズに応じたデフォルトの PCTFREE 設定が適用されます (ページサイズが 4 KB の場合は 200 バイト、2 KB の場合は 100 バイト)。

DROP PCTFREE 句

現在のマテリアライズドビューで有効な PCTFREE 設定を削除し、データベースのページサイズに応じたデフォルトの PCTFREE を適用します。

[NOT] ENCRYPTED 句

マテリアライズドビューデータを暗号化するかどうかを指定します。デフォルトで、マテリアライズドビューデータは作成時に暗号化されません。マテリアライズドビューを暗号化するには、ENCRYPTED を指定します。マテリアライズドビューを復号化するには、NOT ENCRYPTED を指定します。

REFRESH 句

REFRESH 句は、マテリアライズドビューの再表示タイプを変更するときに使用します。

IMMEDIATE REFRESH

IMMEDIATE REFRESH 句は、手動ビューを即時ビューに変更するときに使用します。再表示タイプを IMMEDIATE REFRESH に変更するには、手動ビューが有効であり、初期化されていない必要があります。ビューが初期化された状態である場合は、TRUNCATE 文を実行して初期化されていない状態に変更してから、ALTER MATERIALIZED VIEW...IMMEDIATE REFRESH を実行します。

MANUAL REFRESH

MANUAL REFRESH 句は、即時ビューを手動ビューに変更するときに使用します。

備考

別のユーザが所有するマテリアライズドビューを変更する場合は、所有者を含めて名前を修飾する必要があります (たとえば、GROUPO.EmployeeConfidential)。名前を修飾しなかった場合、データベースサーバは、ユーザ本人が所有する同名のマテリアライズドビューを検索して変更します。見つからない場合は、エラーが返されます。

マテリアライズドビューを無効にすると (DISABLE 句)、データベースサーバがクエリに応答するときにマテリアライズドビューを使用できなくなります。また、データとインデックスが削除され、再表示タイプが手動に変わります。通常の従属ビューもすべて無効にされます。

DISABLE 句によって従属ビューも無効になるため、無効化されるビューだけでなく、すべての従属ビューに対する排他アクセスが必要です。

マテリアライズドビューを暗号化するには (ENCRYPTED 句)、データベースでテーブルの暗号化をあらかじめ有効にしておく必要があります。その後、データベースの作成時に指定された暗号化キーとアルゴリズムを使用してマテリアライズドビューが暗号化されます。

ユーザが自分のデータを変更するためにマテリアライズドビューに対して実行できる操作は、再表示、トランケート、無効化のみです。ただし、即時ビューは、データベースサーバによって自動的に更新されます。つまり、一度即時ビューを有効にして初期化すると、データベースサーバはさらに権限をチェックすることなく、ビューを自動的に管理します。

権限

マテリアライズドビューの所有者であるか、ALTER ANY MATERIALIZED VIEW または ALTER ANY OBJECT のシステム権限を持っていることが必要です。

必要な権限がない場合、マテリアライズドビューを即時ビューに変更する (ALTER MATERIALIZED VIEW...IMMEDIATE REFRESH) には、ビューとそのすべての参照先テーブルを所有している必要があります。

関連する動作

- オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、EmployeeConfid88 というマテリアライズドビューを作成し、それを最適化の際に使用するのを無効にします。このサンプルを実行するには、Employees テーブルと Departments テーブルに対する SELECT 権限とともに、CREATE ANY MATERIALIZED VIEW システム権限も必要になります。

警告

この例を実行し終わったら、作成したマテリアライズドビューを削除してください。このようにしないと、他の例を試すとき、基本となるテーブル Employees および Departments に対するスキーマ変更を実行できなくなります。有効化されている従属マテリアライズドビューを持つテーブルのスキーマは変更できません。

```
CREATE MATERIALIZED VIEW EmployeeConfid88 AS
   SELECT EmployeeID, Employees.DepartmentID, SocialSecurityNumber, Salary,
ManagerID,
      Departments.DepartmentName, Departments.DepartmentHeadID
   FROM GROUPO.Employees, GROUPO.Departments
   WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid88;
ALTER MATERIALIZED VIEW EmployeeConfid88 DISABLE USE IN OPTIMIZATION;
```

関連情報

CREATE MATERIALIZED VIEW 文 [854 ページ]
REFRESH MATERIALIZED VIEW 文 [1252 ページ]
DROP MATERIALIZED VIEW 文 [1045 ページ]
TRUNCATE 文 [1366 ページ]
sa_refresh_materialized_views システムプロシージャ [1591 ページ]

1.4.4.12 ALTER MIRROR SERVER 文

ミラーサーバの属性を変更します。

構文

ミラーサーバの削除

```
ALTER MIRROR SERVER mirror-server-name
[AS { PRIMARY | MIRROR | ARBITER | PARTNER}]
[ server-option = { string | NULL } [ ... ] ]
```

ミラーサーバをコピーとして変更

```
ALTER MIRROR SERVER mirror-server-name
[AS COPY]
[ { FROM SERVER parent-name [ OR SERVER server-name ] | USING AUTO PARENT } |
ALTER PARENT FROM mirror-server-name ]
[ server-option = { string | NULL } [ ... ] ]
```

```
parent-name :
server-name | PRIMARY
```

```
server-option :
connection_string
```

logfile
preferred
state_file

パラメータ

AS 句

AS 句を使用して、ミラーサーバのサーバタイプを PARTNER から COPY または COPY から PARTNER に変更します。ミラーサーバのタイプを変更しない場合、この句は不要であり、推奨されません。

PARTNER

ミラーサーバのタイプが COPY であるデータベースサーバのみ、この値を使用してそのタイプを PARTNER に変更できます。コピーノードの親の定義は削除されます。

ミラーサーバ名は、-n サーバオプションで指定されたデータベースサーバ名に対応している必要があり、connection_string ミラーサーバオプションで指定された SERVER 接続文字列と一致している必要があります。

データベースミラーリングシステムでは、パートナは接続文字列の値を使用して相互に接続します。読み込み専用スケールアウトシステムでは、その親として現在のサーバを備えたコピーノードによって接続文字列が使用されます。

COPY

ミラーサーバのタイプが PARTNER であるデータベースサーバのみ、この値を使用してそのサーバタイプをコピーノードに変更できます。このパートナサーバには現時点で MIRROR ロールも必要です。

読み込み専用のスケールアウトシステムでは、この値は、データベースサーバがコピーノードであることを指定します。このサーバ上のデータベース接続はすべて、読み込み専用です。ミラーサーバ名は、-n サーバオプションで指定されたデータベースサーバ名に対応している必要があり、connection_string ミラーサーバオプションで指定された SERVER 接続文字列と一致している必要があります。

FROM SERVER 句

この句を使用できるのは、AS COPY が指定されている場合だけです。この句はスケールアウトシステムのサーバのツリーを構成し、コピーノードがどのサーバからトランザクションログページを取得するのかを示します。

ミラーサーバ名または PRIMARY を使用して、親を指定できます。コピーノードの代替の親は、OR SERVER 句を使用して指定できます。

2 レベル (パートナノードとコピーノード) のみが存在するデータベースミラーリングシステムでは、コピーノードが、現在のプライマリサーバまたはミラーサーバからトランザクションログページを取得します。

コピーノードは、データベースに格納されているコピーノードのミラーサーバ定義を使用して、接続先となるサーバを判断します。コピーノードの定義から親の定義を探すことができ、親の定義から親に接続するための接続文字列を取得できます。

スケールアウトシステムにコピーノードを明示的に定義する必要はありません。接続時に、ルートノードによってコピーノードを定義できます。

USING AUTO PARENT 句

この句を使用できるのは、AS COPY が指定されている場合だけです。この句を指定すると、このサーバの親がプライマリサーバによって割り当てられます。

ALTER PARENT FROM 句

この句を使用できるのは、AS COPY が指定されている場合だけです。この句は、このミラーサーバの親を変更し、ミラーサーバのすべての兄弟を子に割り当てます。ALTER PARENT FROM 句で指定したサーバ名は、このサーバの現在の親が指定の値と一致するかどうかを確認するために使用されます。これは、すべての兄弟が同時に変更を要求した場合に、いずれか 1 つの兄弟のみが親を置き換えることができるようにするために使用されます。

server-option 句

`server-option` の変数名を指定できます。

次のオプションもサポートされています。

connection_string サーバオプション

サーバに接続するための接続文字列を指定します。1 つのミラーサーバを別のミラーサーバに接続するときにユーザ ID とパスワードは使用されないため、ミラーサーバの接続文字列にはユーザ ID とパスワードを含めないでください。

logfile サーバオプション

データベースミラーリングが使用されている場合に、ミラーサーバ間で送信されるファイル (1 行に 1 つの要求を含む) のロケーションを指定します。このファイルは、デバッグ専用です。

preferred サーバオプション

サーバがミラーリングシステムにおいて優先サーバであるかどうかを指定します。YES または NO を指定できます。優先サーバは、可能なかぎり、プライマリサーバのロールを引き受けます。このオプションは、PARTNER サーバを定義する場合に指定します。

state_file サーバオプション

ミラーリングシステムに関するステータス情報の管理に使用されるファイルのロケーションを指定します。データベースミラーリングには、このオプションは必須です。ミラーリングシステムでは、PARTNER タイプのサーバにはステータスファイルを指定してください。監視サーバの場合は、サーバの起動コマンドの一部としてロケーションが指定されます。

備考

読み込み専用のスケールアウトとデータベースミラーリングには、それぞれ別途ライセンスが必要です。

データベースミラーリングシステムでは、ミラーサーバのタイプに PRIMARY、MIRROR、ARBITER、または PARTNER を使用できます。

読み込み専用のスケールアウトシステムでは、ミラーサーバのタイプに PRIMARY、PARTNER、または COPY を使用できます。

ミラーサーバのタイプは、COPY から PARTNER、または PARTNER から COPY にのみ変更できます。PRIMARY、MIRROR、ARBITER サーバタイプから変更したり、それらのサーバタイプに変更したりするには、ミラーサーバの定義を削除して再作成する必要があります。

PARTNER および COPY タイプのサーバのミラーサーバ名は、ミラーリングシステムの一部であるデータベースサーバの名前 (-n サーバオプションで使用される名前) と一致している必要があります。この要件により、各データベースサーバで、サーバ自体の定義とその親の定義を見つけることができるようになります。前述の `mirror-server-name`、`parent-name`、および `server-name` には、7 ビット ASCII 文字以外を含めることはできません。

コピーノードからパートナに変換すると、パートナ定義はミラーサーバ定義から削除されます。

ミラーサーバ定義を置き換えるには、CREATE MIRROR SERVER 文を OR REPLACE 句とともに使用します。

注記

変数名を受け付けるパラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

MANAGE ANY MIRROR SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

1. この例では次の処理を行います。
   1. scaleout_primary1 というプライマリサーバを作成します。
   2. scaleout_primary1 に対してコピーノード scaleout_child1 を作成します。
   3. ミラーサーバ scaleout_mirror1 を作成します。
   4. ALTER MIRROR SERVER 文を使用して、scaleout_child1 を scaleout_mirror1 のコピーノードになるように再割り当てします。

```
CREATE MIRROR SERVER "scaleout_primary1"
   AS PRIMARY
   connection_string =
'server=scaleout_primary1;host=winxp-2:6871,winxp-3:6872';
CREATE MIRROR SERVER "scaleout_child1"
   AS COPY FROM SERVER  "scaleout_primary1"
   connection_string = 'server=scaleout_child1;host=winxp-2:6878';
CREATE MIRROR SERVER "scaleout_mirror1"
   AS MIRROR
   connection_string =
'server=scaleout_mirror1;host=winxp-2:6871,winxp-3:6872';
ALTER MIRROR SERVER "scaleout_child1"
```

```
    FROM SERVER "scaleout_mirror1"
    connection_string = 'server=scaleout_child1;host=winxp-2:6878';
```

2. 次の例は、connection_string パラメータの変数を使用してミラーサーバを変更します。
   1. 次の文は、接続文字列の変数を作成します。

```
CREATE VARIABLE @connstr_value LONG VARCHAR
SET @connstr_value = ' server=new_scaleout_primary;host=winxp-2:6878 ';
```

   2. 次の文は、connection_string パラメータの変数 @connstr_value を使用してミラーサーバ new_scaleout_primary を変更します。

```
ALTER MIRROR SERVER new_scaleout_primary AS PRIMARY
connection_string = @connstr_value = @connstr_value;
```

関連情報

SYSMIRRORSERVER システムビュー [1818 ページ]
CREATE MIRROR SERVER 文 [858 ページ]
COMMENT 文 [769 ページ]
DROP MIRROR SERVER 文 [1047 ページ]

1.4.4.13 ALTER ODATA PRODUCER 文

OData プロデューサを変更します。

構文

```
ALTER ODATA PRODUCER name [ producer-clause... ]
```

```
producer-clause:
[ ADMIN USER { NULL | user } ]
[ AUTHENTICATION [ DATABASE | USER user ] ]
[ [ NOT ] ENABLED ]
[ MODEL [ FROM ] { FILE string-or-variable [ ENCODING string ] | VALUE string-
or-variable [ ENCODING string ] } ]
[ [ SERVICE ] ROOT string ]
[ USING { string | NULL } ]
```

パラメータ

ADMIN USER clause OData プロデューサがテーブルの作成に使用する管理ユーザ、プロデューサ、および繰返可能要求を管理するイベントを指定します。CREATE TABLE、CREATE PROCEDURE、MANAGE ANY EVENT、および

VERIFY ODATA システム権限が必要です。このユーザは、AUTHENTICATION 句で指定したユーザと同じにすることはできません。

AUTHENTICATION clause OData サーバのデータベースへの接続方法を指定します。パーソナル化クレデンシャルで接続するユーザには、DATABASE を指定します。これらのクレデンシャルは、HTTP 基本認証を使用することで要求されます。指定したユーザの接続文字列を使用して接続するすべてのユーザに対して、USER を指定します。指定したユーザには、VERIFY ODATA システム権限が必要です。USER は、テストの場合や読み取り専用プロデューサに対して使用することを推奨します。

ENABLED clause OData プロデューサが有効か無効かを指定します。デフォルトでは、OData プロデューサは有効です。

MODEL clause

どのテーブルとビューが OData メタデータに公開されるかを示す OData プロデューサのサービスモデルを (ODSL に) 指定します。FILE 句を指定する場合は、値は変数または ODSL データを含むファイル名の文字列にする必要があります。ファイル名が完全修飾パスでない場合、パスはデータベースサーバの現在の作業ディレクトリからの相対パスです。ファイル名が指定される場合、ファイルが読み込まれ、ファイルのコンテンツがデータベースに格納されてトランザクションログに記録されます。デフォルトでは、テーブルとビューはユーザ権限に基づいて公開されます。プライマリキーのないテーブルとビューは公開されません。

ENCODING 句を使用してモデルデータまたはファイルの文字セットを指定します。デフォルトは UTF-8 です。

SERVICE ROOT clause

OData サーバで OData サービスのルートを指定します。この OData プロデューサのすべてのリソースには、次のフォーマットの URI を使用してアクセスします。`scheme:host:port/path-prefix/resource-path[query-options]`。

SERVICE ROOT 句は `/path-prefix` です。値は NULL にできません。また、`/` から開始する必要があります。

USING clause

この文字列は、追加の OData プロデューサオプションを指定します。オプションは、name=value のペアをセミコロンで区切ったリストで指定します。USING 句が変更される場合、維持するすべての値を繰り返し指定する必要があります。

オプション	説明
ConnectionPoolMaximum =`num-max-connections`	この OData プロデューサが接続プールで使用するために開き続ける同時接続の最大数を示します。 少ない接続は、サーバの負荷に応じて、接続プールによって使用されます。 デフォルトでは、接続プールのサイズは、データベースサーバでサポートされる同時接続の最大数の半分に制限されています。
CSRFTokenTimeout =`num-seconds-valid`	CSRF トークンチェックを有効にし、トークンを有効にする秒数を指定します。 デフォルトでは、この値は 0 です。つまり、CSRF トークンチェックは無効です。それ以外の場合は、秒数は 1 ～ 1800 の有効な整数である必要があります。

オプション	説明
PageSize =num-max-entities	次のリンクを発行する前に、取得エンティティセット応答に含めるエンティティの最大数を指定します。 デフォルト設定は 100 です。
ReadOnly = { *true* | *false* }	変更要求を無視するかどうかを示します。 デフォルト設定は false です。
RepeatRequestForDays = { days-number }	繰返可能要求を有効にする長さを指定します。 この値は、1 ～ 31 の範囲内の整数である必要があります。 デフォルト設定は 2 です。 このオプションは、ADMIN USER 句が NULL ではない場合のみ有効です。
SecureOnly = { *true* | *false* }	プロデューサが HTTPS ポートでのみ要求のみを受信するかどうかを示します。 デフォルト設定は false です。
ServiceOperationColumnNames = { *generate* | *database* }	ReturnType で使用される ComplexType のプロパティ名を指定するときに、メタデータのカラム名を生成するか、データベースの結果セットから取得するかを指定します。 デフォルト設定では生成します。

備考

データベースサーバのサブプロセスとして動作する OData プロデューサを変更します。

権限

MANAGE ODATA システム権限が必要です。

AUTHENTICATION USER 句または ADMIN USER 句を指定する場合は、指定したユーザには VERIFY ODATA システム権限が必要です。

関連する動作

OData プロデューサが実行されている場合、停止して再起動することができます。

例

ユーザ Dave が所有するオブジェクトにアクセスする OData プロデューサを、/odata/dave で始まる URL から作成するとします。また、公開するテーブルの詳細が含まれた dave.txt というモデルファイルを保有しているとします。さらに、OData インタフェースからデータを更新できないこと、要求が最大 10 日間繰り返されることも確認します。次の文を実行して、dave_producer という OData プロデューサを作成します。

```
CREATE ODATA PRODUCER dave_producer
MODEL FROM FILE 'dave.txt'
SERVICE ROOT '/odata/dave'
AUTHENTICATION USER dave
USING 'ReadOnly=true;RepeatRequestForDays=10';
```

この OData プロデューサを一時的に無効にする場合は、次の文を実行します。

```
ALTER ODATA PRODUCER dave_producer NOT ENABLED;
```

読み込み専用制限を削除するが繰返可能要求の制限は維持する場合は、次の文を実行します。

```
ALTER ODATA PRODUCER dave_producer
ENABLED
USING 'ReadOnly=false;RepeatRequestForDays=10';
```

関連情報

CREATE ODATA PRODUCER 文 [865 ページ]
DROP ODATA PRODUCER 文 [1050 ページ]
COMMENT 文 [769 ページ]
SYSODATAPRODUCER システムビュー [1822 ページ]

1.4.4.14 ALTER PROCEDURE 文

プロシージャを変更します。

構文

トリガ定義の変更

```
ALTER PROCEDURE [ owner.]procedure-name procedure-definition
```

```
procedure-definition: CREATE PROCEDURE 文を参照
```

プロシージャ定義の難読化

```
ALTER PROCEDURE [ owner.]procedure-name
SET HIDDEN
```

プロシージャの再コンパイル

```
ALTER PROCEDURE [ owner.]procedure-name
RECOMPILE
```

備考

ALTER PROCEDURE 文には、新しいプロシージャ全体を含めてください。PROC は PROCEDURE の同義語として使用できます。

トリガ定義の変更

ALTER PROCEDURE 文の構文は、最初の 1 語を除き、CREATE PROCEDURE 文の構文とまったく同じです。Watcom-SQL ダイアレクトと Transact-SQL ダイアレクトのプロシージャは、いずれも ALTER PROCEDURE を使用して変更できます。

ALTER PROCEDURE では、プロシージャの既存の権限は変更されません。DROP PROCEDURE に続けて CREATE PROCEDURE を実行した場合は、実行の権限が再割り当てされます。

プロシージャ定義の難読化

SET HIDDEN を使用して、関連するプロシージャの定義を難読化し、解読できないようにします。このプロシージャはアンロードして、他のデータベースに再ロードできます。

SET HIDDEN を使用すると、デバッガを使用したデバッグでも、プロシージャプロファイリングによっても、プロシージャ定義は表示されません。

> **注記**
>
> この変更は、元に戻せません。元のプロシージャ定義をデータベースの外部に保持することをお奨めします。

プロシージャの再コンパイル

RECOMPILE 構文を使用して、ストアドプロシージャを再コンパイルします。ストアドプロシージを再コンパイルすると、カタログに格納された定義が再解析され、構文が検証されます。結果セットを生成するものの RESULT 句を含まないプロシージャの場合は、データベースサーバがプロシージャの結果セットの特性を判別しようとし、カタログに情報を格納します。これは、プロシージャの作成後に、プロシージャの参照先テーブルがカラムの追加、削除、または名前変更するように変更されている場合に役立つことがあります。

プロシージャの定義は、再コンパイルしても変わりません。SET HIDDEN 句で隠された定義を持つプロシージャは再コンパイルできますが、これらの定義は隠されたままになります。

> **注記**
>
> 変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。
>
> - 変数が存在しない
> - 変数の内容が NULL
> - 変数がパラメータで許可されている長さを超えている
> - 変数のデータ型がパラメータで要求されているものと一致していない

権限

そのプロシージャの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- ALTER ANY PROCEDURE システム権限
- ALTER ANY OBJECT システム権限

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

ALTER PROCEDURE は、オプションの ANSI/ISO SQL 言語機能 F381 です。ただし、SQL 標準では、ALTER PROCEDURE を使用してストアドプロシージャ定義を再定義することはできず、Transact-SQL ダイアレクトのプロシージャもサポートされていません。ANSI/ISO SQL 標準では、SET HIDDEN または RECOMPILE はサポートされていません。

例

1. 次の例は、NAMESPACE 句の変数を使用してプロシージャを作成して変更します。
   1. 次の文では、NAMESPACE 句の変数を作成します。

      ```
      CREATE VARIABLE @ns LONG VARCHAR
      SET @ns = 'http://wsdl.domain.com/';
      ```

   2. 次の文は、NAMESPACE 句の変数を使用する FtoC という名前のプロシージャを作成します。

      ```
      CREATE PROCEDURE FtoC ( IN temperature LONG VARCHAR )
      URL 'http://localhost:8082/FtoCService'
      TYPE 'SOAP:DOC'
      NAMESPACE @ns;
      ```

   3. 次の文は、temperature パラメータが FLOAT データ型を受け付けるよう FtoC プロシージャを変更します。

      ```
      ALTER PROCEDURE FtoC ( IN temperature FLOAT )
      URL 'http://localhost:8082/FtoCService'
      NAMESPACE @ns;
      ```

関連情報

CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]

CREATE PROCEDURE 文 [Web サービス] [878 ページ]
ALTER FUNCTION 文 [649 ページ]
DROP PROCEDURE 文 [1051 ページ]

1.4.4.15 ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]

パブリケーションを変更します。Mobile Link では、パブリケーションが SQL Anywhere リモートデータベース内の同期データを識別します。SQL Remote では、統合データベース内とリモートデータベース内の両方のレプリケートされたデータがパブリケーションによって識別されます。

構文

```
ALTER PUBLICATION [ owner.]publication-name alterpub-clause, ...
```

```
alterpub-clause :
ADD article-definition
| ALTER article-definition
| { DELETE | DROP } TABLE [ owner.]table-name
| RENAME publication-name
```

```
article-definition :
TABLE table-name [ ( column-name, ... ) ]
[ WHERE search-condition ]
[ SUBSCRIBE BY expression ]
[ USING ( [PROCEDURE ] [ owner.][procedure-name ]
  FOR UPLOAD { INSERT | DELETE | UPDATE }, ... ) ]
```

備考

この文は、Mobile Link と SQL Remote にのみ適用されます。

パブリケーションでは、"アーティクル" が 1 つのテーブルを表します。パブリケーションの変更とは、アーティクルの追加、修正、削除、またはパブリケーションの名前の変更を意味します。アーティクルを修正する場合は、そのアーティクル全体の定義を入力してください。

パブリケーションを変更する場合は、その直前にパブリケーションの同期を正常に完了させておくことをお奨めします。

FOR DOWNLOAD ONLY または WITH SCRIPTED UPLOAD として定義されているパブリケーションに WHERE 句を使用することはできません。

SUBSCRIBE BY 句は SQL Remote にのみ適用されます。

USING 句はスクリプトを使用するアップロード専用です。

Mobile Link パブリケーションのオプションは、ALTER SYNCHRONIZATION SUBSCRIPTION 文または CREATE SYNCHRONIZATION SUBSCRIPTION 文の ADD OPTION 句で設定します。

Mobile Link パブリケーションを変更する場合、START SYNCHRONIZATION SCHEMA CHANGE 文を実行した後でのみアーティクルを削除できます。

文中で参照されるすべてのテーブルと、変更されるパブリケーション内のすべてのテーブルに対する排他アクセスが必要です。

権限

パブリケーションの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- パブリケーションでの ALTER 権限
- SYS_REPLICATION_ADMIN_ROLE システムロール

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Customers テーブルを pub_contact パブリケーションに追加します。

```
ALTER PUBLICATION pub_contact
   ADD TABLE GROUPO.Customers;
```

関連情報

CREATE PUBLICATION 文 [Mobile Link] [SQL Remote] [899 ページ]
DROP PUBLICATION 文 [Mobile Link] [SQL Remote] [1052 ページ]
ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [701 ページ]
CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [948 ページ]
SYSSYNC システムビュー [1839 ページ]
START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link] [1345 ページ]

1.4.4.16 ALTER REMOTE MESSAGE TYPE 文 [SQL Remote]

特定のメッセージシステム、または作成したメッセージタイプに対する、パブリッシャのメッセージシステム、またはパブリッシャのアドレスを変更します。

構文

```
ALTER REMOTE MESSAGE TYPE message-system
ADDRESS address
```

```
message-system : FILE | FTP | SMTP
```

```
address : string
```

パラメータ

message-system

SQL Remote がサポートするメッセージシステムの 1 つ。次のいずれかの値を指定します。FILE、FTP、または SMTP。

address

指定したメッセージシステムに対して有効なアドレスを含む文字列。

備考

この文は、指定したメッセージタイプでパブリッシャのアドレスを変更します。

Message Agent は、サポートしているいずれかのメッセージリンクを使用して、データベースから出力メッセージを送信します。抽出ユーティリティは、リモートデータベースで GRANT CONSOLIDATE 文を実行するときにこのアドレスを使用します。

アドレスには、指定したメッセージシステムに応じた、パブリッシャのアドレスを指定します。電子メールシステムの場合、アドレス文字列には有効な電子メールアドレスを指定します。ファイル共有システムの場合、アドレス文字列には SQLREMOTE 環境変数で指定されているディレクトリのサブフォルダを指定します。SQLREMOTE 環境変数でディレクトリが指定されていない場合は、現在のディレクトリを指定してください。この設定は、リモートデータベースで GRANT CONSOLIDATE 文を使って無効にできます。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、FILE メッセージリンクのパブリッシャのアドレスを new_addr に変更します。

```
ALTER REMOTE MESSAGE TYPE file
ADDRESS 'new_addr';
```

関連情報

CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote] [903 ページ]
GRANT CONSOLIDATE 文 [SQL Remote] [1141 ページ]

1.4.4.17 ALTER ROLE 文

互換ロールをユーザ定義ロールに移行してから、その互換ロールを削除します。

構文

```
ALTER ROLE compatibility-role-name
MIGRATE TO new-role-name [, new-sa-role-name, new-sso-role-name ]
```

パラメータ

compatibility-role-name

このパラメータを使用して、移行する互換ロールの名前を指定します。

new-role-name

このパラメータを使用して、作成する新しいロールの名前を指定します。

new-sa-role-name

このパラメータを使用して、SYS_AUTH_SA_ROLE ロールの移行先となる新しいロールの名前を指定します。このパラメータは SYS_AUTH_DBA_ROLE を移行するときに必要になります (これにより SYS_AUTH_SA_ROLE が自動的に移行されます)。

new-sso-role-name

このパラメータを使用して、SYS_AUTH_SSO_ROLE ロールの移行先となる新しいロールの名前を指定します。このパラメータは SYS_AUTH_DBA_ROLE を移行するときに必要になります (これにより SYS_AUTH_SSO_ROLE が自動的に移行されます)。

備考

新しいロールの名前は、'SYS_' で始まったり、'_ROLE' で終わったりしてはなりません。たとえば、SYS_MyBackup_ROLE はユーザ定義ロールの名前としては使用できず、MyBackup_ROLE や SYS_MyBackup は使用できます。

ALTER ROLE 文を実行すると、互換ロールの被付与者に新しいロールが付与されます。

CREATE ROLE 文を実行して互換ロール名を指定することにより、移行されて削除された互換ロールをリストアすることができます。たとえば、`CREATE ROLE SYS_AUTH_BACKUP_ROLE;` によって SYS_AUTH_BACKUP_ROLE 互換ロールがリストアされます。

最初は、完全な管理権限 (DBA) を持つユーザのみが新しいロールを管理できますが、CREATE ROLE 文を OR REPLACE 句とともに使用して、追加の管理者を指定することができます。

GRANT 文を使用してロールにシステム権限を付与したり、REVOKE 文を使用してロールからシステム権限を取り消すことができます。

SYS_AUTH_SA_ROLE および SYS_AUTH_SSO_ROLE 互換ロールを移行するには、SYS_AUTH_DBA_ROLE 互換ロールを移行します。これにより、SYS_AUTH_SA_ROLE および SYS_AUTH_SSO_ROLE が自動的に移行されます。SYS_AUTH_DBA_ROLE を移行するときは、`new-sa-role-name` および `new-sso-role-name` パラメータを使用して、移行対象の SYS_AUTH_SA_ROLE ロールと SYS_AUTH_SSO_ROLE ロールに新しい名前を付けます。

権限

移行する互換ロールに対する MANAGE ROLES システム権限および管理権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SYS_AUTH_BACKUP_ROLE ロールに付与されたすべてのユーザとその基盤となるシステム権限を新しいロール custom_Backup_ROLE に移行して、データベースから SYS_AUTH_BACKUP_ROLE を削除します。

```
ALTER ROLE SYS_AUTH_BACKUP_ROLE
MIGRATE TO custom_Backup_ROLE;
```

次の文は、SYS_AUTH_DBA_ROLE 互換ロールに付与されたすべてのユーザ、基盤となるシステム権限、およびロールを、新しいロールである custom_DBA に移行します。これにより、SYS_AUTH_SA_ROLE と SYS_AUTH_SSO_ROLE に付与されたすべてのユーザ、基盤となるシステム権限、およびロールが、それぞれ custom_SA および custom_SSO という新しいロールに自動的に移行されます。最後に、SYS_AUTH_DBA_ROLE、SYS_AUTH_SA_ROLE、SYS_AUTH_SSO_ROLE がデータベースから削除されます。

```
ALTER ROLE SYS_AUTH_DBA_ROLE
MIGRATE TO custom_DBA, custom_SA, custom_SSO;
```

関連情報

CREATE ROLE 文 [905 ページ]
REVOKE ROLE 文 [1281 ページ]

1.4.4.18 ALTER SEQUENCE 文

シーケンスを変更します。

構文

```
ALTER SEQUENCE [owner.]sequence-name
[ RESTART WITH signed-integer ]
[ INCREMENT BY signed-integer ]
[ MINVALUE signed-integer | NO MINVALUE ]
[ MAXVALUE signed-integer | NO MAXVALUE ]
[ CACHE integer | NO CACHE ]
[ CYCLE | NO CYCLE ]
```

パラメータ

RESTART WITH 句

指定したシーケンスを指定した値で再起動します。

INCREMENT BY 句

最後に割り当てられた値から次のシーケンス値までの増分量を定義します。デフォルトは 1 です。負の値を指定すると、降順のシーケンスが生成されます。INCREMENT BY 値が 0 の場合は、エラーが返されます。

MINVALUE 句

シーケンスで生成される最小値を定義します。デフォルトは 1 です。INVALUE が (2^63-1) より大きいか -(2^63-1) より小さい場合は、エラーが返されます。また、MINVALUE が MAXVALUE より大きい場合も、エラーが返されます。

MAXVALUE 句

シーケンスで生成される最大値を定義します。デフォルトは (2^63-1) です。MAXVALUE が (2^63-1) より大きいか -(2^63-1) より小さい場合は、エラーが返されます。

CACHE 句

より速くアクセスできるようにメモリに保持される、事前に割り付けられたシーケンス値の数を指定します。キャッシュが不足すると、シーケンスキャッシュが再移植され、対応するエントリがトランザクションログに書き込まれます。キャッシュの現在の値は、チェックポイントの時点で ISYSSEQUENCE システムテーブルに転送されます。デフォルトは 100 です。

CYCLE 句

最大値または最小値に達した後に、値の生成を継続するかどうかを指定します。

備考

指定したシーケンスが見つからない場合は、エラーメッセージが返されます。

権限

シーケンスの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- ALTER ANY SEQUENCE システム権限
- ALTER ANY OBJECT システム権限

関連する動作

なし

標準

ANSI/ISO SQL 標準

ALTER SEQUENCE 文は、オプションの ANSI/ISO SQL 言語機能 P176 の一部です。CACHE 句は標準にありません。

例

次の例は、シーケンス Test に新しい最大値を設定します。

```
ALTER SEQUENCE Test
   MAXVALUE 1500;
```

関連情報

CREATE SEQUENCE 文 [911 ページ]
DROP SEQUENCE 文 [1060 ページ]

1.4.4.19 ALTER SERVER 文

リモートサーバの属性を変更します。

構文

リモートサーバの変更

```
ALTER [ REMOTE ] SERVER server-name
[ CLASS server-class | variable ]
[ USING connection-string-info | variable ]
[ CAPABILITY cap-name { ON | OFF | VALUE variable } ]
[ READ ONLY [ ON | OFF | VALUE variable ] ]
[ DEFAULT LOGIN string | variable [ IDENTIFIED BY string | variable ]  | NO
DEFAULT LOGIN ]
```

```
server-class-string :
{ 'ADSODBC' | 'ADS_ODBC' }
| 'ASEODBC' | 'ASE_ODBC' }
| 'DB2ODBC' | 'DB2_ODBC' }
| 'HANAODBC' | 'HANA_ODBC' }
| 'IQODBC' | 'IQ_ODBC' }
| 'MIRROR'
| 'MSACCESSODBC' | 'MSACCESS_ODBC' }
| 'MSSODBC' | 'MSS_ODBC' }
| 'MYSQLODBC' | 'MYSQL_ODBC' }
| 'ODBC'
| 'ORAODBC' | 'ORA_ODBC' }
| 'SAODBC' | 'SA_ODBC' }
| 'ULODBC' | 'UL_ODBC' }
```

```
connection-info-string :
  { 'data-source-name' | 'sqlanywhere-connection-string' }
```

ディレクトリアクセスサーバの変更

```
ALTER SERVER server-name
[ CLASS 'DIRECTORY' ]
[ USING using-string | variable ]
```

```
[ CAPABILITY cap-name { ON | OFF | VALUE variable } ]
[ READ ONLY [ ON | OFF | VALUE variable ] ]
[ ALLOW { 'ALL' | 'SPECIFIC' | variable } USERS ]
```

```
using-string :
  'ROOT = path [ ;SUBDIRS = n ] [ ;CREATEDIRS = { YES | NO } ] [ ;DELIMITER = { / | ¥ } ]'
```

リモートサーバの変更 (SAP HANA 構文)

```
ALTER REMOTE SOURCE remote-source-name
    ADAPTER adapter-name | variable
    CONFIGURATION connection-info-string | variable
    [ CAPABILITY cap-name { ON | OFF | VALUE variable } ]
    [ READ ONLY [ ON | OFF | VALUE variable ] ]
    [ WITH CREDENTIAL TYPE { 'PASSWORD' | variable } USING { 'USER=remote-user;password=remote-password' | variable } | WITH NO CREDENTIAL ]
```

パラメータ

ALTER [REMOTE] SERVER

リモートサーバを変更する場合は REMOTE キーワードはオプションで、他のデータベースとの互換性を保つために用意されています。

CLASS 句 サーバのクラスを変更するには、この句を指定します。

USING 句

サーバ接続情報を設定します。

USING 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます ({variable-name})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、'DSN={@mydsn}' を含む USING 句は、@mydsn が SQL 変数であり、リモートデータアクセスサーバへの接続時に @mydsn 変数の現在の内容が置き換えられることを示します。

CAPABILITY 句

サーバの機能の ON と OFF を切り替えるには、この句を指定します。サーバ機能の情報は ISYSCAPABILITY システムテーブルに格納されます。サーバ機能の名前には、SYSCAPABILITYNAME システムビューからアクセスできます。リモートサーバとの最初の接続が確立されるまで、ISYSCAPABILITY システムテーブルと SYSCAPABILITYNAME システムビューにはデータが移植されません。2 回目以降の接続では、ISYSCAPABILITY システムテーブルからデータベースサーバの機能が取得されます。

通常、サーバの機能を変更する必要はありません。クラス ODBC の一般的なサーバの機能を変更しなければならない場合はあります。

READ ONLY 句 (リモートサーバ)

リモートサーバが読み込み専用モードでアクセスされるかどうかを指定します。

READ ONLY 句を指定しなくても、サーバの読み込み専用設定に影響はありません。READ ONLY または READ ONLY ON を指定すると、サーバは読み込み専用に変更されます。READ ONLY OFF を指定すると、サーバは読み込み/書き込みに変更されます。

READ ONLY 句 (ディレクトリアクセスサーバ)

ディレクトリからアクセスするファイルが読み込み専用で修正できないかどうかを指定します。デフォルトでは、READ ONLY は NO に設定されています。

ALLOW USERS 句

外部ログイン (externlogin) を行わなくてもディレクトリアクセスサーバを使用できるかどうかを指定します。ALLOW 'ALL' USERS 句を指定すると、ディレクトリアクセスサーバの作成または変更で、ユーザごとの外部ログインが不要になります。ALLOW 'SPECIFIC' USERS を指定すると、ディレクトリアクセスサーバを使用するユーザごとに外部ログインが必要な場合と同じになります。この句を指定しないと、ALLOW 'SPECIFIC' USERS を指定した場合と同じになります。

DEFAULT LOGIN 句

リモートサーバのデフォルトログインを追加または変更するには、DEFAULT LOGIN を指定します。NO DEFAULT LOGIN を指定すると、リモートサーバサーバのデフォルトログインが削除されます (存在する場合)。

ALTER REMOTE SOURCE (SAP HANA 構文)

パラメータとその値を含むこの構文は、リモートサーバを変更する構文 (CREATE [REMOTE]) とセマンティック上同じで、SAP HANA サーバとの互換性を保つために用意されています。2 つの構文は、次のように 1 対 1 で句が一致します。

ADAPTER adapter-name

ALTER [REMOTE] SERVER 構文については、CLASS 句の説明を参照してください。

CONFIGURATION connection-info-string

ALTER [REMOTE] SERVER 構文については、USING 句の説明を参照してください。

CAPABILITY cap-name

ALTER [REMOTE] SERVER 構文については、CAPABILITY 句の説明を参照してください。

READ ONLY [ON | OFF] 句

ALTER [REMOTE] SERVER 構文 (リモートサーバ) については、READ ONLY 句の説明を参照してください。

WITH CREDENTIAL TYPE 句

ALTER [REMOTE] SERVER 構文については、DEFAULT LOGIN 句の説明を参照してください。WITH CREDENTIAL TYPE 句で変数を使用している場合は、その変数は値 'PASSWORD' を含む文字列にする必要があります。また、USING 変数は 'user=...;password=...' のフォーマットを使用した文字列にする必要があります。

備考

変更は、次回リモートサーバに接続する際に有効になります。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合は、データベースサーバからエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL

- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

SERVER OPERATOR システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ase_prod という名前の Adaptive Server Enterprise サーバの DSN を ase_datasource に変更します。

```
ALTER SERVER ase_prod
    USING 'ase_datasource';
```

次の例は、ase_prod という名前の Adaptive Server Enterprise サーバの DSN を、そのデータソース名を変数 ase_source から取得するよう変更します。

```
ALTER SERVER ase_prod
    USING '{ase_source}';
CREATE VARIABLE ase_source VARCHAR(128);
    SET ase_source = 'ase_datasource';
```

次の例は、サーバ ase_prod の機能を変更します。

```
ALTER SERVER ase_prod
    CAPABILITY 'insert select' OFF;
```

次の例は、ディレクトリアクセスサーバが `c:¥temp` ディレクトリ内の 9 レベルのサブディレクトリを取得するように変更します。

```
ALTER SERVER ase_prod
    CLASS 'DIRECTORY'
    USING 'ROOT=c:¥¥temp;SUBDIRS=9';
```

関連情報

CREATE SERVER 文 [914 ページ]
DROP REMOTE CONNECTION 文 [1055 ページ]
DROP SERVER 文 [1061 ページ]
SYSCAPABILITY システムビュー [1794 ページ]
SYSCAPABILITYNAME システムビュー [1794 ページ]

1.4.4.20 ALTER SERVICE 文 [HTTP Web サービス]

既存の HTTP Web サービスを変更します。

構文

```
ALTER SERVICE service-name
[ TYPE { 'RAW' | 'HTML' | 'JSON' | 'XML' } ]
[ URL [ PATH ] { ON | OFF | ELEMENTS } ]
[ common-attributes ]
[ AS { statement | NULL } ]
```

```
common-attributes :
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| GET
| HEAD
| PUT
| DELETE
| NONE
| *
```

パラメータ

service-name

Web サービス名に使用できるのは、任意の順序の英数字文字、またはスラッシュ (/)、ハイフン (-)、アンダースコア (_)、ピリオド (.)、感嘆符 (!)、チルダ (~)、アスタリスク (*) 、アポストロフィ (')、左カッコ ((()、右カッコ ()) です。ただし、スラッシュ (/) は、サービス名の先頭または最後の文字には使用できず、2 つ以上連続 (// など) して使用することもできません。

サービスルートに名前を付けることができますが、この名前には特別な機能があります。

TYPE 句

各サービスが特定の応答フォーマットを定義する、サービスタイプを識別します。リストされたサービスタイプのいずれかを指定します。デフォルト値はありません。

'RAW'

結果セットはフォーマットされずにクライアントに送られます。このサービスを利用するには、すべてのコンテンツのマークアップが明示的に指定されている必要があります。マークアップが付けられた現在のコンテンツを含む複雑な動的コンテンツである JavaScript およびイメージは、オンデマンドで生成できます。sa_set_http_header プロシージャを使用して Content-Type 応答ヘッダを設定することにより、メディアタイプを指定できます。すべてのブラウザでマークアップを text/plaina ではなく HTML として表示するには、HTML マークアップを生成するときに Content-Type ヘッダを 'text/html' に設定することをお奨めします。

'HTML'

結果セットはテーブルまたはビューの HTML 表現で返されます。

'JSON'

結果セットは JavaScript Object Notation (JSON) で返されます。JSON サービスは、自動的には JSON 入力を処理しません。(応答内の) データを JSON 形式で表すことのみ行います。JSON は、**application/x-www-form-urlencoded** がサポートされている POST/PUT メソッドを受け付けます。POST/PUT メソッドで **Content-Type: application/json** を指定すると、アプリケーションは http_variable('body') を使用して JSON (要求) コンテンツを取得します。データベースサーバは自動的には JSON 入力を解析しません。解析するかどうかはアプリケーションによって決まります。

'XML'

結果セットは XML として返されます。結果セットがすでに XML の場合、それ以上フォーマットは適用されません。XML でない場合は、XML として自動的にフォーマットされます。代替手段としては、sa_set_http_header プロシージャを使用して text/xml などの有効な Content-Type を設定した FOR XML RAW 句を使用して、RAW サービスから select を返す方法もあります。

URL 句

URL パスを受け入れるかどうか、受け入れる場合にはどのように処理するかを決定します。URL PATH を指定すると、URL と同じ機能があります。

OFF

URL 要求でサービス名に続いてパスを指定できないことを示します。OFF はデフォルト設定です。たとえば、パス要素 `/aaa/bbb/ccc` のため、次の形式は許可されません。

```
http://host-name/service-name/aaa/bbb/ccc
```

Web サービスの作成時に `CREATE SERVICE echo URL PATH OFF` が指定されたとします。`http://localhost/echo?id=1` に似た URL で次の値が生成されます。

関数呼び出し	結果
HTTP_VARIABLE('id')	1
HTTP_HEADER('@HttpQueryString')	id=1

ON

URL 要求でサービス名に続いてパスを指定できることを示します。URL という専用の HTTP 変数を問い合わせることにより、パスの値が返されます。URL パラメータを明示的に指定するようにサービスを定義したり、HTTP_VARIABLE 関数を使用して取得したりできます。たとえば、次の形式は許可されます。

```
http://host-name/service-name/aaa/bbb/ccc
```

Web サービスの作成時に `CREATE SERVICE echo URL PATH ON` が指定されたとします。`http://localhost/echo/one/two?id=1` に似た URL で次の値が生成されます。

関数呼び出し	結果
HTTP_VARIABLE('id')	1
HTTP_VARIABLE('URL')	one/two
HTTP_HEADER('@HttpQueryString')	id=1

ELEMENTS

URL 要求でサービス名に続いてパスを指定できることを示します。URL1、URL2 などの単一のパラメータキーワードを指定することにより、セグメントでパスが取得されます。各パラメータは、HTTP_VARIABLE 関数または NEXT_HTTP_VARIABLE 関数を使用して検索できます。これらの繰り返し関数は、可変の数のパス要素を指定できるアプリケーションで使用できます。たとえば、次の形式は許可されます。

```
http://host-name/service-name/aaa/bbb/ccc
```

Web サービスの作成時に `CREATE SERVICE echo URL PATH ELEMENTS` が指定されたとします。`http://localhost/echo/one/two?id=1` に似た URL で次の値が生成されます。

関数呼び出し	結果
HTTP_VARIABLE('id')	1
HTTP_VARIABLE('URL1')	one
HTTP_VARIABLE('URL2')	two
HTTP_VARIABLE('URL3')	NULL
HTTP_HEADER('@HttpQueryString')	id=1

10 個までの要素を取得できます。対応する要素が指定されていない場合は、NULL 値が返されます。上記の例では、対応する要素が指定されていないため、`HTTP_VARIABLE('URL3')` は NULL を返します。

AUTHORIZATION 句

サービスに接続するときに、ユーザが基本的な HTTP 認証を通じてユーザ名とパスワードを指定する必要があるかどうかを決定します。デフォルト値は ON です。AUTHORIZATION が OFF の場合、すべてのサービスに AS 句が必要となり、USER 句でユーザを指定する必要があります。すべての要求は、そのユーザのアカウントと権限を使用して実行されます。AUTHORIZATION が ON の場合、すべてのユーザはユーザ名とパスワードを指定する必要があります。オプションとして、USER 句でユーザ名またはグループ名を指定することにより、サービスを使用できるユーザを制限することもできます。ユーザ名が NULL の場合、既知のユーザはすべてサービスにアクセスできます。AUTHORIZATION 句を使用すると、Web サービスで、データベース認証と権限を通じてデータベース内のデータへのアクセスを制御できます。

authorization の値が ON の場合、Web サービスに接続しようとしている HTTP クライアントは、基本認証 (RFC 2617) を使用します。基本認証では、ユーザとパスワードの情報が base-64 エンコーディングを使用して難読化されます。セキュリティを強化するため、HTTPS プロトコルを使用してください。

ENABLE 句と DISABLE 句

サービスを使用できるかどうかを指定します。デフォルトでは、サービスの作成時に有効にされます。サービスを作成または変更するときに、ENABLE 句または DISABLE 句を含めることができます。サービスを無効にすると、事実上、サービスがオフラインになります。無効にしたサービスは、後で ENABLE 句を含む ALTER SERVICE 文を使用して有効にできます。無効にされたサービスに対して HTTP 要求を行うと、通常、`404 Not Found` HTTP ステータスが返されます。

METHODS 句

サービスでサポートされている HTTP メソッドを指定します。有効な値は、DEFAULT、POST、GET、HEAD、PUT、DELETE、NONE です。アスタリスク (*) は、POST、GET、HEAD メソッドを表す省略形として使用できます。これらのメソッドは、RAW、HTML、XML サービスタイプのデフォルトの要求タイプです。すべての HTTP メソッドがすべてのサービスタイプに有効なわけではありません。次の表は、各サービスタイプに適用できる有効な HTTP メソッドの概要を示します。

メソッド値	適用対象のサービス	説明
DEFAULT	すべて	DEFAULT は、指定されたサービスタイプのデフォルトの HTTP メソッドのセットをリセットする場合に使用します。他のメソッド値が存在するリストに含めることはできません。
POST	RAW、HTML、JSON、XML	デフォルトでは有効になっています。
GET	RAW、HTML、JSON、XML	デフォルトでは有効になっています。
HEAD	RAW、HTML、JSON、XML	デフォルトでは有効になっています。
PUT	RAW、HTML、JSON、XML	デフォルトでは有効になっていません。
DELETE	RAW、HTML、JSON、XML	デフォルトでは有効になっていません。
NONE	すべて	NONE は、サービスへのアクセスを無効にするために使用します。
*	RAW、HTML、JSON、XML	'POST,GET,HEAD' と指定することと同じです。

たとえば、次のいずれかの句を使用して、サービスですべての HTTP メソッドタイプがサポートされていることを指定できます。

```
METHODS '*,PUT,DELETE'
METHODS 'POST,GET,HEAD,PUT,DELETE'
```

サービスタイプに対する要求タイプのリストをデフォルトにリセットするには、次の句を使用できます。

```
METHODS 'DEFAULT'
```

SECURE 句

セキュアリスナまたは非セキュアリスナで、サービスにアクセスできるかどうかを指定します。ON は、HTTPS 接続のみが受け入れられ、HTTP ポートが受信する接続は HTTPS ポートに自動的にリダイレクトされることを示します。OFF は、Web サーバの起動時に必要なポートが指定されていると、HTTP 接続と HTTPS 接続の両方が受け入れられることを示します。デフォルト値は OFF です。

USER 句

Web サービス要求の実行権限を持つデータベースユーザまたはユーザグループを指定します。サービスが AUTHORIZATION OFF で設定されている場合は USER 句を必ず指定し、AUTHORIZATION ON (デフォルト) で設定されている場合は、ユーザを制限する場合に指定します。認証を必要とするサービスに対して HTTP 要求を行うと、`401 Authorization Required` HTTP 応答ステータスになります。この応答に基づいて、ユーザ ID とパスワードを要求するプロンプトが Web ブラウザで表示されます。

> ⚠ 警告
>
> 認証が有効になっている (AUTHORIZATION ON: デフォルト) 場合には、USER 句を指定することを強くお奨めします。指定しないと、すべてのユーザが認証されます。

USER 句は、サービス要求を処理するのに使用できるデータベースユーザアカウントを制御します。データベースのアクセスパーミッションは、サービスのユーザに割り当てられた権限に制限されます。

statement

サービスにアクセスしたときに起動する、ストアドプロシージャ呼び出しなどのコマンドを指定します。

`statement` なしの DISH 以外のサービスに対する HTTP 要求により、URL 内で実行する SQL 式が指定されます。認証が必要ですが、この機能はサーバを SQL の挿入に対して脆弱にするため、実稼動システムでは使用しないでください。サービス内で文が定義されると、指定された SQL 文はサービスによって実行できる文のみとなります。

通常の Web サービスアプリケーションでは、`statement` を使用して関数またはプロシージャを呼び出します。ホスト変数をパラメータとして渡して、クライアント提供の HTTP 変数にアクセスできます。

次の `statement` は、AuthenticateUser というプロシージャに 2 つのホスト変数を渡すプロシージャコールを示しています。この呼び出しは Web クライアントが user_name および user_password 変数を供給することを仮定しています。

```
CALL AuthenticateUser ( :user_name, :user_password );
```

備考

ALTER SERVICE 文は、Web サービスの属性を変更します。

権限

そのサービスの所有者であるか、または MANAGE ANY WEB SERVICE システム権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ALTER SERVICE 文を使用して既存の Web サービスを無効にする方法を示します。

```
CREATE SERVICE WebServiceTable
   TYPE 'RAW'
   AUTHORIZATION OFF
   USER DBA
   AS SELECT *
      FROM SYS.SYSTAB;
ALTER SERVICE WebServiceTable DISABLE;
```

関連情報

CREATE SERVICE 文 [HTTP Web サービス] [921 ページ]
DROP SERVICE 文 [1063 ページ]
sa_parse_json システムプロシージャ [1701 ページ]
SYSWEBSERVICE システムビュー [1862 ページ]
sa_set_http_header システムプロシージャ [1619 ページ]
識別子 [6 ページ]
ROW コンストラクタ [複合] [510 ページ]
ARRAY コンストラクタ [複合] [234 ページ]

1.4.4.21 ALTER SERVICE 文 [SOAP Web サービス]

既存の HTTP を介した SOAP または DISH サービスを変更します。

構文

HTTP を介した SOAP サービス

```
ALTER SERVICE service-name
[ TYPE 'SOAP' ]
[ DATATYPE { ON | OFF | IN | OUT } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]
[ AS statement ]
```

```
common-attributes :
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
```

```
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| HEAD
| NONE
```

DISH サービス

```
ALTER SERVICE service-name
[ TYPE 'DISH' ]
[ GROUP { group-name | NULL } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ]| 'XML' | NULL } ]
[ common-attributes ]
```

```
common-attributes :
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| GET
| HEAD
| NONE
| *
```

パラメータ

ALTER SERVICE の句の説明は、CREATE SERVICE 文の場合と同じです。

パラメータ

service-name

Web サービス名に使用できるのは、任意の順序の英数字文字、またはスラッシュ (/)、ハイフン (-)、アンダースコア (_)、ピリオド (.)、感嘆符 (!)、チルダ (~)、アスタリスク (*) 、アポストロフィ (')、左括弧 (()、右括弧 ()) です。ただし、スラッシュ (/) は、サービス名の先頭または最後の文字には使用できず、2 つ以上連続 (// など) して使用することもできません。

他のサービスとは異なり、スラッシュ (/) は、DISH サービス名のどこにも使用できません。

サービスルートに名前を付けることができますが、この名前には特別な機能があります。

TYPE clause

各サービスが特定の応答フォーマットを定義する、サービスタイプを識別します。リストされたサービスタイプのいずれかを指定します。デフォルト値はありません。

'SOAP'

結果セットは、SOAP エンベロープと呼ばれる XML ペイロードとして返されます。FORMAT 句を使用すると、データのフォーマットをさらに絞り込むことができます。SOAP サービスへの要求は、単なる汎用 HTTP 要求ではなく有効な SOAP 要求である必要があります。SOAP 標準の詳細については、Simple Object Access Protocol (SOAP) を参照してください。

'DISH'

DISH サービス (SOAP ハンドラを決定) は、GROUP コンテキスト内で SOAP サービスを参照する SOAP 終了ポイントです。また、SOAP クライアントツールキットで使用するために WSDL (Web Services Description Language) を生成することにより、SOAP サービスに対してインタフェースも公開します。

GROUP clause

GROUP 句が指定されていない DISH サービスは、データベース内で定義されているすべての SOAP サービスを公開します。慣例により、SOAP サービス名は、GROUP 要素と NAME 要素で構成できます。最後のスラッシュ文字によって、名前がグループから区切られます。たとえば、'aaa/bbb/ccc' と定義されている SOAP サービス名は 'ccc' であり、グループは 'aaa/bbb' です。この方法で DISH サービスを区切るのは、無効です。代わりに、SOAP 終了ポイントとなる SOAP サービスのグループを指定するために、GROUP 句が適用されます。

> **注記**
>
> 有効な XML を生成するため、WSDL ではスラッシュがアンダースコアに変換されます。GROUP 句が指定されていない DISH サービスでは、スラッシュを含む可能性のあるすべての SOAP サービスが公開されるため、このような DISH サービスを使用する場合には注意してください。アンダースコアを含む SOAP サービス名とグループを一緒に使用する場合は、あいまいにならないように注意してください。

DATATYPE clause

SOAP サービスにだけ適用されます。DATATYPE OFF が指定されていると、SOAP の入力パラメータと応答データは XMLSchema 文字列型として定義されます。ほとんどの場合、実際のデータ型を使用すると、SOAP クライアントで計算を行う前にデータをキャストする必要がなくなるため、この方法をおすすめします。パラメータのデータ型は、DISH サービスが生成する WSDL のスキーマセクションで公開されます。出力データ型は、データのカラムごとに XML スキーマ型の属性として表されます。

DATATYPE 句に指定できる値を次に示します。

ON

入力パラメータと結果セットの応答のデータ型指定を生成します。

OFF

すべての入力パラメータと応答のデータ型は、XMLSchema 文字列 (デフォルト) になります。

IN

入力パラメータに対してのみ実際のデータ型を生成します。応答のデータ型は XMLSchema 文字列になります。

OUT

応答に対してのみ実際のデータ型を生成します。入力パラメータは XMLSchema 文字列になります。

FORMAT clause

この句は、SOAP クライアントアプリケーションに応答を送信するときの出力フォーマットを指定します。

SOAP サービスフォーマットが SOAP サービスで指定されていない場合は、関連する DISH サービスフォーマットによって指定されます。デフォルトのフォーマットは DNET です。

SOAP サービスのグループにフォーマットルールを活用するには、SOAP 要求を DISH サービス (SQL Anywhere SOAP 終了ポイント) に指示してください (SOAP 操作)。SOAP サービスの FORMAT 指定は、DISH サービスの指定よりも優先されます。DISH サービスのフォーマット指定は、SOAP サービスで FORMAT 句が定義されていない場合に使用されます。どちらのサービスにも FORMAT が指定されていない場合のデフォルトは 'DNET' です。

次のフォーマットがサポートされます。

'DNET'

.NET クライアントアプリケーションで使用するために、互換性のある System.Data.DataSet フォーマットで出力されます (デフォルト)。

'CONCRETE'

この出力フォーマットは、ローとカラムのオブジェクトの配列を表すインタフェースを生成できるが、DNET フォーマットを使用できないクライアント SOAP ツールキットをサポートするために使用されます。Java クライアントおよび .NET クライアントは、この出力フォーマットを簡単に使用できます。

特定のフォーマットは、明示的なデータセットオブジェクトまたは SimpleDataset として WSDL 内で公開されます。どちらのデータセット表現も、各ローにカラムの配列が含まれているローの配列を表すデータ構造を示します。明示的なデータセットオブジェクトは、ロー内の各カラムに対応するパラメータ名とデータ型を提供することにより、結果セットの実際の形式を表す利点があります。一方、SimpleDataset はあらゆるタイプの無数のカラムが含まれているローを公開します。

FORMAT 'CONCRETE' EXPLICIT ON では、RESULT 句を定義するストアドプロシージャを Service 文で呼び出す必要があります。この条件が満たされる場合、SOAP サービスは名前がサービス名で始まり Dataset を付加した明示的なデータセットを公開します。

この条件が満たされない場合、SimpleDataset が使用されます。

'XML'

出力は XMLSchema 文字列フォーマットで生成されます。応答としての XML ドキュメントでは、カラムデータを抽出するために SOAP クライアントでさらに処理が必要です。このフォーマットは、ローおよびカラムの配列を表す中間インタフェースオブジェクトを生成できない SOAP クライアントに適しています。

NULL

NULL タイプを指定すると、SOAP サービスまたは DISH サービスでデフォルトの動作が使用されます。ALTER SERVICE 文で NULL タイプを使用すると、既存のサービスのフォーマットタイプが上書きされます。

AUTHORIZATION clause

サービスに接続するときに、ユーザが基本的な HTTP 認証を通じてユーザ名とパスワードを指定する必要があるかどうかを決定します。デフォルト値は ON です。AUTHORIZATION が OFF の場合、SOAP サービスに AS 句が必要となり、USER 句でユーザを指定する必要があります。すべての要求は、そのユーザのアカウントと権限を使用して実行されます。AUTHORIZATION が ON の場合、すべてのユーザはユーザ名とパスワードを指定する必要があります。オプションとして、USER 句でユーザ名またはグループ名を指定することにより、サービスを使用できるユーザを制限することもできます。ユーザ名が NULL の場合、既知のユーザはすべてサービスにアクセスできます。AUTHORIZATION 句を使用すると、Web サービスで、データベース認証と権限を通じてデータベース内のデータへのアクセスを制御できます。

authorization の値が ON の場合、Web サービスに接続しようとしている HTTP クライアントは、基本認証 (RFC 2617) を使用します。基本認証では、ユーザとパスワードの情報が base-64 エンコーディングを使用して難読化されます。セキュリティを強化するため、HTTPS プロトコルを使用することをおすすめします。

ENABLE and DISABLE clauses

サービスを使用できるかどうかを指定します。デフォルトでは、サービスの作成時に有効にされます。サービスを作成または変更するときに、ENABLE 句または DISABLE 句を含めることができます。サービスを無効にすると、事実上、サービスがオフラインになります。無効にしたサービスは、後で ENABLE 句を含む ALTER SERVICE 文を使用して有効にできます。無効にされたサービスに対して HTTP 要求を行うと、通常、`404 Not Found` HTTP ステータスが返されます。

METHODS clause

サービスでサポートされている HTTP メソッドを指定します。有効な値は、DEFAULT、POST、GET、HEAD、NONE です。POST、GET、HEAD メソッドを表す省略形として、アスタリスク (*) を使用できます。SOAP サービスのデフォルトのメソッドタイプは、POST と HEAD です。DISH サービスのデフォルトのメソッドタイプは、GET、POST、および HEAD です。すべての HTTP メソッドがすべてのサービスタイプに有効なわけではありません。次の表は、各サービスタイプに適用できる有効な HTTP メソッドの概要を示します。

メソッド値	適用対象のサービス	説明
DEFAULT	両方	DEFAULT は、指定されたサービスタイプのデフォルトの HTTP メソッドのセットをリセットする場合に使用します。他のメソッド値が存在するリストに含めることはできません。
POST	両方	SOAP ではデフォルトで有効になります。
GET	DISH のみ	DISH ではデフォルトで有効になります。
HEAD	両方	SOAP と DISH でデフォルトで有効になります。
NONE	両方	NONE は、サービスへのアクセスを無効にするために使用します。SOAP サービスに適用すると、SOAP 要求がそのサービスに直接アクセスできなくなります。このため、DISH サービスの SOAP 終了ポイント経由で SOAP 操作にアクセスします。 各 SOAP サービスに **`METHODS 'NONE'`** を指定することをおすすめします。
*	DISH のみ	**`'POST,GET,HEAD'`** と指定することと同じです。

たとえば、次のいずれかの句を使用して、サービスですべての HTTP を介した SOAP メソッドタイプがサポートされていることを指定できます。

```
METHODS 'POST,HEAD'
```

サービスタイプに対する要求タイプのリストをデフォルトにリセットするには、次の句を使用できます。

```
METHODS 'DEFAULT'
```

SECURE clause

セキュアリスナまたは非セキュアリスナで、サービスにアクセスできるかどうかを指定します。ON は、HTTPS 接続のみが受け入れられ、HTTP ポートが受信する接続は HTTPS ポートに自動的にリダイレクトされることを示します。OFF は、Web サーバの起動時に必要なポートが指定されていると、HTTP 接続と HTTPS 接続の両方が受け入れられることを示します。デフォルト値は OFF です。

USER clause

Web サービス要求の実行権限を持つデータベースユーザまたはユーザグループを指定します。サービスが AUTHORIZATION OFF で設定されている場合は USER 句を必ず指定し、AUTHORIZATION ON (デフォルト) で設定されている場合は、ユーザを制限する場合に指定します。認証を必要とするサービスに対して HTTP 要求を行うと、`401 Authorization Required` HTTP 応答ステータスになります。この応答に基づいて、ユーザ ID とパスワードを要求するプロンプトが Web ブラウザで表示されます。

> ⚠ 警告
>
> 認証が有効になっている (AUTHORIZATION ON: デフォルト) 場合には、USER 句を指定することを強くおすすめします。指定しないと、すべてのユーザが認証されます。

USER 句は、サービス要求を処理するのに使用できるデータベースユーザアカウントを制御します。データベースのアクセスパーミッションは、サービスのユーザに割り当てられた権限に制限されます。

statement

サービスにアクセスしたときに起動する、ストアドプロシージャ呼び出しなどのコマンドを指定します。

DISH サービスは、NULL 文または文のいずれかを指定する必要がある唯一のサービスです。SOAP サービスで文を定義する必要があります。他のサービスが NULL 文を持つ場合もありますが、AUTHORIZATION ON で設定した場合だけです。

`statement` なしの DISH 以外のサービスに対する HTTP 要求により、URL 内で実行する SQL 式が指定されます。認証が必要ですが、この機能はサーバを SQL の挿入に対して脆弱にするため、本稼働システムでは使用しないでください。サービス内で文が定義されると、指定された SQL 文はサービスによって実行できる文のみとなります。

通常の Web サービスアプリケーションでは、`statement` を使用して関数またはプロシージャを呼び出します。ホスト変数をパラメータとして渡して、クライアント提供の HTTP 変数にアクセスできます。

次の `statement` は、AuthenticateUser という名前のプロシージャに 2 つのホスト変数を渡すプロシージャコールを示しています。この呼び出しは Web クライアントが user_name および user_password 変数を供給することを仮定しています。

```
CALL AuthenticateUser ( :user_name, :user_password );
```

備考

ALTER SERVICE 文は、Web サービスの属性を変更します。

権限

そのサービスの所有者であるか、または MANAGE ANY WEB SERVICE システム権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ALTER SERVICE 文を使用して既存の Web サービスを無効にする方法を示します。

```
CREATE SERVICE WebServiceTable
   TYPE 'SOAP'
   AUTHORIZATION OFF
   USER DBA
   AS SELECT *
      FROM SYS.SYSTAB;
ALTER SERVICE WebServiceTable DISABLE;
```

関連情報

CREATE SERVICE 文 [SOAP Web サービス] [927 ページ]
DROP SERVICE 文 [1063 ページ]
SYSWEBSERVICE システムビュー [1862 ページ]

1.4.4.22 ALTER SPATIAL REFERENCE SYSTEM 文

既存の空間参照系の設定を変更します。空間参照系を変更する前に、注意事項について「備考」の項を参照してください。

構文

```
ALTER SPATIAL REFERENCE SYSTEM
srs-name
[ srs-attribute [ srs-attribute ... ] ]
```

```
srs-name : string
```

```
srs-attribute :
SRID srs-id
| DEFINITION { definition-string | NULL }
| ORGANIZATION { organization-name IDENTIFIED BY organization-srs-id | NULL }
```

```
| TRANSFORM DEFINITION { transform-definition-string | NULL }
| LINEAR UNIT OF MEASURE linear-unit-name
| ANGULAR UNIT OF MEASURE { angular-unit-name | NULL }
| TYPE { ROUND EARTH | PLANAR }
| COORDINATE coordinate-name { UNBOUNDED | BETWEEN low-number AND high-
number }
| ELLIPSOID SEMI MAJOR AXIS semi-major-axis-length { SEMI MINOR AXIS semi-minor-axis-
length | INVERSE FLATTENING inverse-flattening-ratio }
| SNAP TO GRID { grid-size | DEFAULT }
| TOLERANCE { tolerance-distance | DEFAULT }
| AXIS ORDER axis-order
| POLYGON FORMAT polygon-format
| STORAGE FORMAT storage-format
```

```
srs-id :integer
```

```
semi-major-axis-length :number
```

```
semi-minor-axis-length :number
```

```
inverse-flattening-ratio :number
```

```
grid-size :DOUBLE: 通常 0 ～ 1
```

```
tolerance-distance :number
```

```
axis-order :  { 'x/y/z/m' | 'long/lat/z/m' | 'lat/long/z/m' }
```

```
polygon-format : { 'CounterClockWise' | 'Clockwise' | 'EvenOdd' }
```

```
storage-format : { 'Internal' | 'Original' | 'Mixed' }
```

パラメータ

各句の詳細な定義については、CREATE SPATIAL REFERENCE SYSTEM 文を参照してください。

IDENTIFIED BY 句

この句は、空間参照系の SRID 番号を変更するときに使用します。

DEFINITION 句

この句は、座標系のデフォルトを設定または上書きするときに使用します。

ORGANIZATION 句

この句は、新しい空間参照系の基となる空間参照系を作成した組織に関する情報を指定するときに使用します。

TRANSFORM DEFINITION 句

この句は、空間参照系に使用する変換の説明を指定するときに使用します。現時点では、PROJ.4 変換のみがサポートされています。

変換定義は、空間参照系間でデータを変換するときに ST_Transform メソッドで使用されます。一部の変換には、`transform-definition-string` が定義されていなくてもかまいません。

COORDINATE 句

この句は、空間参照系の次元に境界を指定するときに使用します。coordinate-name は、空間参照系で使用される座標系の名前です。非地理的座標系の場合は、coordinate-name に x、y、または m を使用できます。地理的座標系の場合は、coordinate-name に LATITUDE、LONGITUDE、z、または m を使用できます。

LINEAR UNIT OF MEASURE 句

この句は、空間参照系の線形測定単位を指定するときに使用します。指定する値は、ST_UNITS_OF_MEASURE 統合ビューで定義された線形測定単位と一致している必要があります。

ANGULAR UNIT OF MEASURE 句

この句は、空間参照系の角度測定単位を指定するときに使用します。指定する値は、ST_UNITS_OF_MEASURE 統合ビューで定義された角度測定単位と一致している必要があります。

TYPE 句

TYPE 句は、空間参照系で点と点を結ぶ線を解釈する方法を制御するために使用します。地理的空間参照系では、TYPE 句で ROUND EARTH (デフォルト) または PLANAR のいずれかを指定できます。非地理的空間参照系の場合は、タイプに PLANAR を指定します。

ELLIPSOID 句

ELLIPSOID 句は、タイプ ROUND EARTH の空間参照系の楕円として、地球を表現するときに使用する値を指定するために使用します。DEFINITION 句が指定されている場合は、この句で楕円定義を指定できます。ELLIPSOID 句が指定されていると、このデフォルトの楕円が上書きされます。

SNAP TO GRID 句

平らな地球 (平面) の空間参照系では、SNAP TO GRID 句を使用して、データベースサーバで計算に使用されるグリッドサイズを定義します。SNAP TO GRID DEFAULT を指定すると、グリッドサイズがデータベースサーバで使用されるデフォルトに設定されます。

曲面の空間参照系では、SNAP TO GRID を 0 に設定してください。

TOLERANCE 句

平らな地球 (平面) の空間参照系では、TOLERANCE 句を使用して、点を比較するときに使用する精度を指定します。

曲面の空間参照系では、TOLERANCE を 0 に設定してください。

POLYGON FORMAT 句

POLYGON FORMAT 句は、多角形の解釈を変更するために使用します。次の値がサポートされます。

- 'CounterClockwise'
- 'Clockwise'
- 'EvenOdd'

デフォルトの多角形フォーマットは 'EvenOdd' です。

STORAGE FORMAT 句

STORAGE FORMAT 句は、空間データをデータベースにロードしたときに格納される内容を制御するために使用します。考えられる値は、次のとおりです。

'Internal'

データベースサーバは正規化された表現のみを格納します。元の入力特性を再現する必要がない場合に指定します。これは、平面 (TYPE PLANAR) の空間参照系のデフォルトです。

> **注記**
>
> Mobile Link を使用して空間データを同期する場合は、代わりに **Mixed** を指定してください。Mobile Link では同期時に等価性がテストされるため、データが元のフォーマットである必要があります。

'Original'

データベースサーバは元の表現のみを格納します。元の入力特性を再現できますが、格納された値のすべての操作に対して正規化の手順を繰り返す必要があり、データの操作が遅くなる可能性があります。

'Mixed'

データベースサーバで内部バージョンが格納されます。内部バージョンが元のバージョンと異なる場合は、元のバージョンも格納されます。両方のバージョンを格納すると、元の表現特性を再現でき、格納された値の操作に対して正規化の手順を繰り返す必要はありません。ただし、ジオメトリごとに 2 つの表現が格納される可能性があるため、記憶領域の要件が大幅に増加します。

Mixed は、曲面の空間参照系 (TYPE ROUND EARTH) のデフォルトフォーマットです。

備考

空間参照系で参照している既存のデータがある場合、空間参照系を変更することはできません。たとえば、ST_Point(SRID=8743) として宣言したカラムがある場合、SRID 8743 を持つ空間参照系は変更できません。これは、格納フォーマットなどの空間参照系の多くの属性が、データの格納フォーマットに影響を及ぼすためです。SRID を参照するデータがある場合は、新しい空間参照系を作成して、データを新しい SRID に変換します。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

空間参照系の所有者であるか、または次のいずれかの権限を持っていることが必要です。

- 空間参照系に対する ALTER 権限
- MANAGE ANY SPATIAL OBJECT システム権限
- ALTER ANY OBJECT システム権限

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、mySpatialRef という架空の空間参照系の多角形フォーマットを EvenOdd に変更します。

```
ALTER SPATIAL REFERENCE SYSTEM mySpatialRef
POLYGON FORMAT 'EvenOdd';
```

関連情報

CREATE SPATIAL REFERENCE SYSTEM 文 [933 ページ]
DROP SPATIAL REFERENCE SYSTEM 文 [1064 ページ]
ST_UNITS_OF_MEASURE 統合ビュー [1869 ページ]

1.4.4.23 ALTER STATISTICS 文

テーブルの 1 つのカラムまたは複数のカラムに関する統計情報を自動的に更新するかどうかを制御します。

構文

```
ALTER STATISTICS
[ ON ] table [ ( column1 [ , column2 ... ] ) ]
AUTO UPDATE { ENABLE | DISABLE }
```

パラメータ

ON

ワード ON はオプションです。ON を指定しても、文の実行には影響がありません。

AUTO UPDATE 句

カラムの統計情報の自動更新を有効にするか無効にするかを指定します。

備考

クエリ、DML 文、LOAD TABLE 文の通常実行時に、データベースサーバはオプティマイザが使用するカラムの統計情報を自動的に維持します。一部のカラムでは、統計情報を維持するときに、生成に必要なオーバヘッドに見合う利点がない場合もあります。たとえば、カラムのクエリ頻度が低い場合、または定期的に大規模な変更があっても最終的にロールバックする場合、継続的に統計情報を更新する意味はほとんどありません。ALTER STATISTICS 文は、このようなカラムの統計情報の自動更新を抑制するときに使用します。

自動更新が無効の場合でも、CREATE STATISTICS 文と DROP STATISTICS 文を使用して、カラムの統計情報を更新できます。パフォーマンスが改善されると判断した場合にのみ、更新してください。通常、カラムの統計情報は無効にしません。

権限

テーブル所有者であるか、または次のいずれかの権限を持っていることが必要です。

- MANAGE ANY STATISTICS システム権限
- ALTER ANY OBJECT システム権限

関連する動作

自動更新を無効にすると、統計情報は古くなります。改めて有効にしてもすぐに最新状態には更新されません。必要に応じて統計情報を再作成するには、CREATE STATISTICS 文を実行します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Customers テーブルの Street カラムに関する統計情報の自動更新を無効にします。

```
ALTER STATISTICS GROUPO.Customers ( Street ) AUTO UPDATE DISABLE;
```

関連情報

CREATE STATISTICS 文 [942 ページ]
DROP STATISTICS 文 [1067 ページ]

1.4.4.24 ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link]

SQL Anywhere 同期プロファイルを変更します。同期プロファイルは、同期の制御に使用できる同期オプションの名前付きコレクションです。

構文

```
ALTER SYNCHRONIZATION PROFILE name
MERGE string
```

パラメータ

name

変更する同期プロファイルの名前。

MERGE 句

この句を使用して、既存のオプションを変更したり、同期プロファイルに新しいオプションを追加したりします。

string

1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列。たとえば、`'option1=value1;option2=value2'` のように記述します。

備考

同期プロファイルによって、SQL Anywhere データベースが Mobile Link サーバと同期する方法を定義します。

ALTER SYNCHRONIZATION PROFILE 文に MERGE が使用されている場合は、同期プロファイルにすでに存在するオプションに文字列に指定されたオプションが追加されます。プロファイルに文字列のオプションがすでに存在する場合は、プロファイルにすでに格納済みの値が文字列の値に置き換わります。

たとえば、次の文を実行すると、プロファイル `myProfile` の値は `subscription=s2;verbosity=high;uploadonly=on` になります。

```
CREATE SYNCHRONIZATION PROFILE myProfile 'subscription=p1;verbosity=high';
ALTER SYNCHRONIZATION PROFILE myProfile MERGE 'subscription=p2;uploadonly=on';
```

拡張オプションの設定には、次の構文を使用します。

```
ALTER SYNCHRONIZATION PROFILE myprofile MERGE
's=mysub;e={ctp=tcpip;adr=''host=localhost;port=2439''}';
```

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link] [946 ページ]
DROP SYNCHRONIZATION PROFILE 文 [Mobile Link] [1070 ページ]

1.4.4.25 ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]

SQL Anywhere リモートデータベースで同期サブスクリプションのプロパティを変更します。

構文

```
ALTER SYNCHRONIZATION SUBSCRIPTION
{ subscription-name | TO publication-name [ FOR ml-username, ...  ] } { alter-
clause ... }
```

```
alter-clause :
RENAME new-subscription-name
| TYPE network-protocol
| ADDRESS protocol-options
| ADD OPTION option=value, ...
| ALTER OPTION option=value, ...
| DELETE { ALL OPTION | OPTION option, ... }
| SET SCRIPT VERSION=script-version
```

```
subscription-name : identifier
```

```
publication-name : identifier

ml-username : identifier

new-subscription-name : identifier

network-protocol : http | https | tls | tcpip | NULL

protocol-options : string | NULL

value : string | integer

option : identifier

script-version : string | NULL
```

パラメータ

TO 句

この句は、パブリケーション名を指定します。

FOR 句を指定しないで TO 句を使用した場合、RENAME 句または SET SCRIPT VERSION 句は使用できません。

FOR 句

この句は、1 つ以上の Mobile Link ユーザ名を指定します。

FOR 句を省略すると、パブリケーションに対するプロトコルタイプ、プロトコルオプション、拡張オプションが設定されます。

FOR 句を指定しないで TO 句を使用した場合、RENAME 句または SET SCRIPT VERSION 句は使用できません。

RENAME 句

この句は、サブスクリプションの新しい名前を指定します。

FOR 句を指定しないで TO 句を使用した場合、RENAME 句は使用できません。

TYPE 句

同期に使用するネットワークプロトコルを指定します。デフォルトのプロトコルは tcpip です。

ADDRESS 句

Mobile Link サーバのロケーションを含むネットワークプロトコルオプションを指定します。

ADD OPTION、ALTER OPTION、DELETE OPTION、DELETE ALL OPTION 句

拡張オプションの追加、修正、削除、すべての削除ができます。それぞれの句に、オプションは 1 つのみ指定できます。Delete All にオプションは指定しません。

各オプションの値に、" = "、" , "、" ; " の記号は使用できません。

SET SCRIPT VERSION 句

この句は、同期中に使用するスクリプトバージョンを指定します。スキーマを変更しないでスクリプトバージョンを変更できます。

FOR 句を指定しないで TO 句を使用した場合、SET SCRIPT VERSION 句は使用できません。

備考

nnetwork-protocol、protocol-options、および options は、複数の個所で設定できます。

この文を使用すると、オプションや他の情報が SQL Anywhere の ISYSSYNC システムテーブルに格納されます。ユーザが持っている権限によっては、パスワードや暗号化の証明書などの情報を表示できます。このとき考えられるセキュリティ上の問題を回避するために、dbmlsync コマンドラインに関する情報を指定できます。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、販売サブスクリプションの Mobile Link サーバのアドレスを変更します。

```
ALTER SYNCHRONIZATION SUBSCRIPTION sales
TYPE TCPIP
ADDRESS 'host=10.11.12.132;port=2439';
```

関連情報

CREATE PUBLICATION 文 [Mobile Link] [SQL Remote] [899 ページ]
DROP PUBLICATION 文 [Mobile Link] [SQL Remote] [1052 ページ]

CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [948 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.26 ALTER SYNCHRONIZATION USER 文 [Mobile Link]

SQL Anywhere リモートデータベースで Mobile Link ユーザのプロパティを変更します。

構文

```
ALTER SYNCHRONIZATION USER ml-username
[ TYPE network-protocol ]
[ ADDRESS protocol-options ]
[ ADD OPTION option=value, ... ]
[ ALTER OPTION option=value, ... ]
[ DELETE { ALL OPTION | OPTION option } ]
```

```
ml-username : identifier
```

```
network-protocol : http | https | tls | tcpip | NULL
```

```
protocol-options : string | NULL
```

```
value : string | integer
```

パラメータ

TYPE 句

同期に使用するネットワークプロトコルを指定します。デフォルトのプロトコルは tcpip です。

ADDRESS 句

この句は、protocol-options を keyword=value の形式でセミコロンで区切って指定します。どのような設定を指定するかは、使用する通信プロトコル (TCP/IP、TLS、HTTP、HTTPS) に応じて異なります。

ADD OPTION、ALTER OPTION、DELETE OPTION、DELETE ALL OPTION 句

拡張オプションの追加、修正、削除、すべての削除ができます。それぞれの句に、オプションは 1 つしか指定できません。Delete All にオプションは指定しません。

備考

nnetwork-protocol、protocol-options、および options は、複数の個所で設定できます。

この文を使用すると、オプションや他の情報が ISYSSYNC システムテーブルに格納されます。ユーザが持っている権限によっては、パスワードや暗号化の証明書を表示できます。このとき考えられるセキュリティ上の問題を回避するために、dbmlsync コマンドラインに関する情報を指定できます。

パブリケーションで参照されるすべてのテーブルへの排他アクセスが必要です。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE SYNCHRONIZATION USER 文 [Mobile Link] [950 ページ]
DROP SYNCHRONIZATION USER 文 [Mobile Link] [1073 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.27 ALTER TABLE 文

テーブルの定義を変更したり、従属ビューを無効にしたりします。

構文

既存のテーブルの変更

```
ALTER TABLE [owner.]table-name { alter-clause, ... }
```

```
alter-clause :
ADD create-clause
| ALTER column-name column-alteration
```

```
| ALTER [ CONSTRAINT constraint-name ] CHECK ( condition )
| DROP drop-object
| RENAME rename-object
| table-alteration
```

```
create-clause :
column-name [ AS ] column-data-type [ new-column-attribute ... ]
| table-constraint
| PCTFREE integer
```

```
column-alteration :
 { column-data-type | alterable-column-attribute } [ alterable-column-
attribute ... ]
| SET COMPUTE ( compute-expression )
| [ SET ] DEFAULT default-value
| ADD [ CONSTRAINT constraint-name ] CHECK ( condition )
| DROP { DEFAULT | COMPUTE | CHECK | CONSTRAINT constraint-name }
```

```
drop-object :
column-name
| CHECK
| CONSTRAINT constraint-name
| UNIQUE [ CLUSTERED ] ( index-columns-list )
| FOREIGN KEY fkey-name
| PRIMARY KEY
```

```
rename-object :
new-table-name
| column-name TO new-column-name
| CONSTRAINT constraint-name TO new-constraint-name
```

```
table-alteration :
PCTFREE DEFAULT
| [ NOT ] ENCRYPTED
```

```
new-column-attribute :
[ NOT ] NULL
| [ SET ] DEFAULT default-value
| COMPRESSED
| INLINE { inline-length | USE DEFAULT }
| PREFIX { prefix-length | USE DEFAULT }
| [ NO ] INDEX
| IDENTITY
| COMPUTE ( expression )
| column-constraint
```

```
table-constraint :
[ CONSTRAINT constraint-name ] {
   CHECK ( condition )
    | UNIQUE [ CLUSTERED | NONCLUSTERED ] ( column-name [ ASC | DESC ], ... )
    | PRIMARY KEY [ CLUSTERED | NONCLUSTERED ] ( column-name [ ASC |
DESC ], ... )
    | foreign-key
  }
```

```
column-constraint :
[ CONSTRAINT constraint-name ] {
   CHECK ( condition )
    | UNIQUE [ CLUSTERED | NONCLUSTERED ] [ ASC | DESC ]
    | PRIMARY KEY [ CLUSTERED | NONCLUSTERED ] [ ASC | DESC ]
```

```
    | REFERENCES table-name [ ( column-name ) ]
       [ MATCH [ UNIQUE ] { SIMPLE | FULL } ]
       [ actions ][ CLUSTERED | NONCLUSTERED ]
    | NOT NULL
   }
```

```
alterable-column-attribute :
[ NOT ] NULL
| DEFAULT default-value
| [ CONSTRAINT constraint-name ] CHECK { NULL | ( condition ) }
| [ NOT ] COMPRESSED
| INLINE { inline-length | USE DEFAULT }
| PREFIX { prefix-length | USE DEFAULT }
| [ NO ] INDEX
```

```
default-value :
 special-value
| string
| global variable
| [ - ] number
| ( constant-expression )
| ( sequence-expression )
| built-in-function( constant-expression )
| AUTOINCREMENT
| GLOBAL AUTOINCREMENT [ ( partition-size ) ]
```

```
special-value :
CURRENT DATABASE
| CURRENT DATE
| CURRENT TIME
| [ CURRENT ] TIMESTAMP
| CURRENT PUBLISHER
| CURRENT REMOTE USER
| [ CURRENT ] USER
| [ CURRENT ] UTC TIMESTAMP
| EXECUTING USER
| INVOKING USER
| LAST USER
| NULL
| PROCEDURE OWNER
| SESSION USER
```

```
foreign-key :
[ NOT NULL ] FOREIGN KEY [ role-name ]
   [ ( column-name [ ASC | DESC ], ... )
   REFERENCES table-name
   [ ( pkey-column-list ) ]
   [ MATCH [ UNIQUE] { SIMPLE | FULL } ]
   [ actions ] [ CHECK ON COMMIT ] [ CLUSTERED ]
   [ FOR OLAP WORKLOAD ]
```

```
actions :
[ ON UPDATE action ] [ ON DELETE action ]
```

```
action :
CASCADE | SET NULL | SET DEFAULT | RESTRICT
```

ビューの依存性の無効化

```
ALTER TABLE [owner.]table-name {
 DISABLE VIEW DEPENDENCIES
```

```
}
```

テーブル所有者の変更

```
ALTER TABLE [owner.]table-name ALTER OWNER TO owner
[ { PRESERVE | DROP } PRIVILEGES ]
[ { PRESERVE | DROP } FOREIGN KEYS ]
```

パラメータ

Adding clauses

テーブルにカラムまたはテーブル制約を追加するために使用可能な句が複数あります。

ADD column-name [AS] column-data-type [new-column-attribute ...] clause

この句は、テーブルに新しいカラムを追加し、カラムのデータ型と属性を指定するときに使用します。データ型を明示的に定義するか、%TYPE 属性を指定して、データ型を別のテーブルまたはビュー内のカラムのデータ型に設定します。%TYPE 属性を指定する場合、%TYPE 仕様内で参照されるオブジェクト上のデフォルト値、制約、および NULL の使用の可否などの他の属性は、継承される定義の一部ではなく、個別に指定する必要があります。

NULL and NOT NULL clauses

この句は、カラムに NULL を許容するかどうかを指定するときに使用します。デフォルトでは、新しいカラムに NULL 値を使用できます。BIT 型のカラムの作成時には、NOT NULL 制約が自動的に設定されますが、NULL 入力可として宣言することもできます。

DEFAULT clause

DEFAULT 値を指定する場合、カラムの値を指定しない INSERT 文のカラムの値としてこのデフォルト値が使用されます。INSERT 文はカラムの値を指定しません。DEFAULT 値を指定しない場合、これは DEFAULT NULL と同じです。

DEFAULT に指定できる値を次に示します。

constant-expression

DEFAULT 句では、データベースオブジェクトを参照していない定数式を使用できます。その結果、GETDATE や DATEADD などの関数を使用できます。式が関数または単純な値でない場合、括弧で囲みます。

global-variable

グローバル変数。

sequence-expression

データベース内のシーケンスの現在の値または次の値に、DEFAULT を設定します。

string

文字列値

AUTOINCREMENT

AUTOINCREMENT を使用する場合、カラムは整数データ型の 1 つ、または真数値型にします。

テーブルに挿入する場合、AUTOINCREMENT カラムの値を指定しないと、カラム内の任意の値より大きいユニーク値が生成されます。INSERT によって、カラムの現在の最大値よりも大きなカラム値を指定すると、その値は挿入され、以降の挿入時に開始ポイントとして使用されます。

ローを削除しても AUTOINCREMENT カウンタは減りません。ローの削除によって作成されたギャップは、挿入を行うときに明示的に割り当てることによってのみ埋めることができます。最大値未満のカラム値を明示的に挿入した後、明示的に割り当てられていない次のローを、以前の最大値より 1 大きい値を使ってオートインクリメントさせます。

カラムに直前に挿入された値は、グローバル変数 @@identity を調べることによって確認できます。

AUTOINCREMENT 値は、SYSTABCOL システムビューの max_identity カラムのデータ型に応じて、符号付き 64 ビット整数として保持されます。生成された次の値が、AUTOINCREMENT が割り当てられたカラムに格納できる最大値を超えた場合は、NULL が返されます。NULL を入力できないように宣言されたカラムであると (プライマリキーのカラムである場合など)、SQL エラーが生成されます。

カラムに使用する次の値は、sa_reset_identity プロシージャを使用してリセットできます。

GLOBAL AUTOINCREMENT

このデフォルトは、Mobile Link 同期環境または SQL Remote レプリケーションにおいて複数のデータベースを使用するときのために用意されたものです。

このオプションは AUTOINCREMENT と同じですが、ドメインはパーティションに分割されます。各分割には同じ数の値が含まれます。データベースの各コピーにユニークなグローバルデータベース ID 番号を割り当てます。データベースサーバでは、データベースのデフォルト値は、そのデータベース番号でユニークに識別された分割からのみ設定されます。

AUTOINCREMENT キーワードの直後に括弧で分割サイズを指定できます。この分割サイズには任意の正の整数を設定できますが、通常、分割サイズは、サイズの値がすべての分割で不足しないように選択されます。

カラムの型が BIGINT または UNSIGNED BIGINT である場合、デフォルトの分割サイズは 2^{32} = 4294967296 です。それ以外の型のカラムの場合、デフォルトの分割サイズは 2^{16} = 65536 です。特に、カラムの型が INT または BIGINT ではない場合は、これらのデフォルト値が適切ではないことがあるため、分割サイズを明示的に指定するのが最も賢明です。

このデフォルトを使用する場合、各データベース内のパブリックオプション global_database_id は、ユニークな正の整数に設定します。この値は、データベースをユニークに識別し、デフォルト値の割り当て元の分割を示します。使用できる値の範囲は、$np + 1$ から $p(n + 1)$ です。ここで、n はパブリックオプション global_database_id の値を表し、p は分割サイズを表します。たとえば、分割サイズを 1000、global_database_id を 3 に設定すると、範囲は 3001 ～ 4000 になります。

前の値が $p(n + 1)$ 未満であれば、このカラム内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になります。カラムに値が含まれていない場合、最初のデフォルト値は $np + 1$ です。デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けません。つまり、$np + 1$ より小さいか、$p(n + 1)$ より大きい数には影響されません。Mobile Link または SQL Remote 経由で別のデータベースからレプリケートした場合、このような値になることがあります。

カラムに直前に挿入された値は、グローバル変数 @@identity を調べることによって確認できます。

GLOBAL AUTOINCREMENT 値は、SYSTABCOL システムビューの max_identity カラムのデータ型に応じて、符号付き 64 ビット整数として保持されます。NULL 値は、分割で値が不足したときにも生成されます。NULL を入力できないように宣言されたカラムであると (プライマリキーのカラムである場合など)、SQL エラーが生成されます。この場合には、別の分割からデフォルト値を選択できるように、データベースに global_database_id の新しい値を割り当ててください。未使用の値が残り少ないことを検出し、このような状態を処理するには、GlobalAutoincrement タイプのイベントを作成します。

public オプション global_database_id は、負の値に設定できないため、選択された値は常に正になります。ID 番号の最大値を制限するのは、カラムデータ型と分割サイズだけです。

public オプション global_database_id がデフォルト値の 2147483647 に設定されると、NULL 値がカラムに挿入されます。NULL 値が許可されていない場合に、ローの挿入を試みるとエラーが発生します。

カラムに使用する次の値は、sa_reset_identity プロシージャを使用してリセットできます。

special-value

DEFAULT 句では、複数の特別値の 1 つを使用します (たとえば、CURRENT DATE)。これには次のもの特別値が含まれますが、これに限定されません。

[CURRENT] TIMESTAMP

テーブル内の各ローが最後に変更された日付を示すことができます。カラムの宣言に DEFAULT TIMESTAMP が指定されている場合は、ローを挿入するとタイムスタンプのデフォルト値が割り付けられます。この値は、ローが更新されるたびに現在の日付と時刻に基づいて更新されます。

挿入されたローにタイムスタンプのデフォルト値を割り付け、そのローが更新されてもタイムスタンプを更新しない場合は、DEFAULT TIMESTAMP の代わりに DEFAULT CURRENT TIMESTAMP を使用します。

DEFAULT TIMESTAMP で宣言されたカラムにはユニークな値が入ります。これにより、アプリケーションは、ほぼ同時に行われた同じローの更新を検出できます。現在の TIMESTAMP 値が直前の値と同じ場合は、default_timestamp_increment オプションの値が加えられます。

default_timestamp_increment オプションに基づいて、TIMESTAMP 値を自動的にトランケートできます。これは、記録されるタイムスタンプ値の精度が低い他のデータベースソフトウェアとの互換性を維持する場合に便利です。

グローバル変数 @@dbts は、DEFAULT TIMESTAMP を使用するカラムの最後に生成された値を表す TIMESTAMP 値を返します。

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこれらの値が計算されます。

[CURRENT] UTC TIMESTAMP

テーブル内の各ローが最後に変更された日付を示すことができます。カラムの宣言に DEFAULT UTC TIMESTAMP が指定されている場合は、ローを挿入するとタイムスタンプのデフォルト値が割り付けられます。この値は、ローが更新されるたびに現在の協定世界時 (UTC: Coordinated Universal Time) に基づいて更新されます。

挿入されたローにタイムスタンプのデフォルト値を割り付け、そのローが更新されてもタイムスタンプを更新しない場合は、DEFAULT UTC TIMESTAMP の代わりに DEFAULT CURRENT UTC TIMESTAMP を使用します。

このデフォルトの動作は TIMESTAMP や CURRENT TIMESTAMP と同じですが、日付と時刻が協定世界時 (UTC: Coordinated Universal Time) になる点が異なります。

LAST USER

LAST USER は、ローを最後に変更したユーザのユーザ ID です。

LAST USER は文字データ型のカラムでデフォルト値として使用できます。

INSERT の場合、このデフォルトは CURRENT USER と同じ効果があります。

UPDATE では、LAST USER のデフォルトを持つカラムが明示的に変更されていなければ、現在のユーザ名に変更されます。

DEFAULT TIMESTAMP または DEFAULT UTC TIMESTAMP とともに使用すると、LAST USER のデフォルトを使用して、ローを最後に変更したユーザと日時の両方を記録できます (ただし、別々のカラムに記録されます)。

column-constraint clause

この句はカラムに制約を追加するときに使用します。

> **注記**
>
> 検査制約の例外を指定して新規の制約を追加すると、データベースサーバは既存の値を検証して、制約を満たすことを確認します。テーブルの変更が完了した後に発生する操作の場合にのみ、検査制約が実行されます。

使用できるカラムの制約を次に示します。

CHECK clause

この制約で、任意の条件を検証できます。たとえば、CHECK 制約を使うと、Sex というカラムには M または F の値だけが確実に入ります。

テーブル内の 2 つ以上のカラム間の関係 (カラム A はカラム B 未満である必要があるなど) が関与する CHECK 制約の作成が必要な場合は、代わりにテーブル制約を定義します。

UNIQUE clause

このサブ句は、カラムの値をユニークにする必要があることを指定するとき、クラスタドインデックスと非クラスタドインデックスのどちらを作成するかを指定するときに使用します。

PRIMARY KEY clause

このサブ句は、カラムをプライマリキーにするとき、クラスタドインデックスを使用するかどうかを指定するときに使用します。

REFERENCES clause

このサブ句は、別のテーブルへの参照を追加または変更するとき、一致を処理する方法を指定するとき、クラスタドインデックスを使用するかどうかを指定するときに使用します。

MATCH clause

このサブ句は、複数カラムの外部キーを使用するときに、何を一致と見なすかを制御するために使用します。また、キーの一意性を指定することで、一意性を別に宣言する必要がなくなります。

NULL and NOT NULL clauses

この句は、カラムに NULL 値を許容するかどうかを指定するときに使用します。デフォルトでは、NULL を使用できます。

COMPRESSED clause

この句はカラムを圧縮するときに使用します。

INLINE and PREFIX clauses

INLINE 句は、ローに格納する最大 BLOB サイズをバイト単位で指定します。INLINE 句で指定された値以下のサイズの BLOB がローに格納されます。INLINE 句で指定された値を超えるサイズの BLOB は、ロー外のテーブル拡張ページに格納されます。また、BLOB のサイズが INLINE 句で指定された値より大きい場合、BLOB の先頭から数バイト分を複製してローに保持することができます。PREFIX 句は、ローに保持されるバイト数を指定します。ローの

受け入れと拒否の判断に BLOB のプレフィクスバイトを必要とする要求のパフォーマンスは、PREFIX 句によって向上する可能性があります。

圧縮されたカラムのプレフィクスデータは圧縮されずに格納されるため、要求を満たすために必要なすべてのデータがプレフィクス内に格納されている場合は、解凍は必要ありません。

INLINE と PREFIX のどちらも指定しない場合、または USE DEFAULT を指定している場合、デフォルト値は次のように適用されます。

- CHAR、NCHAR、LONG VARCHAR などの文字データ型のカラムの場合、INLINE のデフォルト値は 256 で、PREFIX のデフォルト値は 8 です。
- BINARY、LONG BINARY、VARBINARY、BIT、VARBIT、LONG VARBIT、BIT VARYING、UUID などのバイナリデータ型のカラムの場合、INLINE のデフォルト値は 256 で、PREFIX のデフォルト値は 0 です。

  注記

  デフォルト以外の設定が必要な特殊な環境でない限り、デフォルト値を使用するようにしてください。デフォルト値は、パフォーマンスとディスク領域の要件のバランスを取って選択されています。たとえば、INLINE に大きな値を設定し、すべての BLOB をインラインで格納するようにした場合、ローの処理パフォーマンスは低下することがあります。また、PREFIX の値を大きくしすぎると、BLOB の一部を複製したプレフィクスデータのために、BLOB の格納に必要なディスク領域のサイズが増えることになります。

  値の 1 つのみを指定する場合、その他の値は指定した値と競合しない最大の値に自動的に設定されます。INLINE 値と PREFIX 値のどちらも、データベースページサイズを超えるサイズを指定できません。また、ローデータの格納に使用できないテーブルページには、小さいサイズですが予約済みのオーバヘッドがあります。そのため、データベースのページサイズに近い INLINE 値を指定すると、やや少ないバイト数がインラインで格納される可能性があります。

INDEX and NO INDEX clauses

BLOB の格納時に (文字型またはバイナリ型のみ)、BLOB サイズの内部しきい値 (約 8 データベースページ) を超える挿入値に対して BLOB インデックスを作成する場合は、INDEX を指定します。これはデフォルトの動作です。

BLOB インデックスは、BLOB 内のランダムアクセス検索が必要なときにパフォーマンスを改善する可能性があります。ただし、ランダムアクセスの必要がない画像ファイルやマルチメディアファイルなど、BLOB 値の種類によっては、BLOB インデックスをオフにするとパフォーマンスが改善されることがあります。カラムの BLOB インデックスをオフにするには、NO INDEX を指定します。

注記

BLOB インデックスは、テーブルインデックスとは異なります。テーブルインデックスは、1 つ以上のカラムの値のインデックスとして作成されます。

IDENTITY clause

IDENTITY 句は、DEFAULT AUTOINCREMENT の使用に対する Transact-SQL 互換の代替手段です。IDENTITY を指定して定義されたカラムは DEFAULT AUTOINCREMENT として実装されます。

COMPUTE clause

カラムが COMPUTE 句を使って作成される場合、すべてのローの値は式で提供されます。この制約を付けて作成されたカラムは、読み込み専用カラムです。この値は、ローが修正されたときにデータベースサーバによって変更されます。COMPUTE 式は、非決定的な値を返しません。たとえば、CURRENT TIMESTAMP などの特別値や非決定的関数は指定できません。COMPUTE 式が非決定的な値を返したとしても、それをクエリに含まれる式との一致に使用することはできません。

リモートテーブルでは COMPUTE 句は無視されます。

計算カラムの値を変更しようとする UPDATE 文は、そのカラムに対応するトリガを起動します。

ADD table-constraint clause

この句はテーブルの制約を追加するときに使用します。テーブルの制約によって、テーブルのデータカラムが保持できる内容が制限されます。テーブルの制約を追加または変更するときに、オプションの制約名を使用して各制約を修正または削除することができます。追加できるテーブルの制約一覧を以下に示します。

UNIQUE

このサブ句は、`column-list` に指定するカラム値をユニークにすることを指定するとき、オプションでクラスタドインデックスを使用するかどうかを指定するときに使用します。

PRIMARY KEY

このサブ句は、テーブルのプライマリキーを追加または変更するとき、クラスタドインデックスを使用するかどうかを指定するときに使用します。テーブルは CREATE TABLE 文または別の ALTER TABLE 文が作成したプライマリキーを持っていてはなりません。

foreign-key

このサブ句は外部キーを制約として追加するときに使用します。従属マテリアライズドビューを持つテーブルに対して、ADD FOREIGN KEY 以外のサブ句を ALTER TABLE 文で使用すると、ALTER TABLE 文は失敗します。その他すべての句の場合は、従属マテリアライズドビューを無効にし、変更が完了してから、再度有効にする必要があります。

MATCH サブ句を指定すると、複数カラムの外部キーを使用するときに、何が一致と見なされるようにするかを制御できます。また、キーの一意性を指定することで、一意性を別に宣言する必要がなくなります。

ADD PCTFREE clause

各テーブルページに確保する空き領域の割合を指定します。空き領域は、データが更新されたときにローのサイズが増えた場合に使用されます。テーブルページに空き領域がない場合は、ページのローのサイズが増えるたびに、ローを複数のテーブルページに分割することが必要になり、ローの断片化が発生します。また、パフォーマンス低下の可能性があります。空き領域のパーセンテージを 0 に指定すると、各ページに空き領域を残しません。各ページは完全にパックされます。空き領域のパーセントを高い値に設定すると、各ローは単独でページに挿入されます。PCTFREE が設定されない場合、または削除された場合、データベースのページサイズに応じたデフォルトの PCTFREE 値が適用されます (ページサイズが 4 KB 以上の場合は 200 バイト)。PCTFREE の値は、ISYSTAB システムテーブルに格納されます。PCTFREE を設定すると、テーブルページに対するそれ以降のすべての挿入操作で、新しい値が使用されます。しかし、すでに挿入済みであったローは影響を受けません。値を変更しなければ、そのままの値が維持されます。PCTFREE は、ベーステーブル、グローバルテンポラリテーブル、またはローカルテンポラリテーブルに対して指定できます。

Altering clauses

カラムまたはテーブルを変更するために使用可能な句が複数あります。

ALTER column-name column-alteration clause

この句は、指定したカラムの属性を変更するときに使用します。カラムに一意性制約、外部キー、またはプライマリキーが設定されている場合は、カラムのデフォルトのみを変更できます。ただし、その他を変更する場合は、キーまたは制約を削除してからカラムを修正してください。

column-data-type clause

この句は、カラムの長さまたはデータ型を変更するときに使用します。新しいデータ型を指定するか、%TYPE 属性を使用して、データ型を別のテーブルまたはビュー内のカラムのデータ型に設定します。必要に応じて、変更されるカラムのデータを新しいデータ型に変換します。変換エラーが発生すると、操作は失敗となり、テーブルは変更されませ

ん。カラムのサイズを減らすことはできません。たとえば、VARCHAR(100) を VARCHAR(50) に変更することはできません。

[NOT] NULL clause

この句は、カラムに NULL を許容するかどうかの指定を変更するときに使用します。NOT NULL を指定し、既存ローのカラム値が NULL の場合、操作は失敗し、テーブルは変更されません。

CHECK NULL

この句は、カラムの検査制約をすべて削除するときに使用します。

DEFAULT clause

この句は、カラムのデフォルト値を変更するときに使用します。

DEFAULT NULL clause

この句は、カラムのデフォルト値を削除するときに使用します。

[CONSTRAINT constraint-name] CHECK { NULL | (condition) } clause

この句はカラムに検査制約を追加するために使用します。

テーブル内の 2 つ以上のカラム間の関係 (カラム A はカラム B 未満である必要があるなど) が関与する CHECK 制約の作成が必要な場合は、代わりにテーブル制約を定義します。

[NOT] COMPRESSED clause

この句は、カラムを圧縮するかどうかの設定を変更するときに使用します。

INLINE and PREFIX clauses

INLINE 句は、ローに格納する最大 BLOB サイズをバイト単位で指定します。INLINE 句で指定された値以下のサイズの BLOB がローに格納されます。INLINE 句で指定された値を超えるサイズの BLOB は、ロー外のテーブル拡張ページに格納されます。また、BLOB のサイズが INLINE 句で指定された値より大きい場合、BLOB の先頭から数バイト分を複製してローに保持することができます。PREFIX 句は、ローに保持されるバイト数を指定します。ローの受け入れと拒否の判断に BLOB のプレフィクスバイトを必要とする要求のパフォーマンスは、PREFIX 句によって向上する可能性があります。

圧縮されたカラムのプレフィクスデータは圧縮されずに格納されるため、要求を満たすために必要なすべてのデータがプレフィクス内に格納されている場合は、解凍は必要ありません。

INLINE と PREFIX のどちらも指定しない場合、または USE DEFAULT を指定している場合、デフォルト値は次のように適用されます。

- CHAR、NCHAR、LONG VARCHAR などの文字データ型のカラムの場合、INLINE のデフォルト値は 256 で、PREFIX のデフォルト値は 8 です。
- BINARY、LONG BINARY、VARBINARY、BIT、VARBIT、LONG VARBIT、BIT VARYING、UUID などのバイナリデータ型のカラムの場合、INLINE のデフォルト値は 256 で、PREFIX のデフォルト値は 0 です。

  > **注記**
  >
  > デフォルト以外の設定が必要な特殊な環境でない限り、デフォルト値を使用するようにしてください。デフォルト値は、パフォーマンスとディスク領域の要件のバランスを取って選択されています。たとえば、INLINE に大きな値を設定し、すべての BLOB をインラインで格納するようにした場合、ローの処理パフォーマンスは低下することがあります。また、PREFIX の値を大きくしすぎると、BLOB の一部を複製したプレフィクスデータのために、BLOB の格納に必要なディスク領域のサイズが増えることになります。

  値の 1 つのみを指定する場合、その他の値は指定した値と競合しない最大の値に自動的に設定されます。INLINE 値と PREFIX 値のどちらも、データベースページサイズを超えるサイズを指定できません。また、ローデ

ータの格納に使用できないテーブルページには、小さいサイズですが予約済みのオーバヘッドがあります。そのため、データベースのページサイズに近い INLINE 値を指定すると、やや少ないバイト数がインラインで格納される可能性があります。

INDEX and NO INDEX clauses

BLOB の格納時に (文字型またはバイナリ型のみ)、BLOB サイズの内部しきい値 (約 8 データベースページ) を超える挿入値に対して BLOB インデックスを作成する場合は、INDEX を指定します。これはデフォルトの動作です。

BLOB インデックスは、BLOB 内のランダムアクセス検索が必要なときにパフォーマンスを改善する可能性があります。ただし、ランダムアクセスの必要がない画像ファイルやマルチメディアファイルなど、BLOB 値の種類によっては、BLOB インデックスをオフにするとパフォーマンスが改善されることがあります。カラムの BLOB インデックスをオフにするには、NO INDEX を指定します。

> **注記**
>
> BLOB インデックスは、テーブルインデックスとは異なります。テーブルインデックスは、1 つ以上のカラムの値のインデックスとして作成されます。

SET COMPUTE clause

カラムが COMPUTE 句を使って作成される場合、すべてのローの値は式で提供されます。この制約を付けて作成されたカラムは、読み込み専用カラムです。この値は、ローが修正されたときにデータベースサーバによって変更されます。COMPUTE 式は、非決定的な値を返しません。たとえば、CURRENT TIMESTAMP などの特別値や非決定的関数は指定できません。COMPUTE 式が非決定的な値を返したとしても、それをクエリに含まれる式との一致に使用することはできません。

リモートテーブルでは COMPUTE 句は無視されます。

計算カラムの値を変更しようとする UPDATE 文は、そのカラムに対応するトリガを起動します。

ALTER CONSTRAINT constraint-name CHECK clause

この句は、指定したテーブルの検査制約を変更するときに使用します。

テーブル内の 2 つ以上のカラム間の関係 (カラム A はカラム B 未満である必要があるなど) を指定する制約を変更する場合は、代わりにテーブル制約を定義します。

Dropping clauses

DROP DEFAULT

テーブルまたは指定したカラムに設定されたデフォルト値を削除します。既存の値は変更されません。

DROP COMPUTE

指定したカラムの COMPUTE 属性を削除します。この文はテーブル内の既存の値を変更しません。

DROP CHECK

テーブルまたは指定したカラムのすべての検査制約を削除します。DELETE CHECK も使用できます。

DROP CONSTRAINT constraint-name

テーブルまたは指定したカラムの指定した制約を削除します。DELETE CONSTRAINT も使用できます。

DROP column-name

テーブルから指定したカラムを削除します。DELETE `column-name` も使用できます。カラムがインデックス、一意性制約、外部キー、またはプライマリキーに含まれている場合は、インデックス、制約またはキーを削除してからカラムを削除してください。このようにするとカラムを参照する検査制約は削除されません。

DROP UNIQUE (column-name ...)

指定したカラムの一意性制約を削除します。この一意性制約を参照する外部キーがあれば、それも削除されます。DELETE UNIQUE (column-name ...) も使用できます。

DROP FOREIGN KEY fkey-name

指定した外部キーを削除します。DELETE FOREIGN KEY fkey-name も使用できます。

DROP PRIMARY KEY

プライマリキーを削除します。このテーブルのプライマリキーを参照するすべての外部キーも削除します。DELETE PRIMARY KEY も使用できます。

Renaming clauses

RENAME new-table-name

テーブルの名前を new-table-name に変更します。場合によっては、古いテーブル名を使用しているアプリケーションを修正する必要があります。

RENAME column-name TO new-column-name

カラムの名前を new-column-name に変更します。場合によっては、古いカラム名を使用しているアプリケーションを修正する必要があります。

RENAME CONSTRAINT constraint-name TO new-constraint-name

制約の名前を new-constraint-name に変更します。

ALTER TABLE...RENAME CONSTRAINT constraint-name TO new-constraint-name を RI 制約に使用すると、制約名のみが変更され、基本となるインデックス名や外部キーの役割名 (該当する場合) は変更されません。基本となるインデックス名または役割名を変更する場合は、ALTER INDEX 文を使用します。

table-alteration clauses

この句は、次のテーブルの属性を変更するときに使用します。

PCTFREE DEFAULT

この句は、テーブルの空き割合設定をデフォルト値 (ページサイズが 4 KB 以上の場合は 200 バイト) に変更するときに使用します。

[NOT] ENCRYPTED

この句は、テーブルを暗号化するかどうかの設定を変更するときに使用します。テーブルを暗号化するには、データベースでテーブルの暗号化があらかじめ有効になっている必要があります。その後、データベースの作成時に指定された暗号化キーとアルゴリズムを使用してテーブルが暗号化されます。

テーブルを暗号化した後でも、暗号化前にテンポラリファイルまたはトランザクションログに含まれていたテーブルのデータはすべて暗号化されない形式で保存されます。これを解決するには、データベースを再起動してテンポラリファイルを削除します。-o オプションでバックアップユーティリティ (dbbackup) を実行するか、BACKUP DATABASE 文を使用して、トランザクションログをバックアップして新規のログを開始します。

テーブルの暗号化が有効の場合、暗号化されるテーブルのテーブルページ、関連するインデックスページ、テンポラリファイルのページ、および暗号化されるテーブルのトランザクションを含むトランザクションログページが暗号化されます。

DISABLE VIEW DEPENDENCIES clause

この句は、通常の従属ビューを無効にするときに使用します。従属マテリアライズドビューは無効になりません。ALTER MATERIALIZED VIEW...DISABLE 文を実行して、それぞれの従属マテリアライズドビューを無効にします。

ALTER OWNER clause

テーブルの所有者を変更するには、次の手順に従います。

- ALTER ANY OBJECT OWNER 権限が必要です。
- 次のいずれかのシステム権限が必要です。テーブルに対する ALTER 権限、ALTER ANY TABLE 権限、または ALTER ANY OBJECT 権限。
- 新しい所有者は、同じ名前のテーブルを所有することはできません。
- 有効化されたマテリアライズドビューはテーブルを参照できません。

  PRESERVE or DROP PRIVILEGES

  新しい所有者に古い所有者と同じ権限を持たせたくない場合は、DROP PRIVILEGES (デフォルト) を指定して、明示的に付与されていたテーブルへのユーザアクセスを許可する権限をすべて取り消すことができます。そのテーブルの所有者の権限が暗黙的に与えられたものである場合、その権限が新しい所有者に付与され、前の所有者から削除されます。
  PRESERVE or DROP FOREIGN KEYS

  新しい所有者が参照テーブル内のデータにアクセスできないようにするには、DROP FOREIGN KEYS (デフォルト) を指定して、テーブル内のすべての外部キーと、そのテーブルを参照しているすべての外部キーを削除できます。

備考

ALTER TABLE 文は、既存テーブルのテーブル属性 (カラム定義、制約など) を変更します。

文に対して複数のタイプの句を使用しないでください。

データベースサーバは、データベース内のオブジェクトの依存関係を追跡します。テーブルのスキームを変更すると、従属ビューに影響が及ぶ場合があります。また、変更するテーブルに依存しているマテリアライズドビューがある場合、あらかじめ ALTER MATERIALIZED VIEW...DISABLE 文を使用して無効にしておきます。

ローカルテンポラリテーブル上では ALTER TABLE を使用できません。

ALTER TABLE 文は、他の接続で現在使用中のテーブルに影響を及ぼす場合には実行できません。ALTER TABLE には時間がかかり、データベースサーバは、文の処理中にそのテーブルを参照する処理ができません。

IMMEDIATE REFRESH として定義されたテキストインデックスが構築されたカラムを変更すると、テキストインデックスがすぐに再構築されます。テキストインデックスが AUTO REFRESH または MANUAL REFRESH として定義されている場合は、テキストインデックスが次回の再表示時に再構築されます。

ALTER TABLE 文を実行すると、データベースサーバは、自動的に再コンパイルされる従属ビューに対するカラム権限をリストアしようとします。再コンパイルされたビューに存在しないカラムに対する権限は失われます。

ALTER TABLE には、テーブルへの排他的アクセスが必要です。

グローバルテンポラリテーブルは、このテンポラリテーブルを参照したすべてのユーザが切断されるまで変更できません。

この文はスナップショットトランザクション内では使用できません。

権限

そのテーブルの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- そのテーブルに対する ALTER 権限

- ALTER ANY TABLE システム権限
- ALTER ANY OBJECT システム権限
- MANAGE ANY DBSPACE システム権限

テーブル所有者を変更するには、ALTER ANY OBJECT OWNER システム権限も必要です。

テーブルのインデックスを作成、変更、削除するには、それぞれ CREATE ANY INDEX、ALTER ANY INDEX、DROP ANY INDEX システム権限が必要です。

関連する動作

オートコミット。

チェックポイントは ALTER TABLE 操作の開始時に実行されます。また、ALTER 操作が完了するまでチェックポイントは中断されます。

カラムまたはテーブルを変更すると、変更を加えたカラムを参照するストアドプロシージャ、ビュー、その他のアイテムは動作しなくなる場合があります。

宣言されたカラムの長さまたは型を変更した場合、またはカラムを削除した場合、そのカラムの統計情報は削除されます。

標準

ANSI/ISO SQL 標準

コア機能。ANSI/ISO SQL 標準では、ADD CONSTRAINT と DROP CONSTRAINT と同様に、ADD COLUMN と DROP COLUMN がコア機能としてサポートされています。カラムの DEFAULT 値を追加、変更、削除する ALTER [COLUMN] は、SQL 機能 F381 です。ANSI/ISO SQL 標準でカラムのデータ型を変更するには、SQL 言語機能 F382 である SET DATA TYPE 句を指定します。ただし、ソフトウェアでは ALTER 句を直接指定してカラムのデータ型を変更できます。

ALTER CONSTRAINT、RENAME、PCTFREE、ENCRYPTED、DISABLE MATERIALIZED VIEW など、ソフトウェアでサポートされている他の句は標準にありません。カラム定義や、カラムとテーブルの制約定義に対する拡張機能のサポートは、標準にないか、標準のオプション機能です。

Transact-SQL

ALTER TABLE は、Adaptive Server Enterprise でサポートされています。Adaptive Server Enterprise では、ADD CONSTRAINT と DROP CONSTRAINT 以外に、ADD COLUMN 句と DROP COLUMN 句がサポートされています。Adaptive Server Enterprise では、ALTER 句のキーワード ALTER ではなく MODIFY が使用されます。Adaptive Server Enterprise では、カラムの DEFAULT 値の変更に REPLACE 句が使用されます。Adaptive Server Enterprise では、特定のテーブルのトリガを有効/無効にするときに ALTER TABLE も使用されます。この機能は、ソフトウェアではサポートされていません。

例

次の例は、新しいタイムスタンプカラム TimeStamp を Customers テーブルに追加します。この次の文を実行するには、Customers テーブルに対する ALTER 権限が必要です。

```
ALTER TABLE GROUPO.Customers
   ADD TimeStamp AS TIMESTAMP DEFAULT TIMESTAMP;
```

次の例は、前の例で追加した新しいタイムスタンプカラム TimeStamp を削除します。

```
ALTER TABLE GROUPO.Customers
DROP TimeStamp;
```

Customers テーブルの Street カラムは、現在 35 文字まで保持できます。最大 50 文字まで保持できるようにするには、次を実行します。

```
ALTER TABLE GROUPO.Customers
ALTER Street CHAR(50);
```

次の例は、Customers テーブルにカラムを追加して、各顧客に販売担当を割り当てます。この次の文を実行するには、外部キーを作成するために CREATE ANY INDEX システム権限も必要です。

```
ALTER TABLE GROUPO.Customers
ADD SalesContact INTEGER
REFERENCES GROUPO.Employees ( EmployeeID )
ON UPDATE CASCADE
ON DELETE SET NULL;
```

この外部キーは、カスケード更新で構成され、削除される時に NULL が設定されます。従業員がその従業員 ID を変更すると、カラムが更新してこの変更を反映します。従業員が会社を辞めて、従業員 ID が削除されると、カラムは NULL に設定されます。

次の例は、SalesOrders.SalesRepresentative カラムに外部キー FK_SalesRepresentative_EmployeeID2 を作成し、Employees.EmployeeID にリンクします。次の文を実行するには、SalesOrder テーブルに対する ALTER 権限を持っていることが必要です。

```
ALTER TABLE GROUPO.SalesOrders
   ADD CONSTRAINT FK_SalesRepresentative_EmployeeID2
   FOREIGN KEY ( SalesRepresentative )
   REFERENCES GROUPO.Employees (EmployeeID);
```

次の例では、デフォルトが AUTOINCREMENT であるカラムを追加します。この例では、カラム値を割り当てられた NULL 入力可の AUTOINCREMENT カラムを含むように既存のすべての顧客ローが変更されますが、データベースサーバはどのローにどの値が割り当てられるかを保証しません。

```
ALTER TABLE GROUPO.Customers
   ADD Surrogate_key INTEGER DEFAULT AUTOINCREMENT;
```

次の例は、架空のテーブル mytable の所有者を Bob に変更します。

```
ALTER TABLE mytable ALTER OWNER TO bob;
```

関連情報

文字列 [10 ページ]
特別値 [83 ページ]
%TYPE および %ROWTYPE 属性 [110 ページ]
sa_reset_identity システムプロシージャ [1597 ページ]
ALTER MATERIALIZED VIEW 文 [659 ページ]
BACKUP DATABASE 文 [739 ページ]
ALTER INDEX 文 [652 ページ]
CREATE TABLE 文 [952 ページ]
DROP TABLE 文 [1074 ページ]
SQL データ型 [124 ページ]
SQL 変数 [118 ページ]
@@identity グローバル変数 [121 ページ]

1.4.4.28 ALTER TEXT CONFIGURATION 文

テキスト設定オブジェクトを変更します。

構文

```
ALTER TEXT CONFIGURATION [ owner.]config-name
STOPLIST stoplist-string
| DROP STOPLIST
| { MINIMUM | MAXIMUM } TERM LENGTH integer }
| TERM BREAKER { GENERIC [ EXTERNAL NAME external-call ] | NGRAM }
| PREFILTER EXTERNAL NAME external-call
| DROP PREFILTER
| SAVE OPTION VALUES [ FROM CONNECTION ]
```

```
external-call :
[ system-configuration:]function-name@library-file-prefix[.{ so | dll} ]
```

```
system-configuration :
{ generic-operating-system | specific-operating-system } [ (processor-
architecture) ]
```

```
generic-operating-system :
{ Unix | Windows }
```

```
specific-operating-system :
{ AIX | HPUX | Linux | OSX | Solaris | WindowsNT }
```

```
processor-architecture :
{ 32 | 64 | ARM | IA64 | PPC | SPARC | X86 | X86_64 }
```

パラメータ

STOPLIST 句

この句は、テキストインデックスの構築時に無視する単語のリストを作成したり、置き換えたりするときに使用します。このテキスト設定オブジェクトを使用すると、このリストに指定される単語もクエリで無視されます。ストップリストの単語は、スペースで区切ります。たとえば、`STOPLIST 'because about therefore only'` のように記述します。ストップリストの単語には、ホワイトススペースを含めることはできません。

各言語用のストップリストのサンプルは、*%SQLANYSAMP17%*`¥SQLAnywhere¥SQL` にあります。

ストップリストの単語に、英数字以外の文字は使用できません。ストップリストの長さは、8000 バイト未満にしてください。

単語をストップリストに加えるかどうかは、慎重に検討してください。

DROP STOPLIST 句

この句は、テキスト設定オブジェクトのストップリストを削除するときに使用します。

MINIMUM TERM LENGTH 句

NGRAM テキストインデックスを使用する場合、MINIMUM TERM LENGTH 句は無視されます。

テキストインデックスに使用できる単語の最小文字数。テキストインデックスの構築時または再表示時に、この設定より短い単語は無視されます。このオプションの値は、0 より大きくする必要があります。このオプションを MAXIMUM TERM LENGTH より大きい値に設定すると、MAXIMUM TERM LENGTH の値が自動的に調整され、MINIMUM TERM LENGTH の新しい値と同じになります。

MAXIMUM TERM LENGTH 句

NGRAM テキストインデックスで MAXIMUM TERM LENGTH 句を使用して、文字列の分割後の N-gram のサイズを設定します。MAXIMUM TERM LENGTH より短い単語は、インデックスが作成されません。

GENERIC テキストインデックスで MAXIMUM TERM LENGTH 句を使用して、テキストインデックスに含める単語の最大文字数を設定します。テキストインデックスの構築時または再表示時に、この設定より長い単語は無視されます。MAXIMUM TERM LENGTH の値は、60 以下にする必要があります。このオプションを MINIMUM TERM LENGTH より小さい値に設定すると、MINIMUM TERM LENGTH の値が自動的に調整され、MAXIMUM TERM LENGTH の新しい値と同じになります。

TERM BREAKER 句

カラム値を単語に分割するときに使用するアルゴリズムの名前。選択肢は GENERIC (デフォルト) または NGRAM です。

GENERIC

GENERIC では、TERM BREAKER GENERIC を指定して GENERIC 単語区切りの組み込みアルゴリズムを使用するか、または、TERM BREAKER GENERIC EXTERNAL NAME 句を使用して外部アルゴリズムを指定できます。

組み込み GENERIC アルゴリズムは、英数字以外の文字で区切られた 1 つまたは複数の英数字の文字列を単語として扱います。

外部ライブラリにある単語区切りファンクションのエントリポイントを指定するには、TERM BREAKER GENERIC EXTERNAL NAME 句を指定します。このことは、インデックスの作成やクエリを行う前の単語の区切り方に関してカスタム要件 (アポストロフィを単語区切りとしてではなく、単語の一部と見なすなど) がある場合に便利です。

`external-call` は、複数のファンクションやライブラリを指定でき、ライブラリのファイル拡張子 (通常、Windows の場合は `.dll`、UNIX の場合は `.so`) を含めることができます。ファイル拡張子がない場合は、ライブラリに対するプラットフォーム固有のデフォルトのファイル拡張子がデータベースサーバで使用されます。たとえば、`EXTERNAL`

NAME 'TermBreakFunct1@myTBlib;Unix:TermBreakFunct2@myTBlib' は、Windows の場合は myTBlib.dll から TermBreakFunct1 ファンクションを呼び出し、UNIX の場合は myTBlib.so から TermBreakFunct2 ファンクションを呼び出します。

NGRAM

組み込み NGRAM アルゴリズムは文字列を N-gram に分割します。N-gram は、ある文字列中の n 文字分の部分文字列です。外部単語区切りが指定されていない場合、NGRAM 単語区切りは、あいまい (近似) 一致、または単語の区切りにホワイトスペースや非英数字文字を使用しないドキュメントに必要です。

PREFILTER EXTERNAL NAME 句

PREFILTER EXTERNAL NAME 句を指定して、外部ライブラリにある事前フィルタファンクションのエントリポイントを指定します。このことは、テキストデータをバイナリデータ (PDF など) から抽出する必要がある場合に便利です。また、データ (HTML など) のインデックスを作成する前に削除するフォーマット情報やイメージが、インデックスを作成するテキストに含まれている場合にも便利です。

external-call にはオペレーティングシステム、プロセッサ、ライブラリ、および関数の複数のセットを指定できるため、より精密に指定された設定が、それよりも精密さに欠けて定義された設定よりも優先されます。たとえば、Solaris(X86_64):myfunc64@mylib.so は Solaris:myfunc64@mylib.so よりも優先されます。

system-configuration がサポートされている構文で system-configuration を指定しないと、プロシージャがすべてのプラットフォームで稼働すると見なされます。UNIX は、AIX、HPUX、Linux、OS X、および Solaris の UNIX ベースオペレーティングシステムを意味します。一般的な用語である Windows は、Windows オペレーティングシステムのすべてのバージョンを意味します。

いずれかの呼び出しで UNIX を指定すると、他の呼び出しは Windows 用であると見なされます。

specific-operating-system および processor-architecture の値は、SQL Anywhere サーバでサポートされるオペレーティングシステムおよびプロセッサです。

ライブラリ名 (library-file-prefix) にはファイル拡張子が付きます。この拡張子は通常、Windows では .dll、UNIX では .so です。ファイル拡張子がない場合は、ライブラリに対するプラットフォーム固有のデフォルトのファイル拡張子がデータベースサーバで使用されます。たとえば、PREFILTER EXTERNAL NAME 'PrefilterFunct1@myPreFilterlib;Unix:PrefilterFunct2@myPreFilterlib' は、Windows の場合は myPreFilterlib.dll から PrefilterFunct1 ファンクションを呼び出し、UNIX の場合は myPreFilterlib.so から PrefilterFunct2 ファンクションを呼び出します。

DROP PREFILTER 句

DROP PREFILTER 句は、テキスト設定オブジェクトのために指定された事前フィルタライブラリの使用を削除するために使用します。このテキスト設定オブジェクトを使用するインデックスをデータベースサーバが構築すると、事前フィルタは実行されなくなります。

SAVE OPTION VALUES 句

テキスト設定オブジェクトが作成されると、現在の date_format、time_format、timestamp_format、timestamp_with_time_zone_format の各データベースオプションには、DATE、TIME、TIMESTAMP の各カラムがテキスト設定オブジェクトにどのように保存されているかが反映されます。SAVE OPTION VALUES 句は、テキスト設定オブジェクトに保存されたオプションを値を更新して、接続に現在有効なオプションを反映するために使用します。

備考

単語長の設定を変更する前に、インデックス化の対象とクエリ単語の解釈に対する各種設定による影響に関する説明をお読みください。

テキストインデックスは、テキスト設定オブジェクトに依存しています。この文を使用する前に、依存する AUTO テキストインデックスまたは MANUAL REFRESH テキストインデックスをトランケートし、IMMEDIATE REFRESH テキストインデックスを削除してください。

テキスト設定オブジェクトの設定を表示するには、SYSTEXTCONFIG システムビューを問い合わせます。

権限

テキスト設定オブジェクトの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- テキスト設定オブジェクトに対する ALTER 権限
- ALTER ANY TEXT CONFIGURATION システム権限
- ALTER ANY OBJECT システム権限

外部単語区切りを変更する場合は、CREATE EXTERNAL REFERENCE システム権限も必要です。

事前フィルタを指定または解除したり、外部単語区切りを指定する場合は、ALTER ANY TEXT CONFIGURATION または ALTER ANY OBJECT システム権限も必要です。

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、テキスト設定オブジェクト myTextConfig を作成し、最大単語長を 16 に変更します。この分を実行するには、CREATE TEXT CONFIGURATION 権限が必要です。

```
CREATE TEXT CONFIGURATION myTextConfig FROM default_char;
ALTER TEXT CONFIGURATION maxTerm16
   MAXIMUM TERM LENGTH 16;
```

次の文は、myTextConfig 設定オブジェクトにストップリストを追加します。

```
ALTER TEXT CONFIGURATION myTextConfig
```

```
   STOPLIST 'because about therefore only';
```

次の文は、myTextConfig テキスト設定オブジェクトの外部単語区切りを設定します。Windows と UNIX の両方のインタフェースが指定されています。

```
ALTER TEXT CONFIGURATION myTextConfig
   TERM BREAKER GENERIC
   EXTERNAL NAME
'my_termbreaker@termbreaker.dll;Unix:my_termbreaker@libtermbreaker_r.so'
```

次の例は、myTextConfig テキスト設定オブジェクトの外部事前フィルタを設定します。Windows と UNIX の両方のインタフェースが指定されています。

```
ALTER TEXT CONFIGURATION myTextConfig
   PREFILTER EXTERNAL NAME
'html_xml_filter@html_xml_filter.dll;UNIX:html_xml_filter@libhtml_xml_filter_r.so'
;
```

次の例は、myTextConfig テキスト設定オブジェクトの外部事前フィルタを削除します。

```
ALTER TEXT CONFIGURATION myTextConfig DROP PREFILTER;
```

関連情報

CREATE TEXT CONFIGURATION 文 [975 ページ]
DROP TEXT CONFIGURATION 文 [1076 ページ]
sa_char_terms システムプロシージャ [1452 ページ]
sa_nchar_terms システムプロシージャ [1574 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]
sa_text_index_stats システムプロシージャ [1641 ページ]
SYSTEXTCONFIG システムビュー [1849 ページ]

1.4.4.29 ALTER TEXT INDEX 文

テキストインデックスの定義を変更します。

構文

```
ALTER TEXT INDEX [ owner.]text-index-name
ON [ owner.]table-name
alter-clause
```

```
alter-clause :
rename-object
| refresh-alteration
```

```
rename-object :
```

```
RENAME { AS | TO } new-name
```

```
refresh-alteration :
{ MANUAL REFRESH
| AUTO REFRESH [ EVERY integer { MINUTES | HOURS } ] }
```

パラメータ

RENAME 句

RENAME 句は、テキストインデックスの名前を変更するときに使用します。

REFRESH 句

REFRESH 句は、テキストインデックスの再表示タイプを設定するときに指定します。

備考

いったん作成したテキストインデックスは、IMMEDIATE REFRESH に変更したり、IMMEDIATE REFRESH から変更したりできません。いずれかの変更が必要な場合は、テキストインデックスを削除して再度作成する必要があります。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

マテリアライズドビューで作成されたテキストインデックスは、名前を変更することのみ可能です。マテリアライズドビューで作成されたテキストインデックスの再表示タイプは変更できません。

権限

テーブルのテキストインデックスを変更するには、テーブルの所有者であるか、次のいずれかの権限を持っている必要があります。

- そのテーブルに対する REFERENCES 権限
- ALTER ANY INDEX システム権限
- ALTER ANY OBJECT システム権限

マテリアライズドビューのテキストインデックスを変更するには、マテリアライズドビューの所有者であるか、次のいずれかの権限を持っている必要があります。

- ALTER ANY INDEX システム権限
- ALTER ANY OBJECT システム権限

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

最初の文は、テキストインデックス txt_index_manual を作成し、MANUAL REFRESH として定義します。2 つ目の文は、テキストインデックスを変更して毎日自動的に再表示します。3 つ目の文は、テキストインデックスの名前を txt_index_daily に変更します。

```
CREATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation ( Description )
   MANUAL REFRESH;
ALTER TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
   AUTO REFRESH EVERY 24 HOURS;
ALTER TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
   RENAME AS txt_index_daily;
```

関連情報

CREATE TEXT INDEX 文 [976 ページ]
DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
COMMENT 文 [769 ページ]

1.4.4.30 ALTER TIME ZONE 文

タイムゾーンオブジェクトを削除します。

構文

```
ALTER TIME ZONE name time-zone-option [ ... ]
```

```
time-zone-option :
 { OFFSET offset
 | DST OFFSET offset
 | NO DST
```

```
  | [ DST ] STARTING month-day-rule AT { minutes | hours:minutes }
  | [ DST ] ENDING month-day-rule AT { minutes | hours:minutes } }
```

備考

すべての句はオプションですが、何も指定しないと文が失敗します。

以前は夏時間がなかったタイムゾーンに夏時間を追加するには、STARTING 句および ENDING 句を使用します。タイムゾーンから夏時間を削除するには、NO DST 句を指定して STARTING 句および ENDING 句を省略します。

権限

MANAGE TIME ZONE システム権限または ALTER ANY OBJECT システム権限が必要です。

関連する動作

オートコミット。

この文を実行すると、ISYSTIMEZONE システムテーブルへの移植が行われます。

例

夏時間を使用している東部標準時というタイムゾーンを夏時間のなりオーストラリア東部標準時変更するには、次の文を実行します。

```
ALTER TIME ZONE EasternTime OFFSET '10:00'
NO DST;
```

関連情報

COMMENT 文 [769 ページ]
CREATE TIME ZONE 文 [979 ページ]
DROP TIME ZONE 文 [1078 ページ]
SYSTIMEZONE システムビュー [1851 ページ]

1.4.4.31 ALTER TRACE EVENT SESSION 文

トレースイベントのセッションへの追加およびセッションからの削除、ターゲットのセッションへの追加およびセッションからの削除、またはトレースセッションの開始および停止を実行します。

構文

```
ALTER TRACE EVENT SESSION session-name
[ ON SERVER ]
{ ADD TRACE EVENT trace-event-name [,...]
  | DROP TRACE EVENT trace-event-name [,...]
  | ADD TARGET FILE [ ( SET target-parameter-name=target-parameter-value [, ...] ) ]
  | DROP TARGET target-name [, ...] ]
  | STATE = { START | STOP }
}
```

```
target-parameter-name :
{ filename_prefix
  | max_size
  | num_files
  | flush_on_write
  | compressed }
```

パラメータ

ON SERVER 句

データベースサーバ上のすべてのデータベースのトレースイベントを記録しているトレースイベントセッションが変更されます。この句を指定しない場合、現在のデータベース上のトレースイベントセッションが変更されます。

ADD TRACE EVENT

セッションに追加されるトレースイベントの名前。

DROP TRACE EVENT

セッションから削除されるトレースイベントの名前。

ADD TARGET

セッションに追加されるターゲットの名前。セッションの一部であるトレースイベントに関する情報は、このターゲット (ファイル) に記録されます。

DROP TARGET

セッションから削除されるターゲットの名前。

備考

実行中のトレースセッションでトレースイベントやターゲットを追加または削除すると、セッションが一時的に停止して変更が行われ、変更が終わると再開されます。トレースイベントやターゲットを追加または削除するためにトレースセッションを停止する必要はありません。すでに開始されているセッションでトレースイベントを追加または削除すると、セッションが一時的に停止している間にいくつかのトレースイベントが失われる副作用があります。

システム権限

MANAGE ANY TRACE SESSION システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、my_event というトレースイベントを作成し、my_session というトレースイベントセッションを作成して、セッションを開始します。

```
CREATE TEMPORARY TRACE EVENT my_event( id INTEGER, information LONG VARCHAR );
CREATE TEMPORARY TRACE EVENT SESSION my_session
    ADD TRACE EVENT my_event, -- user event
    ADD TRACE EVENT SYS_ConsoleLog_Information -- system event
    ADD TARGET FILE ( SET filename_prefix='my_trace_file' ); -- add a target
ALTER TRACE EVENT SESSION my_session
     STATE = START;
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.4.4.32 ALTER TRIGGER 文

トリガの定義を修正したものに置き換えます。新しいトリガの定義全体を ALTER TRIGGER 文にインクルードします。この文は、SAP IQ のカタログストアテーブルのみに適用されます。

構文

トリガ定義の変更

```
ALTER TRIGGER trigger-name trigger-definition
```

```
trigger-definition :CREATE TRIGGER 構文
```

トリガ定義の難読化

```
ALTER TRIGGER trigger-name ON [owner.] table-name SET HIDDEN
```

備考

トリガ定義の変更

ALTER TRIGGER 文の構文は、最初の 1 語を除き、CREATE TRIGGER 文の構文とまったく同じです。

Transact-SQL 形式と Watcom-SQL 形式のどちらの CREATE TRIGGER 構文も使用できます。

トリガ定義の難読化

SET HIDDEN を使用して、関連するトリガの定義を難読化し、解読できないようにします。このトリガはアンロードして、他のデータベースに再ロードできます。SET HIDDEN を使用すると、デバッガを使用したデバッグでも、プロシージャプロファイリングによっても、トリガ定義は表示されません。

注記

SET HIDDEN 操作は元に戻せません。

権限

基本となるテーブルの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- 基本となるテーブルに対する ALTER 権限と CREATE ANY OBJECT システム権限
- ALTER ANY TRIGGER システム権限
- ALTER ANY OBJECT システム権限

別のユーザが所有するビューのトリガを変更するには、ALTER ANY TRIGGER および ALTER ANY VIEW システム権限のどちらかを持っているか、ALTER ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE TRIGGER 文 [982 ページ]
CREATE TRIGGER 文 [T-SQL] [989 ページ]
DROP TRIGGER 文 [1082 ページ]

1.4.4.33 ALTER USER 文

ユーザ設定を変更します。

構文

データベースユーザ定義の変更

```
ALTER USER user-name
  [ IDENTIFIED BY password ]
  [ LOGIN POLICY policy-name ]
  [ FORCE PASSWORD CHANGE { ON | OFF } ]
```

データベースユーザのロック解除

```
ALTER USER user-name
[ RESET LOGIN POLICY ]
```

LDAP ユーザの識別名 (DN) のリフレッシュ

```
ALTER USER user-name
REFRESH DN
```

パスワードの部分の変更

```
ALTER USER user-name
  [ IDENTIFIED { FIRST | LAST } BY password-part ]
```

パラメータ

user-name

ユーザの名前。

IDENTIFIED 句

ユーザのパスワード。パスワードのないユーザは、データベースに接続できません。これは、グループを作成し、他のユーザはグループユーザ ID を使用してデータベースに接続させないようにする場合に便利です。

IDENTIFIED BY 句

この句を使用して、ユーザのパスワードをリセットします。パスワードを持たないユーザをリセットするには、`password` を NULL に設定します。

IDENTIFIED { FIRST | LAST } BY 句

この句を使用して、二重制御パスワードを持っているユーザが持つパスワードの部分をリセットします。二重制御パスワードを持っているユーザのログインポリシーでは、CHANGE_PASSWORD_DUAL_CONTROL ログインポリシーオプションが有効になっています。

二重制御パスワードをリセットするには 2 名の管理者が必要です。1 人の管理者が IDENTIFIED FIRST BY 句を実行してパスワードの前半部分を設定し、もう 1 人の管理者が IDENTIFIED LAST BY 句を実行してパスワードの後半部分を設定します。ユーザは、パスワードの 2 つの部分を組み合わせ、この組み合わせたパスワードを使用してデータベースに接続します。

policy-name

ユーザに割り当てるログインポリシーの名前。LOGIN POLICY 句が指定されていない場合、変更は行われません。

FORCE PASSWORD CHANGE 句

ログイン時にユーザが新しいパスワードを指定する必要があるかどうかを制御します。この設定は、ユーザのポリシーの password_expiry_on_next_login オプション設定を上書きします。

RESET LOGIN POLICY 句

失敗ログインの試行回数と、ユーザの最終ログイン時刻および最終ログイン失敗時刻をリセットします。ログインポリシー制限を超えたためにユーザアカウントがロックされている場合は、ロック解除されます。

REFRESH DN 句

REFRESH DN は、ユーザの識別名 (DN) およびタイムスタンプをクリアして、次回の LDAP 認証のときに DN の検索が行われるようにします。このユーザの次回の LDAP 認証時に認証が成功した場合、DN とタイムスタンプの両方が新しい DN と現在時刻に更新されます。

備考

次のリストは、ユーザ ID とパスワードの要件を示しています。パスワードの一部の要件や制限は、各部の最大長が 127 バイトであることを除いて、パスワードに対して示されている要件や制限と同じです。

verify_password_function ログインポリシーオプションは、パスワード規則 (1 桁以上の長さであることなど) の実装に使用できる関数を指定します。パスワード検証機能を使用する場合、GRANT CONNECT 文に複数のユーザ ID とパスワードを指定することはできません。

password_expiry_on_next_login 値を ON に設定すると、ユーザが同じポリシーに割り当てられている場合でも、次回ログインすると同時にユーザのパスワードの有効期限が切れます。ALTER USER 句と LOGIN POLICY 句を使用すると、ユーザが次にログインしたときにパスワードの変更を強制できます。

ALTER USER...REFRESH DN 構文は、ユーザの識別名 (DN) およびタイムスタンプをクリアして、次回の LDAP 認証のときに、古くなっている可能性のあるキャッシュ済みの DN を使用せずに、DN の検索が行われるようにします。認証が成功した場合、DN とタイムスタンプの両方が新しい DN と現在時刻に更新されます。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

どのユーザも自分のパスワードを変更できます。

別のユーザのパスワードを変更するには、CHANGE PASSWORD システム権限が必要です。

ユーザに各自のパスワード変更を強制するなど、別のユーザに対するその他のすべての変更には、MANAGE ANY USER システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SQLTester という名前のユーザに対し、パスワードを welcome123、ログインポリシーを Test1、強制的なパスワード変更を回避できるよう設定します。

```
ALTER USER SQLTester IDENTIFIED BY welcome123
LOGIN POLICY Test1
FORCE PASSWORD CHANGE off;
```

次の例は、ユーザ myusername の LDAP 識別名をリフレッシュします。

```
ALTER USER myusername REFRESH DN;
```

関連情報

ALTER LOGIN POLICY 文 [657 ページ]
COMMENT 文 [769 ページ]
CREATE LOGIN POLICY 文 [852 ページ]
CREATE USER 文 [990 ページ]
DROP LOGIN POLICY 文 [1043 ページ]
DROP USER 文 [1084 ページ]
GRANT CONNECT 文 [1139 ページ]

1.4.4.34 ALTER VIEW 文

ビュー定義を修正したものに置き換えます。

構文

ビュー定義の変更

```
ALTER VIEW
[ owner.]view-name [ ( column-name, ... ) ] AS query-expression
[ WITH CHECK OPTION ]
```

ビューの属性の変更

```
ALTER VIEW
[ owner.]view-name { SET HIDDEN | RECOMPILE | DISABLE | ENABLE }
```

パラメータ

AS 句

ビューのベースとなる SELECT 文です。SELECT 文はローカルテンポラリテーブルを参照してはなりません。また、`query-expression` には GROUP BY 句、HAVING 句、WINDOW 句、または ORDER BY 句を指定でき、UNION、EXCEPT、INTERSECT または共通テーブル式を含めることができます。

クエリのセマンティックは、SELECT 文で TOP 句または FIRST 句に ORDER BY 句を組み合わせないと、返されるローの順序が定義されないことを示します。

WITH CHECK OPTION 句

WITH CHECK OPTION 句を指定すると、`query-expression` で定義された条件を満たさないビューへの更新や挿入が拒否されます。

SET HIDDEN 句

SET HIDDEN 句は、ビューと句の定義を難読化し、このビューを *SQL Central* などのビューから見えないようにします。ビューの明示的な参照は機能します。

> **注記**
>
> SET HIDDEN 操作は元に戻せません。

RECOMPILE 句

RECOMPILE 句は、ビューのカラム定義を再作成するときに使用します。この句は、無効にされていないビューに使用できるという点を除き、ENABLE 句の機能と同じです。ビューが再コンパイルされると、データベースサーバは、新しいビュー定義に指定されたカラム名に基づいて、カラム権限をリストアします。カラムが再コンパイル後に存在しない場合、既存の権限は失われます。

DISABLE 句

DISABLE 句は、データベースサーバからビューを使用できないようにします。

ENABLE 句

ENABLE 句は無効にしたビューを有効にするときに使用します。ビューを有効にすると、データベースサーバでビューのカラム定義が再作成されます。依存しているビューを有効にしてから、ビューを有効にします。

備考

`query-expression` では、ORDER BY 句がなくても、TOP n、FIRST、または LIMIT の句を指定できます。最大で指定数のローが返りますが、返ってくるローの順序は定義されません。そのため、ORDER BY は指定できますが、必須ではありません。クエリにビューが使用されている場合、たとえ FROM 句内に他のテーブルがなくても、ビュー内の ORDER BY は、クエリ内のローの順序に作用しません。このため、ORDER BY をビューに入れるのは、TOP n、FIRST、または LIMIT の句に含めるローを選択する必要がある場合だけにしてください。それ以外の場合、ORDER BY 句に効果はなく、データベースサーバはこの句を無視します。

1 つ以上の INSTEAD OF トリガがあるビューで ALTER VIEW 文を実行すると、エラーが返されます。トリガを削除してからでないと、ビューを削除または変更できません。

別のユーザが所有するビューを変更する場合は、所有者を含めて名前を修飾する必要があります (たとえば、GROUPO.ViewSalesOrders)。名前を修飾しなかった場合、データベースサーバは、ユーザ本人が所有する同名のビューを検索して変更します。見つからない場合は、エラーが返されます。

ビューを変更しても、ビューに対する既存の権限はそのまま維持されます。このため、権限の再割り当ては必要ありません。ALTER VIEW 文を使用する代わりに、DROP VIEW 文でビューを削除してから CREATE VIEW 文でビューを再作成することもできます。ただし、削除してから再作成する場合、ビューの権限を再割り当てする必要があります。

クエリでは、ORDER BY 句がなくても、TOP n、FIRST、または LIMIT の句を指定できます。最大で指定数のローが返りますが、返ってくるローの順序は定義されません。そのため、ORDER BY は指定できますが、必須ではありません。クエリにビューが使用されている場合、たとえ FROM 句内に他のテーブルがなくても、ビュー内の ORDER BY は、クエリ内のローの順序に作用しません。このため、ORDER BY をビューに入れるのは、TOP n、FIRST、または LIMIT の句に含めるローを選択する必要がある場合だけにしてください。それ以外の場合、ORDER BY 句に効果はなく、データベースサーバはこの句を無視します。

ビュー定義を変更する構文を使用したビュー変更を完了した後に、データベースサーバはビューを再コンパイルします。変更の種類によっては、従属ビューがあれば、データベースサーバは従属ビューも再コンパイルしようとします。従属ビューに影響を及ぼす変更を加える場合、従属ビューの定義も変更が必要なことがあります。

⚠ 警告

ビューを定義する SELECT 文にアスタリスク (*) が含まれていた場合、ビュー内のカラム数は、基本となるテーブルでカラムが追加または削除された場合に変化することがあります。ビューカラムの名前とデータ型も変化する場合があります。

ビュー定義の変更

この構文はビューの構造を変更するときに使用します。変更が各カラムに制限されるテーブルを変更する場合とは異なり、ビュー構造の変更には、新しい定義で全体のビュー定義を置換する必要があります。ビューを新規作成するときと同様です。

ビューの属性の変更

この構文は、ビュー定義を非表示にするかどうかなど、ビューの属性を変更するときに使用します。

SET HIDDEN を使用すると、ビューをアンロードしてから他のデータベースにリロードすることができます。SET HIDDEN を使用すると、デバッガを使用したデバッグでも、プロシージャプロファイリングによっても、ビュー定義は表示されません。非表示のビュー定義を変更するには、ビューを削除してから、CREATE VIEW 文を使用して再作成します。

DISABLE 句を使用すると、データベースサーバがクエリに応答するときにビューを使用できなくなります。ビューの無効化は削除と似ていますが、無効化はビュー定義がデータベースに残る点が異なります。ビューを無効にすると、従属ビューも無効になります。DISABLE 句によって従属ビューも無効になるため、無効化されるビューだけでなく、すべての従属ビューに対する排他アクセスが必要です。

権限

そのビューの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- ALTER ANY VIEW システム権限
- ALTER ANY OBJECT システム権限

そのビューが利用する基本オブジェクトから選択する権限も必要です。

関連する動作

オートコミット。

すべてのプロシージャとトリガがメモリからアンロードされるため、元のビューを参照するプロシージャやトリガには新しいビュー定義が反映されます。ビューを頻繁に変更すると、プロシージャとトリガのアンロードとロードによってパフォーマンスが低下することがあります。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE VIEW 文 [995 ページ]
DROP VIEW 文 [1087 ページ]
CREATE MATERIALIZED VIEW 文 [854 ページ]
ALTER MATERIALIZED VIEW 文 [659 ページ]

1.4.4.35 ATTACH TRACING 文 (廃止)

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。ATTACH TRACING 文は、診断トレースセッションを開始します (診断テーブルへの診断情報の送信を開始します)。

構文

```
ATTACH TRACING TO { LOCAL DATABASE | connect-string }
  [ LIMIT { size | history } ]
```

```
connect-string: データ ベースの接続文字列
```

```
size : SIZE nnn { MB | GB }
```

```
history : HISTORY nnn { MINUTES | HOURS | DAYS }
```

```
nnn : integer
```

パラメータ

connect-string

トレーシング情報を受信するデータベースに接続するときに必要な接続文字列。このパラメータは、プロファイルされているデータベースがデータを受信するデータベースと異なる場合にのみ必要です。

DBF、DBKEY、DBN、Host、Server、LINKS、PWD、UID の接続パラメータは、`connect-string` で割り当てられます。

DBF は、接続するデータベースサーバに対する相対パスで指定します。別のデータベースサーバを指定しない場合、現在接続しているデータベースサーバが、DBF 接続パラメータで指定したトレーシングデータベースの起動を試みます。

DBF パラメータと接続パラメータ LINKS または Server を一緒に指定すると、エラーが返されます。

LIMIT 句

トレーシングデータベースに格納されているデータサイズの制限。サイズまたは期間で指定します。

備考

ATTACH TRACING 文は、プロファイルするデータベースのトレースセッションを開始するときに使用します。この文はトレーシングレベルを設定した後にのみ使用できます。sa_set_tracing_level システムプロシージャを使用してトレーシングレベルを設定できます。

セッションを開始すると、sa_diagnostic_tracing_level テーブルに設定したトレーシングレベルに応じたトレーシング情報が生成されます。LOCAL DATABASE を指定して、プロファイル対象の同じデータベース内にあるトレーシングテーブルにトレーシングデータを送信できます。または、接続文字列をそのデータベースに指定することで (`connect-string`)、トレーシングデータを別のトレーシングデータベースに送信できます。トレーシングデータベースはあらかじめ作成されている必要があります。また、トレーシングデータベースへのパーミッションも必要です。

LIMIT SIZE 句または LIMIT HISTORY 句を使用して、トレーシングデータの格納サイズを制限できます。トレーシングデータを特定のサイズに制限するときは、LIMIT SIZE 句を使用します。単位は MB または GB です。期間を指定してトレーシングデータのサイズを制限するときは、LIMIT HISTORY 句を使用します。単位は分、時間、日です。たとえば、`HISTORY 8 DAYS` と指定すると、トレーシングデータベースにトレーシングデータが格納される期間は 8 日間に制限されます。

トレースセッションを開始するには、トレーシングデータベースと運用データベースが稼働しているデータベースサーバで TCP/IP が実行されている必要があります。

ローカルデータベースに対するトレーシングの場合でも、機密データを含むパケットはネットワークインタフェース上に表示されます。セキュリティ上の理由から、接続文字列に暗号化を指定できます。

現在のトレーシングレベルを参照するには、sa_diagnostic_tracing_level テーブルを参照します。

トレーシングデータの送信先を表示するには、SendingTracingTo データベースプロパティを確認します。

この文を実行するには、プロファイルされるデータベースに接続している必要があります。

権限

DIAGNOSTICS システムロールと MANAGE PROFILING システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、sa_set_tracing_level システムプロシージャを使用して、トレーシングレベルを 1 に設定します。次に、トレースセッションを開始します。ローカルデータベースで生成されたトレーシングデータは、別のコンピュータにある mytracingdb トレーシングデータベースに送信されます。指定した接続文字列を参照してください。トレースセッション中に、最長 2 時間のトレーシングデータが保持されます。

```
CALL sa_set_tracing_level( 1 );
ATTACH TRACING TO
'UID=DBA;PWD=passwd;Server=remotedbsrv;DBN=mytracingdb;Host=mytracing-pc'
  LIMIT HISTORY 2 HOURS;
```

関連情報

DETACH TRACING 文 (廃止) [1027 ページ]
REFRESH TRACING LEVEL 文 (廃止予定) [1258 ページ]
sa_diagnostic_tracing_level テーブル (廃止予定) [1426 ページ]
sa_set_tracing_level システムプロシージャ (廃止予定) [1627 ページ]

1.4.4.36 BACKUP DATABASE 文

データベースとトランザクションログをバックアップします。

構文

イメージバックアップ

```
BACKUP DATABASE
DIRECTORY backup-directory
[ backup-option [  backup-option ... ] ]
```

```
backup-directory : { string | variable }
```

```
backup-option :
WAIT BEFORE START
| WAIT AFTER END
| DBFILE ONLY
| TRANSACTION LOG ONLY
| TRANSACTION LOG RENAME [ MATCH ]
| TRANSACTION LOG TRUNCATE
| ON EXISTING ERROR
| WITH COMMENT comment-string
| HISTORY { ON | OFF }
| AUTO TUNE WRITERS { ON | OFF }
| WITH CHECKPOINT LOG { AUTO | COPY | NO COPY | RECOVER }
```

アーカイブバックアップ

```
BACKUP DATABASE TO archive-root
```

```
[ backup-option [  backup-option ... ] ]
```

```
archive-root :  { string | variable }
```

```
backup-option :
WAIT BEFORE START
| WAIT AFTER END
| DBFILE ONLY
| TRANSACTION LOG ONLY
| TRANSACTION LOG RENAME [ MATCH ]
| TRANSACTION LOG TRUNCATE
| ATTENDED { ON | OFF }
| WITH COMMENT comment-string
| HISTORY { ON | OFF }
| WITH CHECKPOINT LOG [ NO ] COPY
| MAX WRITE { number-of-writers | AUTO }
| FREE PAGE ELIMINATION { ON | OFF }
```

```
comment-string : string
```

```
number-of-writers : integer
```

パラメータ

DIRECTORY 句

バックアップファイルを作成するディスク上のロケーション。データベースサーバ起動時の現在のディレクトリを基準に指定します。このディレクトリが存在しない場合は作成されます。ディレクトリとして空の文字列を指定すると、最初にトランザクションログをコピーしないで名前を変更したり、トランケートしたりできます。データベースミラーリングを使用している場合は、この句を使用しないでください。

WAIT BEFORE START 句

この句は、アクティブなトランザクションがなくなるまでバックアップを遅らせます。データベースに対する他のすべてのアクティビティが抑制され、チェックポイントが実行されます。

この句を WITH CHECKPOINT LOG NO COPY 句と一緒に使用して、データベースのバックアップコピーをリカバリする必要がないことを確認し、データベースのバックアップコピーを読み込み専用モードで開始し、検証できるようにします。バックアップデータベースを検証するときに、データベースの追加のコピーを作成する必要はありません。

WAIT AFTER END 句

この句を指定すると、すべてのトランザクションが完了してから、トランザクションログの名前変更またはトランケートが行われます。データベースサーバは、他の接続で実行中のトランザクションがすべてコミットまたはロールバックするまで待機してから、バックアップを完了します。この句は、新しい入力トランザクションがバックアップを無期限に待機させることがあるため、注意して使用してください。

DBFILE ONLY 句

この句は、メインデータベースファイルと、トランザクションログ以外のすべての関連 DB 領域について、バックアップコピーを作成します。DBFILE ONLY 句は、TRANSACTION LOG RENAME 句または TRANSACTION LOG TRUNCATE 句と同時に使用できません。

TRANSACTION LOG ONLY 句

TRANSACTION LOG ONLY 句を指定すると、他のデータベースファイルをコピーしないで、トランザクションログのバックアップコピーを作成できます。

TRANSACTION LOG RENAME [MATCH] 句

この句を指定すると、データベースサーバによって、現在のトランザクションログの名前が `YYMMDDnn.log` 形式のファイル名に変更され、データベースファイルと同じ名前を持つ新しいトランザクションログが起動されます。バックアップトランザクションログの名前は、MATCH が指定されていない限り、アクティブなトランザクションログと同じ名前になります。MATCH が指定されている場合、トランザクションログのバックアップコピーは、名前変更されたファイル (`YYMMDDnn.log`) と同じ名前になります。MATCH キーワードを指定すると、古いデータを上書きしないで同じ文を 2 度以上実行できます。

バックアップを完了しないでトランザクションログの名前を変更して再起動するには、TRANSACTION LOG ONLY 句で空のディレクトリ名を指定します。例:

```
BACKUP DATABASE DIRECTORY ''
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME;
```

TRANSACTION LOG TRUNCATE 句

この句を指定すると、バックアップの完了時に現在のトランザクションログはトランケートされて再起動します。データベースミラーリングを使用している場合は、この句を使用しないでください。

バックアップを完了しないでトランザクションログをトランケートするには、TRANSACTION LOG ONLY 句で空のディレクトリ名を指定します。例:

```
BACKUP DATABASE DIRECTORY ''
TRANSACTION LOG ONLY
TRANSACTION LOG TRUNCATE;
```

ON EXISTING ERROR 句

この句は、イメージバックアップにだけ適用されます。デフォルトでは、既存のファイルは BACKUP DATABASE 文を実行したときに上書きされます。この句が使用されている場合は、バックアップによって作成されるファイルのいずれかがすでに存在するとエラーが発生します。

WITH COMMENT 句

この句は、バックアップ履歴ファイルにコメントを記録します。アーカイブバックアップの場合、コメントはアーカイブファイルにも記録されます。

HISTORY 句

この句は、バックアップ履歴を有効または無効にします。デフォルトでは、この句は ON であり、バックアップ操作ごとに `backup.syb` ファイルに行が追加されます。HISTORY OFF を指定すると、`backup.syb` ファイルの更新が抑制されるため、次の場合に推奨されます。

- データベースが頻繁にバックアップされます。
- `backup.syb` ファイルを定期的にアーカイブまたは削除するプロシージャがありません。
- ディスク領域が限られています。

AUTO TUNE WRITERS 句

この句を指定すると、ライタの自動チューニングが有効または無効になります。バックアッププロセスの間、1 つのライタがバックアップファイルをバックアップディレクトリに書き込みます。バックアップディレクトリが、ライタの負荷の増加に対処できるデバイス (RAID アレイなど) にある場合は、デフォルトの AUTO TUNE WRITERS ON によって、ライタの数を増やすことで全体のバックアップパフォーマンスが改善します。データベースサーバは、バックアップに参加しているすべての

デバイスについて、読み込みと書き込みのパフォーマンスを定期的に検証します。AUTO TUNE WRITERS OFF を指定すると、データベースサーバがそれ以上ライタを作成しなくなります。

WITH CHECKPOINT LOG 句

この句では、バックアップ先のディレクトリに書き込む前にバックアップでデータベースファイルに対してどのような処理を実行するかを指定します。バックアップ中に更新前イメージを適用するか、チェックポイントログをバックアップとしてコピーするかを選択します。どちらを選択するかによって、パフォーマンスに違いが生じます。デフォルト設定は、イメージバックアップの場合は AUTO で、アーカイブバックアップの場合は COPY です。

COPY 句

このオプションを BACKUP DATABASE 文で WAIT BEFORE START 句と併用することはできません。

COPY を指定すると、バックアップは、変更されたページを適用しないでデータベースファイルを読み込みます。チェックポイントログ全体とシステム DB 領域がバックアップディレクトリにコピーされます。このデータベースサーバを次回起動すると、データベースは自動的にバックアップ開始時のチェックポイントの状態にリカバリされます。

このオプションを使用すると、ページをテンポラリファイルに書き込む必要がないため、バックアップパフォーマンスが向上し、バックアップ中に動作している他の接続との内部サーバ競合が減少します。ただし、チェックポイントログには変更したページの元のイメージが含まれるため、データベースが更新されるとサイズが増えます。コピーを指定すると、データベースファイルのバックアップコピーが、バックアップ開始時のデータベースファイルよりも大きくなる可能性があります。COPY オプションは、バックアップ先ディレクトリのディスク領域が十分である場合に使用してください。

NO COPY 句

NO COPY を指定すると、チェックポイントログはバックアップの一部としてコピーされません。この場合、変更されたページはテンポラリファイルに保存され、バックアップの実行中、バックアップに適用されます。データベースファイルのバックアップコピーは、バックアップを開始した時点のデータベースと同じサイズになります。

このオプションを指定すると、データベースファイルのバックアップは小さくなりますが、バックアップの速度が遅くなり、データベースサーバで実行される他の操作のパフォーマンスが低下することがあります。バックアップ先のドライブの空き領域が少ない場合に、このオプションが役に立ちます。

RECOVER 句

RECOVER を指定すると、(COPY オプションを指定した場合と同じように) チェックポイントログがコピーされますが、このチェックポイントログはバックアップの完了時にデータベースに適用されます。したがって、データベースファイルのバックアップは、バックアップ操作を開始した時点と同じ状態 (および同じサイズ) となります。このオプションが役に立つのは、バックアップドライブの空き領域が少ない場合です (チェックポイントログをバックアップする COPY オプションの場合と同じ量の空き領域が必要ですが、バックアップファイルのサイズは COPY オプションを指定した場合より小さくなります)。

AUTO 句

AUTO を指定すると、データベースサーバはバックアップディレクトリがあるボリュームの空きディスク領域をチェックします。バックアップを開始する時点でデータベースサイズの 2 倍以上の空きディスク領域がある場合、このオプションは COPY を指定した場合と同様に動作します。それ以外の場合、NO COPY の場合と同様に動作します。AUTO がデフォルトの動作です。

MAX WRITE 句

アーカイブバックアップの場合、デフォルトで、バックアップファイルの書き込み専用にスレッドが 1 つ割り当てられます。バックアップディレクトリが、ライターの負荷の増加に対処できるデバイス (RAID アレイなど) 上にある場合、ライターとして動作するスレッド数を増やすことで全体のバックアップパフォーマンスを改善できます。

AUTO を指定すると、各リーダースレッドに出力ストリームが 1 つ作成されます。値 n には、作成可能な出力ストリームの最大数を、リーダースレッド数を上限として指定します。この句のデフォルト値は 1 です。

最初のストリームであるストリーム 0 が myarchive.X という名前のファイルを生成します。X は 1 から始まる番号で、必要なファイルの数まで 1 ずつ増えていきます。それ以外のすべてのストリームは myarchive.Y.Z という名前のファイルを生成します。Y は 1 から始まるストリーム番号です。Z は 1 から始まる番号で、必要なファイル数まで 1 ずつ増えていきます。

FREE PAGE ELIMINATION 句

アーカイブバックアップでは、デフォルトで空きページがスキップされるため、バックアップが小規模で高速になる可能性があります。トランザクションログファイルには空きページがないため、空きページが削除されてもトランザクションログファイルのバックアップには効果がありません。大規模なトランザクションログファイルを使用するデータベースでは、小規模なトランザクションログファイルを使用するデータベースと比較して、空きページを削除するメリットがあまりないこともあります。

空きページの削除を有効にして強力に暗号化されているデータベースをバックアップした場合は、データベースをリストアするときに暗号化キーを指定してください。空きページの削除を無効にして強力に暗号化されているデータベースをバックアップした場合は、データベースをリストアするときに暗号化キーを指定する必要はありません。

バージョン 12 では、バージョン 11 以前のデータベースサーバで作成したアーカイブバックアップをリストアできません。

備考

BACKUP DATABASE 文は、サーバ側のバックアップを実行します。クライアント側のバックアップを実行するには、dbbackup ユーティリティを使用します。

ディスクサンドボックス機能がデータベースで有効になっている場合は、データベースサーバがサンドボックス外部のディレクトリ (メインデータベースファイルがあるディレクトリと、このディレクトリのすべてのサブディレクトリ) にバックアップを作成できるように、ディスクサンドボックス機能を無効にするセキュリティ機能キーを指定する必要があります。

バックアップ操作では、イメージかアーカイブかに関係なく、履歴ファイル backup.syb が更新されます。backup.syb ファイルには、特定のデータベースサーバで実行された BACKUP 操作と RESTORE 操作が記録されます。

リカバリを実行しないで読み込み専用データベースサーバで開始できるバックアップを作成するには、WAIT BEFORE START 句と WITH CHECKPOINT LOG NO COPY 句を両方とも使用する必要があります。WAIT BEFORE START 句はロールバックログを空にし、WITH CHECKPOINT LOG NO COPY 句はチェックポイントログを空にします。どちらのファイルが欠けても、リカバリが必要になります。バックアップしたデータベースをリカバリする必要がない場合は、WAIT BEFORE START 句と WITH CHECKPOINT LOG NO COPY 句の代わりに、WITH CHECKPOINT LOG RECOVER を使用できます。

構文 - イメージバックアップ

イメージバックアップでは、各データベースファイルのコピーが、バックアップユーティリティ (dbbackup) の場合と同じ方法で作成されます。デフォルトでは、バックアップユーティリティはクライアントコンピュータにバックアップを作成しますが、-s オプションを使用すると、バックアップユーティリティの使用時にデータベースサーバ上にバックアップを作成することもできます。一方、BACKUP DATABASE 文の場合、バックアップはデータベースサーバ上にのみ作成できます。

オプションで、データベースファイルまたはトランザクションログのどちらかだけを保存できます。また、トランザクションログは、バックアップの完了後に名前を変更するか、またはトランケートすることもできます。

または、ディレクトリとして空の文字列を指定して、ログをコピーしないで名前を変更したり、トランケートしたりできます。これは、スペースが問題となるレプリケーション環境で便利です。この機能をトランザクションログサイズのイベントハンドラ

と一緒に使用すると、トランザクションログが所定のサイズに達したときに名前を変更したり、delete_old_logs オプションと一緒に使用して、トランザクションログが必要なくなったときに削除したりできます。

イメージバックアップからリストアするには、保存されたファイルを元のロケーションにコピーして、トランザクションログを再度適用します。
構文 - アーカイブのバックアップ

アーカイブのバックアップでは、必要なバックアップ情報をすべて保持する 1 つのファイルが作成されます。

アーカイブのバックアップからデータベースをリストアするには、RESTORE DATABASE 文を使用します。

RESTORE DATABASE 文が、トランザクションログのみを含むアーカイブファイルを参照している場合、リストアされるデータベースファイルのロケーションにファイルが存在しない場合でも、文ではそのファイル名を指定します。たとえば、トランザクションログだけを含むアーカイブからディレクトリ `C:¥MYNEWDB` に復元する場合、RESTORE DATABASE 文は次のようになります。

```
RESTORE DATABASE 'c:¥¥temp¥¥mynewdb¥my.db' FROM archive-root
```

> ⚠ 警告
>
> データベースとトランザクションログのバックアップコピーには、どのような変更でも加えるべきではありません。バックアップ中に処理中のトランザクションがなかった場合、または BACKUP DATABASE WITH CHECKPOINT LOG RECOVER か WITH CHECKPOINT LOG NO COPY を指定した場合は、読み込み専用モードを使用するか、バックアップデータベースのコピーを検証することによって、バックアップデータベースの妥当性をチェックできます。
>
> 一方、トランザクションの処理中だった場合、または BACKUP DATABASE WITH CHECKPOINT LOG COPY を指定した場合は、検証の開始時にデータベースサーバがデータベースのリカバリを実行する必要が生じます。リカバリを実行するとバックアップコピーに変更が加えられますが、これは望ましいことではありません。

この文の実行中に、進行状況メッセージの表示を要求できます。

また、Progress 接続プロパティを使用して、文がどの程度実行されたかを確認することもできます。

権限

BACKUP DATABASE システム権限が必要です。

関連する動作

チェックポイントを発生させます。

標準

ANSI/ISO SQL 標準

標準になし。

例

現在のデータベースとトランザクションログをそれぞれ別のファイルにバックアップし、既存のトランザクションログ名を変更します。イメージバックアップが作成されます。

```
BACKUP DATABASE
DIRECTORY 'c:¥¥temp¥¥backup'
TRANSACTION LOG RENAME;
```

トランザクションログの名前を変更するオプションは、古いトランザクションログが引き続き必要になるレプリケーション環境で特に役立ちます。

コピーを作成しないでトランザクションログの名前を変更します。

```
BACKUP DATABASE DIRECTORY ''
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME;
```

動的に構成されたディレクトリ名を指定して、BACKUP DATABASE 文を実行します。

```
CREATE EVENT NightlyBackup
SCHEDULE
START TIME '23:00' EVERY 24 HOURS
HANDLER
BEGIN
    DECLARE dest LONG VARCHAR;
    DECLARE day_name CHAR(20);

    SET day_name = DATENAME( WEEKDAY, CURRENT DATE );
    SET dest = 'd:¥¥backups¥¥' || day_name;
            BACKUP DATABASE DIRECTORY dest
    TRANSACTION LOG RENAME;
END;
```

関連情報

RESTORE DATABASE 文 [1269 ページ]

1.4.4.37 BEGIN 文

複合文を指定します。

構文

```
[ statement-label : ]
BEGIN [ [ NOT ] ATOMIC ]
   [ local-declaration; ... ]
   statement-list
   [ EXCEPTION [ exception-case ... ] ]
END [ statement-label ]
```

```
local-declaration :
variable-declaration
| cursor-declaration
| exception-declaration
| temporary-table-declaration
```

```
exception-case :
WHEN exception-name [, ... ] THEN statement-list
| WHEN OTHERS THEN statement-list
```

variable-declaration および exception-declaration: DECLARE 文を参照

cursor-declaration: DECLARE CURSOR 文を参照

temporary-table-declaration: DECLARE LOCAL TEMPORARY TABLE 文を参照

パラメータ

statement-label

終了の statement-label を指定する場合は、開始の statement-label と一致させる必要があります。LEAVE 文を使うと、複合文に続く最初の文から実行を再開できます。プロシージャまたはトリガの本文である複合文は、プロシージャまたはトリガの名前と同じ暗黙のラベルを持っています。

ATOMIC clause

"アトミック" な文とは、完全に実行されるか、まったく実行されない文です。たとえば、何千ものローを挿入する UPDATE 文では、数多くのローの更新後にエラーが発生することがあります。文が完了しないと、すべての変更内容が元の状態に戻ります。同様に、BEGIN 文を ATOMIC と指定すると、この文は完全に実行されるか、あるいはまったく実行されないかのどちらかです。

local-declaration

BEGIN のすぐ後で、複合文は複合文の中にだけ存在するオブジェクトをローカルに宣言できます。複合文は、変数、カーソル、テンポラリテーブル、または例外に対してローカル宣言を行います。ローカル宣言は、複合文またはその中でネストされる複合文の中のどの文からでも参照できます。複合文のローカル宣言は、文の例外ハンドラに表示されます。ローカル宣言は、複合文の中から呼び出される他のプロシージャには表示されません。

備考

プロシージャまたはトリガの本文は複合文です。複合文は、プロシージャまたはトリガ内の制御文の中でも使用できます。

複合文によって、1 つまたは複数の SQL 文をまとめて、1 つの単位として扱うことができます。複合文はキーワード BEGIN で始まり、キーワード END で終わります。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

複合文を識別する BEGIN は、オプションの ANSI/ISO SQL 言語機能 P002 の一部です。

例

プロシージャまたはトリガの本文は複合文です。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
   DECLARE err_notfound EXCEPTION FOR
      SQLSTATE '02000';
   DECLARE curThisCust CURSOR FOR
      SELECT CompanyName, CAST(
            sum( SalesOrderItems.Quantity *
            Products.UnitPrice ) AS INTEGER) VALUE
      FROM GROUPO.Customers
            LEFT OUTER JOIN SalesOrders
            LEFT OUTER JOIN SalesOrderItems
            LEFT OUTER JOIN Products
      GROUP BY CompanyName;
   DECLARE ThisValue INT;
   DECLARE ThisCompany CHAR( 35 );
   SET TopValue = 0;
   OPEN curThisCust;
   CustomerLoop:
   LOOP
      FETCH NEXT curThisCust
         INTO ThisCompany, ThisValue;
      IF SQLSTATE = err_notfound THEN
         LEAVE CustomerLoop;
      END IF;
      IF ThisValue > TopValue THEN
         SET TopValue = ThisValue;
         SET TopCompany = ThisCompany;
      END IF;
   END LOOP CustomerLoop;
   CLOSE curThisCust;
END;
```

下の例は、次の変数を宣言しています。

- INT、初期設定値 5 の v1。

- CHAR(10)、初期値 abc の v2 と v3。

```
BEGIN
   DECLARE v1 INT = 5
   DECLARE v2, v3 CHAR(10) = 'abc'
            // ...
END
```

関連情報

DECLARE 文 [1010 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
DECLARE LOCAL TEMPORARY TABLE 文 [1007 ページ]
CONTINUE 文 [778 ページ]
SIGNAL 文 [SP] [1332 ページ]
RESIGNAL 文 [SP] [1268 ページ]
RAISERROR 文 [1246 ページ]
BEGIN 文 [TSQL] [748 ページ]

1.4.4.38 BEGIN 文 [TSQL]

複合文を指定します。

構文

```
BEGIN
statement-list
END
```

```
statement-list :
sql-statement
| variable-declaration
| cursor-declaration
| temporary-table-declaration
```

```
variable-declaration: DECLARE 文を参照
```

```
cursor-declaration: DECLARE CURSOR 文を参照
```

```
temporary-table-declaration: DECLARE LOCAL TEMPORARY TABLE 文を参照
```

パラメータ

statement-list

文および宣言のリスト

備考

BEGIN 文によって、1 つまたは複数の SQL 文をまとめて、1 つの単位として扱い、キーワード BEGIN で開始して、キーワード END で終了することができます。

Transact-SQL 複合文ではエラー処理が異なります。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

複合文を識別する BEGIN は、オプションの ANSI/ISO SQL 言語機能 P002 の一部です。

例

下の例は、次の変数を宣言しています。

- INT、初期設定値 5 の v1。
- CHAR(10)、初期値 abc の v2 と v3。

```
BEGIN
   DECLARE v1 INT = 5
   DECLARE v2, v3 CHAR(10) = 'abc'
            // ...
END
```

関連情報

DECLARE 文 [1010 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
DECLARE LOCAL TEMPORARY TABLE 文 [1007 ページ]
BEGIN 文 [745 ページ]
CONTINUE 文 [778 ページ]
GOTO 文 [1129 ページ]
RAISERROR 文 [1246 ページ]

1.4.4.39 BEGIN PARALLEL WORK 文

複数の論理プロセッサを使用することで、コンピュータ上で CREATE INDEX または LOAD TABLE 文のリストを実行するときの時間を短縮します。

構文

```
BEGIN PARALLEL WORK
statement-list
END PARALLEL WORK
```

```
statement-list:
  list of CREATE INDEX statements
| list of LOAD TABLE statements
```

パラメータ

CREATE INDEX statement BEGIN PARALLEL WORK 文内で CREATE INDEX 文を実行することに関する情報や制限については、"CREATE INDEX 文" を参照してください。CREATE TEXT INDEX 文はサポートされていません。

LOAD TABLE statement

BEGIN PARALLEL WORK 文内で LOAD TABLE 文を実行することに関する情報や制限については、"LOAD TABLE 文" を参照してください。

備考

BEGIN PARALLEL WORK 文は、CREATE INDEX 文のリストまたは LOAD TABLE 文のリストを並列で実行することで、パフォーマンスを向上させることができます。

同時に実行できる文の数は、次の条件によって決まります。

- データベースサーバが実行されているコンピュータ上で使用可能な論理プロセッサの数。
- sa_server_option システムプロシージャにおける、-gtc および -gta データベースサーバオプションやプロセッサアフィニティオプションの設定。
- max_parallel_statements オプションの設定。

BEGIN PARALLEL WORK 文はアトミックに実行されます。BEGIN PARALLEL WORK 文内のいずれかの文が失敗すると、文全体がロールバックされます。BEGIN PARALLEL WORK 文は、その内部にリストされた文を含め、トランザクションログに記録されます。

細分化レベルはテーブルレベルです。

権限

リスト内の文を実行するための権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準 標準になし。ベンダ拡張。

例

次の例は、BEGIN PARALLEL WORK 文を使用して、2 つのインデックスを作成します。

```
BEGIN PARALLEL WORK
    CREATE INDEX L_SHIPDATE_IDX
            ON LINEITEM(l_shipdate);
    CREATE INDEX O_ORDERDATE_IDX
            ON ORDERS(o_orderdate);
END PARALLEL WORK;
```

次の例は、BEGIN PARALLEL WORK 文を使用して、データベースに 3 つのテーブルをロードします。

```
BEGIN PARALLEL WORK
    LOAD TABLE dba.Part
        FROM 'D:¥¥data¥¥part.tbl'
        FORMAT 'ASCII'
        QUOTES OFF ESCAPES ON STRIP OFF HEXADECIMAL OFF
        DELIMITED BY '|'
        ORDER OFF;
    LOAD TABLE dba.Supplier
        FROM 'D:¥¥data¥¥supplier.tbl'
        FORMAT 'ASCII'
        QUOTES OFF ESCAPES ON STRIP OFF HEXADECIMAL OFF
        DELIMITED BY '|'
        ORDER OFF;
```

```
    LOAD TABLE dba.Partsupp
        FROM 'D:¥¥data¥¥partsupp.tbl'
        FORMAT 'ASCII'
        QUOTES OFF ESCAPES ON STRIP OFF HEXADECIMAL OFF
        DELIMITED BY '|'
        ORDER OFF;
END PARALLEL WORK;
```

関連情報

CREATE INDEX 文 [842 ページ]
LOAD TABLE 文 [1182 ページ]

1.4.4.40 BEGIN TRANSACTION 文 [T-SQL]

ユーザ定義のトランザクションを開始します。

構文

```
BEGIN TRAN[SACTION] [ transaction-name ]
```

備考

オプションのパラメータ `transaction-name` はこのトランザクションに割り当てられた名前です。有効な識別子を指定します。トランザクション名はネストされた BEGIN/COMMIT または BEGIN/ROLLBACK 文の最も外側の組でのみ使用してください。

BEGIN TRANSACTION 文をトランザクション内で実行すると、トランザクションのネストレベルが 1 つ増加します。ネストレベルは COMMIT 文で減少します。トランザクションがネストされているときは、最も外側の COMMIT だけがデータベースへの変更を保存します。

Adaptive Server Enterprise と SQL Anywhere には、それぞれ 2 つのトランザクションモードがあります。

Adaptive Server Enterprise のデフォルトのトランザクションモードは、"非連鎖モード" と呼ばれ、明示的な BEGIN TRANSACTION 文が実行されてトランザクションを開始しないかぎり、各文を個々にコミットします。反対に、ANSI/ISO SQL 標準互換の連鎖モードは、明示的 COMMIT が実行されるときか、オートコミットする文 (データ定義文など) が実行されるときにのみトランザクションをコミットします。

chained データベースオプションを設定してモードを制御できます。SQL Anywhere の ODBC 接続と Embedded SQL 接続のデフォルト設定が ON の場合、データベースサーバは連鎖モードで実行されます (ODBC を使用している場合、AutoCommit ODBC 設定も確認が必要です)。TDS 接続のデフォルトは Off です。

非連鎖モードでは、トランザクションはデータ検索文やデータ操作文の前に暗黙的に開始されます。これらの文には、次のようなものがあります。DELETE、INSERT、OPEN、FETCH、SELECT、UPDATE が挙げられます。ただし、COMMIT 文または ROLLBACK 文を使用してトランザクションを明示的に終了してください。

トランザクション内では chained オプションの設定を変更できません。

> ⚠ 警告
>
> ストアドプロシージャを呼び出すときは、目的のトランザクションモードで正しく動作することを確認してください。

現在のネストレベルはグローバル変数 @@trancount に入っています。@@trancount 変数の値は、最初の BEGIN TRANSACTION 文が実行される前は 0 であり、@@trancount の値が 1 のときに実行された COMMIT だけがデータベースへの変更を永続的なものにすることができます。

@@trancount の値は、実行された明示的な BEGIN TRANSACTION 文を数える以上の目的には使用しないでください。

Adaptive Server Enterprise が暗黙的にトランザクションを起動する場合、@@trancount 変数を 1 に設定します。トランザクションが暗黙的に起動された場合、SQL Anywhere は @@trancount 値を 1 に設定しません。代わりに、SQL Anywhere の @@trancount 変数は、BEGIN TRANSACTION 文が始まる前は (現在のトランザクションがある場合でも) 0 の値を持ちますが、Adaptive Server Enterprise (連鎖モード) では、この変数は 1 の値を持ちます。

BEGIN TRANSACTION 文で開始するトランザクションでは、最初の BEGIN TRANSACTION 文の後、@@trancount の値は SQL Anywhere と Adaptive Server Enterprise の両方で 1 になります。トランザクションが異なる文で暗黙的に起動し、その後、BEGIN TRANSACTION 文が実行された場合、BEGIN TRANSACTION 文の後、@@trancount の値は、SQL Anywhere と Adaptive Server Enterprise の両方で 2 になります。

トランザクションまたはセーブポイント名のない ROLLBACK 文は、常に文を最も外側の BEGIN TRANSACTION (明示的または暗黙的) 文にロールバックし、トランザクション全体をキャンセルします。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

BEGIN TRANSACTION は Adaptive Server Enterprise によってサポートされます。

例

次のバッチは、@@trancount の値を、連続した値 0、1、2、1、0 で報告します。値はデータベースサーバのメッセージウィンドウに出力されます。

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT
PRINT @@trancount
COMMIT
PRINT @@trancount
```

関連情報

COMMIT 文 [771 ページ]
ROLLBACK 文 [1284 ページ]
SAVEPOINT 文 [1290 ページ]

1.4.4.41 BEGIN SNAPSHOT 文

スナップショットアイソレーショントランザクションで使用するスナップショットを指定の期間に開始します。

構文

BEGIN SNAPSHOT

備考

デフォルトでは、トランザクションが始まると、アプリケーションによってテーブルの最初のローがフェッチされるまで、データベースサーバはスナップショットの作成を遅延します。BEGIN SNAPSHOT 文を使用すると、トランザクション内で事前にスナップショットを開始できます。スナップショットトランザクションによって BEGIN SNAPSHOT 文が実行されると、データベースサーバはスナップショットを作成します。

次の条件に一致する場合、文は失敗し、エラーを返します。

- データベースに対してスナップショットトランザクションのサポートが有効にされていない。
- 現在のトランザクションに対してスナップショットがすでに開始されている。

この文は非スナップショットトランザクションにも有効です。これは、この文を使用すると、文レベルのスナップショット操作のためにトランザクションで後から使用できるスナップショットを、非スナップショットトランザクションで開始できるためです。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.42 BREAK 文 [T-SQL]

複合文またはループから出ます。

構文

BREAK

備考

BREAK 文は、ループから出るための制御文です。実行はループの後に記述されている最初の文から再開されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、BREAK 文は、最も高い製品の価格が 50 ドルを超える場合、WHILE ループをブレークします。そうでない場合、ループは平均価格が 30 ドル以上になるまで継続します。

```
WHILE ( SELECT AVG( UnitPrice ) FROM Products ) < $30
BEGIN
   UPDATE GROUPO.Products
   SET UnitPrice = UnitPrice + 2
   IF ( SELECT MAX(UnitPrice) FROM Products ) > $50
      BREAK
END
```

関連情報

WHILE 文 [T-SQL] [1408 ページ]
CONTINUE 文 [778 ページ]
BEGIN 文 [745 ページ]

1.4.4.43 CALL 文

プロシージャを呼び出します。

構文

位置による引数の指定

```
[variable = ] CALL procedure-name ( [ expression, ... ] ) [ AS USER { string | variable } IDENTIFIED BY { string | variable } ]
```

キーワードフォーマットによる引数の指定

```
[variable = ] CALL procedure-name ( [ parameter-name = expression, ... ] ) [ AS USER { string | variable } IDENTIFIED BY { string | variable } ]
```

パラメータ

AS USER ...IDENTIFIED BY 句 このオプション句は、別のユーザとしてプロシージャまたは関数を呼び出します。データベースサーバは、入力されたユーザ ID とパスワードが有効であることを確認し、指定されたユーザでプロシージャまたは

関数を実行します。プロシージャの呼び出し元は指定されたユーザです。プロシージャまたは関数の終了時に、ユーザコンテキストが元の状態にリストアされます。

> **注記**
>
> 文字列値はすべて一重引用符で囲みます。それ以外の場合は、データベースサーバによって変数名として解釈されます。

備考

CALL 文は、CREATE PROCEDURE 文を使ってあらかじめ作成されたプロシージャを呼び出します。プロシージャが完了すると、INOUT または OUT パラメータ値がコピーし直されます。

> **注記**
>
> AS USER ... IDENTIFIED BY 句は CALL 文のみに適用され、FROM 句のプロシージャや select リストの関数ではサポートされていません。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

プロシージャの呼び出し時に、データベーススコープ変数を INOUT または OUT パラメータとして指定することはできません。

引数リストは、引数の順序を守って指定するか、キーワードフォーマットを使うことにより指定できます。順序を守って指定する場合、引数はプロシージャのパラメータリストの順番に指定します (オプションのパラメータには DEFAULT を使用できます)。キーワードを使用する場合、引数名にはパラメータ名を使用します。

プロシージャ引数には、CREATE PROCEDURE 文のデフォルト値が割り当てられます。パラメータが見つからない場合は、デフォルト値が割り当てられます。また、デフォルトが設定されていない場合、パラメータが NULL に設定されます。デフォルトが設定されていないうえ、引数も指定されていない場合には、エラーが発生します。

プロシージャの内部では、プロシージャが結果セットを返す場合、DECLARE 文で CALL 文を使用できます。

サブクエリと空間メソッド呼び出しは、CALL 文のストアドプロシージャに対する引数としては使用できません。

プロシージャは RETURN 文を使用して整数値 (たとえば、ステータスインジケータとして) を返すことができます。変数の戻り値を保存するには、割り当て演算子として等号を使います。

呼び出されたプロシージャが INT を返し、値が NULL の場合は、代わりにエラーステータス値 0 が返されます。この場合と実際の値の 0 が返される場合とを区別する方法はありません。

> **注記**
>
> この文を使用した関数の呼び出しは廃止される予定です。呼び出す関数がある場合は、代入文を使用して関数を呼び出し、その結果を変数に代入することを検討してください。次に例を示します。

```
DECLARE varname INT;
SET varname=test( );
```

権限

プロシージャを呼び出すユーザまたは AS USER ... IDENTIFIED BY 句で指定されたユーザは、そのプロシージャの所有者であるか、次のいずれかの権限を持っていることが必要です。

- プロシージャに対する EXECUTE 権限
- EXECUTE ANY PROCEDURE システム権限

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。ストアドプロシージャから値を返すための RETURN 文の使用は、標準にありません。ANSI/ISO SQL 標準では、プロシージャの戻り値ではなく、SQL 呼び出し関数の戻り値のみがサポートされます。ストアドプロシージャの引数に対するデフォルト値は、標準にありません。

例

次の例は、指定された ID を持つ顧客からの注文の数を返すプロシージャを作成し、結果を保持する変数を作成後、そのプロシージャを呼び出して結果を表示します。

```
CREATE PROCEDURE OrderCount ( IN customer_ID INT, OUT Orders INT )
BEGIN
   SELECT COUNT( GROUPO.SalesOrders.ID )
   INTO Orders
   FROM GROUPO.Customers
   KEY LEFT OUTER JOIN SalesOrders
   WHERE Customers.ID = customer_ID;
END
go
 -- Create a variable to hold the result
CREATE VARIABLE Orders INT
go
-- Call the procedure, FOR customer 101
CALL OrderCount ( 101, Orders )
go
--  Display the result
SELECT Orders FROM SYS.DUMMY
go
```

次の例は、AS USER...IDENTIFIED BY 句を使用して、別のユーザとしてプロシージャを実行します。

1. 次の文を実行し、サンプルユーザを 3 つ作成します。

```
CREATE USER u IDENTIFIED BY pwdforu;
CREATE USER u1 IDENTIFIED BY pwdforu1;
CREATE USER u2 IDENTIFIED BY pwdforu2;
```

2. 次の文を実行し、2 つのテーブルを作成および移植します。

```
CREATE TABLE u1.t1 (c1 INT);
CREATE TABLE u2.t2 (c1 INT);
INSERT INTO u1.t1 VALUES(1);
INSERT INTO u2.t2 VALUES(2);
COMMIT;
```

3. 次の文を実行し、2 つのストアドプロシージャを作成します。

```
CREATE PROCEDURE u1.p1( OUT ret INT, OUT inv_user CHAR(128), OUT exec_user
CHAR(128) )
SQL SECURITY INVOKER
BEGIN
   SELECT c1 INTO ret FROM t1;
   SET inv_user = invoking_user;
   SET exec_user = executing_user;
END;
```

```
CREATE PROCEDURE u2.p2( OUT ret INT, OUT inv_user CHAR(128), OUT exec_user
CHAR(128) )
SQL SECURITY DEFINER
BEGIN
   SELECT c1 INTO ret FROM t2;
   SET inv_user = invoking_user;
   SET exec_user = executing_user;
END;
```

4. AS USER...IDENTIFIED BY 句の文字列値を使用して、ユーザ u1 として 1 番目のプロシージャを呼び出す 3 番目のプロシージャを作成します。

```
CREATE PROCEDURE u.p1( OUT ret INT, OUT inv_user CHAR(128), OUT exec_user
CHAR(128) )
SQL SECURITY DEFINER
BEGIN
   CALL u1.p1 ( ret, inv_user, exec_user ) AS USER 'u1' IDENTIFIED BY
'pwdforu1';
END;
```

5. AS USER...IDENTIFIED BY 句の変数値を使用して、ユーザ u として 2 番目のプロシージャを呼び出す 4 番目のプロシージャを作成します。

```
CREATE PROCEDURE u.p2( IN u CHAR(128), IN p CHAR(128), OUT ret INT, OUT
inv_user CHAR(128), OUT exec_user CHAR (128) )
SQL SECURITY DEFINER
BEGIN
   CALL u2.p2( ret, inv_user, exec_user ) AS USER u IDENTIFIED BY p;
END;
```

6. ユーザ u がログインして `CALL u1.p1( ret, inv_user, exec_user )` または `CALL u2.p2( ret, inv_user, exec_user )` を実行すると、ユーザ u は u1.p1 または u2.p2 を実行する権限がないため、ユーザ u のパーミッションは拒否されます。ただし、ユーザ u は次のプロシージャを実行することができます。

```
CALL u1.p1( ret, inv_user, exec_user ) AS USER 'u1' IDENTIFIED BY 'pwdforu1';
```

u1.p1 は SQL SECURITY INVOKER プロシージャですが、inv_user および exec_user の両方が u1 として返されます。これは、ユーザがユーザ u としてログインしても、データベースサーバがユーザ u1 として u1.p1 を実行するためです。プロシージャは u1 が呼び出しているように実行するため、u1.t1 が存在するので完全に修飾されてはいないテーブル t1 にアクセスできます。

同様に、u は次のプロシージャを実行することができます。

```
CALL u2.p2( ret, inv_user, exec_user ) AS USER uvar IDENTIFIED BY pvar;
```

ここで、uvar は値 'u2' を含む変数で、pvar は値 'pwdforu2' を含む変数です。プロシージャがエラーなく実行されて inv_user および exec_user の両方が u2 として返り、プロシージャは完全に修飾されてはいないテーブル t2 にアクセスできます。

7. ユーザ u が次の 2 つのプロシージャを実行した場合、データベースサーバは他のストアドプロシージャの中でネストされた AS USER...IDENTIFIED BY 句を受け入れるため、両方の呼び出しが成功します。

```
CALL u.p1( ret, inv_user, exec_user );
```

```
CALL u.p2( 'u2', 'pwdforu2', ret, inv_user, exec_user );
```

関連情報

CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
EXECUTE 文 [T-SQL] [1096 ページ]

1.4.4.44 CASE 文

複数の状況に基づいて実行パスを選択します。

構文

値式の指定

```
CASE value-expression
WHEN [ constant | NULL ] THEN statement-list ...
[ WHEN [ constant | NULL ] THEN statement-list ] ...
[ ELSE statement-list ]
END [ CASE ]
```

検索条件の指定

```
CASE
WHEN [ search-condition | NULL] THEN statement-list ...
[ WHEN [ search-condition | NULL] THEN statement-list ] ...
[ ELSE statement-list ]
END [ CASE ]
```

備考

値式を使用した CASE 文

CASE 文は制御文であり、これを使用して SQL 文のリストから式の値に対応する文を選択して実行できます。`value-expression` は、文字列、数値、日付、その他の SQL データ型などの単一の値を取る式です。`value-expression` の値に対して WHEN 句が存在する場合、WHEN 句の `statement-list` が実行されます。適切な WHEN 句が存在せず、ELSE 句が存在する場合、ELSE 句の `statement-list` が実行されます。END CASE の後に記述されている最初の文から実行が再開されます。

`value-expression` が null でよい場合は、ISNULL 関数を使用して NULL の `value-expression` を異なる式で置き換えます。

検索条件を使用した CASE 文

このフォームの文は、CASE 文中で最初に条件と一致した `search-condition` に対して実行されます。どの `search-conditions` も一致しなかった場合は、ELSE 句が実行されます。

NULL が許容される式の場合は、最初の `search-condition` に次の構文を使用します。

```
WHEN search-condition IS NULL THEN statement-list
```

注記

CASE 文の構文と CASE 式の構文を混同しないでください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

CASE 文は、言語機能 P002 (Computational completeness) の一部です。END CASE ではなく END の単独使用は標準にありません。

Transact-SQL

CASE 文は Adaptive Server Enterprise によってサポートされます。

例

CASE 文を使用する次のプロシージャは、サンプルデータベースの Products テーブルにリストされている製品を、シャツ、帽子、ショートパンツ、不明のいずれかに分類します。

```
CREATE PROCEDURE ProductType (IN product_ID INT, OUT type CHAR(10))
BEGIN
   DECLARE prod_name CHAR(20);
   SELECT Name INTO prod_name FROM GROUPO.Products
   WHERE ID = product_ID;
   CASE prod_name
   WHEN 'Tee Shirt' THEN
      SET type = 'Shirt'
   WHEN 'Sweatshirt' THEN
      SET type = 'Shirt'
   WHEN 'Baseball Cap' THEN
      SET type = 'Hat'
   WHEN 'Visor' THEN
      SET type = 'Hat'
   WHEN 'Shorts' THEN
      SET type = 'Shorts'
   ELSE
      SET type = 'UNKNOWN'
   END CASE;
END;
```

次の例は、検索条件を使用して、SQL Anywhere サンプルデータベース内の製品数量に関するメッセージを生成します。

```
CREATE PROCEDURE StockLevel (IN product_ID INT)
BEGIN
   DECLARE qty INT;
   SELECT Quantity INTO qty FROM GROUPO.Products
   WHERE ID = product_ID;
   CASE
   WHEN qty < 30 THEN
      MESSAGE 'Order Stock' TO CLIENT;
   WHEN qty > 100 THEN
      MESSAGE 'Overstocked' TO CLIENT;
   ELSE
      MESSAGE 'Sufficient stock on hand' TO CLIENT;
   END CASE;
END;
```

関連情報

ISNULL 関数 [その他] [410 ページ]
BEGIN 文 [745 ページ]
CASE 式 [36 ページ]
CASE 文 [T-SQL] [763 ページ]

1.4.4.45 CASE 文 [T-SQL]

複数の状況に基づいて実行パスを選択します。

構文

値式の指定

```
CASE value-expression
WHEN [ constant | NULL ] THEN statement-list ...
[ WHEN [ constant | NULL ] THEN statement-list ] ...
[ ELSE statement-list ]
END
```

検索条件の指定

```
CASE
WHEN [ search-condition | NULL] THEN statement-list ...
[ WHEN [ search-condition | NULL] THEN statement-list ] ...
[ ELSE statement-list ]
END
```

備考

値式の使用

CASE 文は制御文であり、これを使用して SQL 文のリストから式の値に対応する文を選択して実行できます。`value-expression` は、文字列、数値、日付、その他の SQL データ型などの単一の値を取る式です。`value-expression` の値に対して WHEN 句が存在する場合、WHEN 句の `statement-list` が実行されます。適切な WHEN 句が存在せず、ELSE 句が存在する場合、ELSE 句の `statement-list` が実行されます。END CASE の後に記述されている最初の文から実行が再開されます。

`value-expression` が null でよい場合は、ISNULL 関数を使用して NULL の `value-expression` を異なる式で置き換えます。

検索条件の使用

このフォームの文は、CASE 文中で最初に条件と一致した `search-condition` に対して実行されます。どの `search-conditions` も一致しなかった場合は、ELSE 句が実行されます。

NULL が許容される式の場合は、最初の `search-condition` に次の構文を使用します。

```
WHEN search-condition IS NULL THEN statement-list
```

注記

CASE 文の構文と CASE 式の構文を混同しないでください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

CASE 文は、言語機能 P002 (Computational completeness) の一部です。ただし、ANSI/ISO SQL 標準では、CASE 文を終了するのに、END 単独ではなく END CASE が必要です。

Transact-SQL

Adaptive Server Enterprise と互換性があります。

例

CASE 文を使用する次のプロシージャは、サンプルデータベースの Products テーブルにリストされている製品を、シャツ、帽子、ショートパンツ、不明のいずれかに分類します。

```
CREATE PROCEDURE DBA.ProductType( @product_ID INTEGER,@TYPE CHAR(10) OUTPUT ) AS
BEGIN
  DECLARE @prod_name CHAR(20)
  SELECT Name INTO @prod_name FROM GROUPO.Products
    WHERE ID = @product_ID
  IF @prod_name
   = 'Tee Shirt'
    SET @TYPE = 'Shirt'
  ELSE IF @prod_name
   = 'Sweatshirt'
    SET @TYPE = 'Shirt'
  ELSE IF @prod_name
   = 'Baseball Cap'
    SET @TYPE = 'Hat'
  ELSE IF @prod_name
   = 'Visor'
    SET @TYPE = 'Hat'
  ELSE IF @prod_name
   = 'Shorts'
    SET @TYPE = 'Shorts'
  ELSE
    SET @TYPE = 'UNKNOWN'
END;
```

次の例は、検索条件を使用して、サンプルデータベース内の製品数量に関するメッセージを生成します。

```
CREATE PROCEDURE DBA.StockLevel( @product_ID INTEGER ) AS
BEGIN
  DECLARe @qty INTEGER
  SELECT Quantity INTO @qty FROM GROUPO.Products
```

```
    WHERE ID = @product_ID
  IF @qty < 30
    MESSAGE 'Order Stock' TO CLIENT
  ELSE IF @qty > 100
    MESSAGE 'Overstocked' TO CLIENT
  ELSE
    MESSAGE 'Sufficient stock on hand' TO CLIENT
END;
```

関連情報

ISNULL 関数 [その他] [410 ページ]
BEGIN 文 [745 ページ]
CASE 式 [36 ページ]

1.4.4.46 CHECKPOINT 文

データベースにチェックポイントを設定します。

構文

```
CHECKPOINT
```

備考

CHECKPOINT 文はデータベースサーバにチェックポイントの実行を強制します。またチェックポイントは、内部アルゴリズムに従ってデータベースサーバによっても自動的に実行されます。通常、アプリケーションが、CHECKPOINT 文を発行する必要はありません。

権限

ネットワークサーバ (dbsrv) 上で実行されているデータベースに対してチェックポイントを実行する場合は、CHECKPOINT システム権限が必要です。

パーソナルデータベースサーバ (dbeng17) 上で実行されているデータベースに対してチェックポイントを実行する場合は、権限は不要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

CHECKPOINT 文は Adaptive Server Enterprise によってサポートされます。

1.4.4.47 CLEAR 文 [Interactive SQL]

Interactive SQL で開いているすべての結果セットを閉じます。

構文

```
CLEAR
```

備考

開いている結果セットをすべて閉じ、[SQL 文] ウィンドウ枠の内容は変更せずに保持します。

権限

なし。

関連する動作

クリアされているデータに関連付けられているカーソルを閉じます。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.48 CLOSE 文 [ESQL] [SP]

カーソルを閉じます。

構文

```
CLOSE cursor-name
```

```
cursor-name : identifier | hostvar
```

備考

この文は指定したカーソルを閉じます。

事前にカーソルを開いておきます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。Embedded SQL で使用される CLOSE 文は、オプション言語機能 B031 (Basic dynamic SQL) の一部です。

Transact-SQL

Adaptive Server Enterprise によってサポートされます。

例

次の例は Embedded SQL のカーソルを閉じます。

```
EXEC SQL CLOSE employee_cursor;
EXEC SQL CLOSE :cursor_var;
```

次のプロシージャはカーソルを使用します。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
   DECLARE err_notfound EXCEPTION
      FOR SQLSTATE '02000';
   DECLARE curThisCust CURSOR FOR
   SELECT CompanyName, CAST(    sum(SalesOrderItems.Quantity *
   Products.UnitPrice) AS INTEGER) VALUE
   FROM GROUPO.Customers
   LEFT OUTER JOIN SalesOrders
   LEFT OUTER JOIN SalesOrderItems
   LEFT OUTER JOIN Products
   GROUP BY CompanyName;
DECLARE ThisValue INT;
   DECLARE ThisCompany CHAR(35);
   SET TopValue = 0;
   OPEN curThisCust;
   CustomerLoop:
   LOOP
      FETCH NEXT curThisCust
      INTO ThisCompany, ThisValue;
         IF SQLSTATE = err_notfound THEN
            LEAVE CustomerLoop;
         END IF;
         IF ThisValue > TopValue THEN
            SET TopValue = ThisValue;
            SET TopCompany = ThisCompany;
         END IF;
      END LOOP CustomerLoop;
   CLOSE curThisCust;
END
```

関連情報

OPEN 文 [ESQL] [SP] [1218 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
PREPARE 文 [ESQL] [1238 ページ]

1.4.4.49 COMMENT 文

データベースオブジェクトに対するコメントをシステムテーブルに格納します。

構文

```
COMMENT ON {
    COLUMN [ owner.]table-name.column-name
  | CERTIFICATE certificate-name
  | DBSPACE dbspace-name
  | EVENT [ owner.]event-name
  | EXTERNAL ENVIRONMENT environment-name
  | EXTERNAL [ ENVIRONMENT ] OBJECT object-name
  | FOREIGN KEY [ owner.]table-name.key-name
  | INDEX [ [ owner.] table.]index-name
  | INTEGRATED LOGIN integrated-login-id
  | JAVA CLASS java-class-name
  | JAVA JAR java-jar-name
  | KERBEROS LOGIN "client-Kerberos-principal"
  | LDAP SERVER ldapua-server-name
  | LOGIN POLICY policy-name
  | MATERIALIZED VIEW [ owner.]materialized-view-name
  | MIRROR SERVER mirror-server-name
  | ODATA PRODUCER name
  | PRIMARY KEY ON [ owner.]table-name
  | PROCEDURE [ owner.]procedure-name
  | PUBLICATION [ owner.] publication-name
  | REMOTE MESSAGE TYPE remote-message-type-name
  | ROLE role-name
  | SEQUENCE sequence-name
  | SERVICE web-service-name
  | SPATIAL REFERENCE SYSTEM srs-name
  | SPATIAL UNIT OF MEASURE uom-identifier
  | SYNCHRONIZATION PROFILE synchronization-profile-name
  | TABLE [ owner.]table-name
  | TEXT CONFIGURATION [ owner.]text-config-name
  | TEXT INDEX text-index-name ON [ owner.]table-name
  | TIME ZONE name
  | TRIGGER [ [ owner.]tablename.]trigger-name
  | USER userid
  | VIEW [ owner.]view-name
}
IS comment
```

```
comment : string | NULL
```

```
environment-name :
JAVA
| PERL
| PHP
| CLR
| C_ESQL32
| C_ESQL64
| C_ODBC32
| C_ODBC64
```

備考

COMMENT 文は、データベース内のオブジェクトに注釈 (コメント) を設定するときに使用します。COMMENT 文を使用すると、ISYSREMARK システムテーブル内の remarks カラムが更新されます。コメントは、NULL に設定すると削除できます。インデックスまたはトリガに対するコメントの場合、コメントの所有者はインデックスまたはトリガが定義されているテーブルの所有者です。

ローカルのテンポラリテーブルにはコメントを追加できません。

データベースドキュメントウィザードを使用してデータベースをドキュメント化する場合、プロシージャ、ファンクション、トリガ、イベント、ビューのコメントを出力に含めるオプションがあります。

権限

COMMENT ANY OBJECT システム権限を持っている場合は、CREATE ANY OBJECT システム権限を使用して作成できるすべてのオブジェクトに対してコメントを追加できます。COMMENT ANY OBJECT システム権限を持っていない場合は、以下に示すような同等の権限が必要です。

- データベースオブジェクトについては、少なくとも次のいずれかの条件に該当する必要があります。
  - オブジェクトを所有していること
  - 他のユーザが所有する同じタイプのオブジェクトを作成または変更できること (CREATE ANY TABLE または ALTER ANY OBJECT など)
  - そのタイプのオブジェクトを管理できること (MANAGE ANY USER など)
- システムロールについては、そのロールに対する管理権限が必要です。
- ユーザ定義ロールについては、MANAGE ROLES システム権限か、またはそのロールに対する管理権限が必要です。
- Kerberos ログインまたは統合化ログインについては、MANAGE ANY USER システム権限が必要です。
- Java クラスまたは JAR については、MANAGE ANY EXTERNAL OBJECT システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

Adaptive Server Enterprise によってサポートされません。

例

次の例は、コメントの追加と削除の方法を示します。

1. Employees テーブルにコメントを追加します。

```
COMMENT ON TABLE GROUPO.Employees
IS 'Employee information';
```

2. Employees テーブルからコメントを削除します。

```
COMMENT
ON TABLE GROUPO.Employees
IS NULL;
```

オブジェクトのコメント設定を表示するには、SELECT 文を使用します。次の文は、SQL Anywhere サンプルデータベースの ViewSalesOrders ビューについてコメント設定を取得します。

```
SELECT remarks
FROM SYSTAB t, SYSREMARK r
WHERE t.object_id = r.object_id
AND t.table_name = 'ViewSalesOrders';
```

関連情報

ALTER ODATA PRODUCER 文 [666 ページ]
DROP ODATA PRODUCER 文 [1050 ページ]

1.4.4.50 COMMIT 文

データベースを永続的に変更したり、ユーザ定義のトランザクションを終了したりします。

構文

コミット処理

```
COMMIT [ WORK ]
```

トランザクションレベルでのコミット

```
COMMIT TRAN[SACTION] [ transaction-name ]
```

パラメータ

transaction-name

このトランザクションに割り当てられた名前です (任意)。有効な識別子を指定します。トランザクション名はネストされた BEGIN/COMMIT または BEGIN/ROLLBACK 文の最も外側の組でのみ使用してください。

次のオプションは、COMMIT 文の動作を制御します。

- cooperative_commit_timeout オプション
- cooperative_commits オプション
- delayed_commits オプション
- delayed_commit_timeout オプション

現在の接続のコミット数を返すには、Commit 接続プロパティを使用できます。

備考

コミット処理

COMMIT 文は、トランザクションを終了し、このトランザクション中で行われたすべての変更内容をデータベースに永続化します。

すべてのデータ定義文で、コミットが自動的に実行されます。詳細については、各 SQL 文の「関連する動作」を参照してください。

データベースサーバが無効な外部キーを検知すると、COMMIT 文は失敗します。この動作によって、無効な外部キーを持つトランザクションは終了できなくなります。通常、外部キー整合性をそれぞれのデータ操作オペレーションごとにチェックします。ただし、データベースオプション wait_for_commit を On に設定しているか、または特定の外部キーを CHECK ON COMMIT 句で定義した場合、データベースサーバは COMMIT 文が実行されるまで整合性検査を遅らせます。

COMMIT 単独での使用は、COMMIT WORK と同じです。

トランザクションレベルでのコミット

BEGIN TRANSACTION 文と COMMIT TRANSACTION 文をペアで使用すると、ネストされたトランザクションを作成できます。ネストされたトランザクションはセーブポイントに似ています。COMMIT 文がネストされたトランザクションの一番外側で実行された場合は、データベースに対する変更が保存されます。トランザクション内で実行する場合、COMMIT TRANSACTION はトランザクションのネストされているレベルを 1 つずつ減らします。トランザクションがネストされているときは、最も外側の COMMIT だけがデータベースへの変更を保存します。

トランザクションレベルでのコミットは、Transact-SQL の拡張機能です。

権限

なし。

関連する動作

WITH HOLD によって開かれたカーソルを除き、すべてのカーソルを閉じます。

この接続で宣言されたテンポラリテーブルのすべてのローを削除します。ただし、宣言に ON COMMIT PRESERVE ROWS が指定されているテーブルは除きます。

データベースでトランザクションログを使用していない場合、COMMIT 操作ごとに暗黙的なチェックポイントが発生します。

標準

ANSI/ISO SQL 標準

コミット処理はコア機能です。トランザクションレベルでのコミットは、Transact-SQL の拡張機能です。

例

次の文は現在のトランザクションをコミットします。

```
COMMIT;
```

次の Transact-SQL バッチは、@@trancount の値を、連続した値 0、1、2、1、0 で報告します。

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
go
```

関連情報

BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
SAVEPOINT 文 [1290 ページ]
BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
PREPARE TO COMMIT 文 [1241 ページ]
ROLLBACK 文 [1284 ページ]

1.4.4.51 CONFIGURE 文 [Interactive SQL]

Interactive SQL のオプションウィンドウを開きます。

構文

CONFIGURE

備考

CONFIGURE 文は、Interactive SQL のオプションウィンドウを開きます。このウィンドウには、すべての Interactive SQL オプションの現在の設定が表示されます。データベースオプションへの変更は、表示も許可もしません。このウィンドウで Interactive SQL を設定できます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

SET OPTION 文 [1310 ページ]

1.4.4.52 CONNECT 文 [ESQL] [Interactive SQL]

データベースへの接続を確立します。

構文

共有メモリ接続

```
CONNECT
[ TO database-server-name ]
```

```
[ DATABASE database-file ]
[ AS connection-name ]
[ USER ] userid [ IDENTIFIED BY password ]
```

```
database-server-name, database-file, connection-name, userid, password :
{ identifier | string | hostvar }
```

TCP/IP 接続

```
CONNECT USING connect-string
```

```
connect-string : { identifier | string | hostvar }
```

パラメータ

AS 句

オプションとして AS 句を指定して、接続に名前を付けることができます。名前を付けると、同じデータベースへの複数の接続、あるいは同じまたは異なるデータベースサーバへの複数の接続が、すべて同時に行えるようになります。それぞれの接続には、固有のトランザクションがあります。たとえば、2 つの異なる接続から同じデータベース内の同じレコードを修正しようとする場合は、トランザクション間でロックの競合が起こることもあります。

TCP/IP 接続の場合、`connect-string` はセミコロンで区切った keyword=value 形式のパラメータ設定リストです。一重引用符で囲んでください。

備考

CONNECT 文は、`database-server-name` で識別されるサーバ上で実行している、`database-file` で識別されるデータベースへの接続を確立します。この文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

共有メモリ接続は、同じコンピュータ上で実行しているデータベースサーバへの接続に対してだけサポートされます。TCP/IP を使用してローカルデータベースサーバに接続したり、別のコンピュータ上で実行しているデータベースサーバに接続したりする場合は、TCP/IP 接続用の構文を使用してください。

Embedded SQL の動作

Embedded SQL では、`database-server-name` を指定しない場合、デフォルトのローカルデータベースサーバ (最初に起動したデータベースサーバ) が使用されます。`database-file` を指定していない場合、指定したサーバ上の最初のデータベースが使用されます。

WHENEVER 文、SET SQLCA 文、一部の DECLARE 文はコードを生成しないので、ソースファイル内の CONNECT 文の前に置いてもかまいません。それ以外の場合は、CONNECT 文が正しく実行されるまで、どのような文も使用できません。

ユーザ ID とパスワードを使って、動的 SQL 文ごとに権限をチェックします。

> **注記**
>
> SQL Anywhere の場合、Embedded SQL では共有メモリ接続のみがサポートされます。Ultra Light の場合、Embedded SQL では共有メモリと TCP/IP 接続の両方を使用できます。

Interactive SQL の動作

CONNECT 文でデータベースまたはサーバを指定しない場合、Interactive SQL はデフォルトのサーバとデータベースには接続しないで、現在のデータベースとの接続を継続します。サーバ名を指定せずに、データベース名を指定する場合、Interactive SQL は現在のサーバの指定したデータベースに接続しようとします。データベース名を指定せずにサーバ名を指定する場合、Interactive SQL は指定したサーバのデフォルトデータベースに接続します。

たとえば、データベースに接続した状態で次のようなバッチを実行すると、同じデータベースに 2 つのテーブルが作成されます。

```
CREATE TABLE t1( c1 int );
CONNECT DBA IDENTIFIED BY passwd;
CREATE TABLE t2 (c1 int );
```

CONNECT 文が正しく実行されるまで、他のデータベース文を使用できません。

Interactive SQL をウィンドウモードで実行している場合、接続パラメータが足りないというプロンプトが表示されます。

Interactive SQL がコマンドプロンプトモード (-nogui はコマンドラインから Interactive SQL を開始したときに指定します) またはバッチモードで実行中の場合、または AS 句を付けずに CONNECT を実行した場合は、無名の接続が開かれます。別の無名の接続がすでに開いている場合には、古い接続は自動的に閉じられます。それ以外の場合は、CONNECT 文を実行しても、既存の接続は閉じられません。

複数の接続も、現在の接続という概念を使って管理されます。接続文が成功した後、新しい接続が現在の接続になります。別の接続に切り替えるには、SET CONNECTION 文を使います。DISCONNECT 文を使って接続を切断することができます。

Interactive SQL に接続するとき、CONNECT [USER] `userid` を指定することは、SETUSER WITH OPTION `userid` 文を実行することと同じです。

Interactive SQL では、接続情報 (データベース名、ユーザ ID、データベースサーバなど) は、[SQL 文] ウィンドウ枠の上のタイトルバーに表示されます。データベースに接続していない場合は、タイトルバーに [未接続] と表示されます。

> **注記**
>
> Interactive SQL では両方の構文が有効です。ただし、Interactive SQL では `hostvar` 引数をサポートしていないことに注意してください。

この SQL 文は SAP HANA データベースではサポートされていません。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

共有メモリ接続は、オプションの ANSI/ISO SQL 言語機能 F771 です。TCP/IP 接続は標準にありません。

Transact-SQL

両方の構文は、Adaptive Server Enterprise でサポートされています。

例

次は、Embedded SQL 内での CONNECT 使用の例です。

```
EXEC SQL CONNECT AS :conn_name
USER :userid IDENTIFIED BY :password;
EXEC SQL CONNECT USER "DBA" IDENTIFIED BY "passwd";
```

次の例は、SQL Anywhere サンプルデータベースが起動していることを前提としています。

Interactive SQL からデータベースに接続します。Interactive SQL がユーザ ID とパスワードを要求するプロンプトを表示します。

```
CONNECT;
```

ユーザ DBA として Interactive SQL からデフォルトデータベースに接続します。Interactive SQL はパスワードを要求するプロンプトを表示します。

```
CONNECT USER "DBA";
```

DBA として Interactive SQL からサンプルデータベースに接続します。

```
CONNECT
TO demo17
USER DBA
IDENTIFIED BY sql;
```

接続文字列を使って Interactive SQL からサンプルデータベースに接続します。

```
CONNECT
USING 'UID=DBA;PWD=sql;DBN=demo';
```

関連情報

GRANT CONNECT 文 [1139 ページ]

DISCONNECT 文 [ESQL] [Interactive SQL] [1028 ページ]
SET CONNECTION 文 [Interactive SQL] [ESQL] [1302 ページ]
SETUSER 文 [1329 ページ]

1.4.4.53 CONTINUE 文

ループを再開します。

構文

```
CONTINUE [ statement-label ]
```

備考

CONTINUE 文は、ループを再開するための制御文です。実行はループの後に記述されている最初の文から継続されます。

Transact-SQL を使用する一連の文の中で CONTINUE を使用する場合は、`statement-label` を使用しないでください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

文ラベルを使用しない CONTINUE は、SAP Adaptive Server Enterprise によってサポートされます。

例

次のフラグメントは、CONTINUE 文がループを再開する方法を示します。この例は、1 ～ 10 の間の奇数を表示します。

```
BEGIN
   DECLARE i INT;
   SET i = 0;
   lbl:
   WHILE i < 10 LOOP
      SET i = i + 1;
      IF mod( i, 2 ) = 0 THEN
         CONTINUE lbl
      END IF;
      MESSAGE 'The value ' || i || ' is odd.' TO CLIENT;
   END LOOP lbl;
END
```

関連情報

LOOP 文 [1203 ページ]
WHILE 文 [T-SQL] [1408 ページ]
FOR 文 [1106 ページ]
BEGIN 文 [745 ページ]

1.4.4.54 CREATE CERTIFICATE 文

指定されたファイルまたは文字列からデータベース内の証明書を追加したり置き換えます。証明書を作成するには、証明書作成ユーティリティ (createcert) を使用します。

構文

```
CREATE [ OR REPLACE ] CERTIFICATE certificate-name
FROM { certificate-string | variable-name | FILE file-name }
```

パラメータ

FROM 句

この句は、証明書が含まれているファイル、文字列、または変数を指定します。

備考

CREATE CERTIFICATE 文は、指定されたファイル、文字列、または変数からデータベース内の証明書を追加したり置き換えます。そのファイル、文字列、または変数には、バイナリ DER フォーマットの証明書か、テキスト PEM フォーマットの証明書が含まれている必要があります。DER フォーマットの証明書は PEM 証明書に変換されて格納されます。

データベースに格納されている証明書は、Web サーバへの安全な HTTPS 接続を確立する Web サービスのプロシージャおよび関数によって使用できます。また、xp_startsmtp システムプロシージャを使用して安全なメッセージを送るために使用することもできます。

追加した証明書は ISYSCERTIFICATE システムテーブルに追加されます。テーブルを表示するには、対応するシステム ビュー SYSCERTIFICATE を使用します。

実際の証明書を作成する場合には CREATE CERTIFICATE は使用されません。その場合は証明書作成ユーティリティ (createcert) を使用します。

権限

MANAGE CERTIFICATES システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、指定された証明書ファイルの内容を使用して、データベース内に mycert という証明書を作成します。

```
CREATE CERTIFICATE mycert
FROM FILE 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥Certificates¥
¥rsaroot.crt';
```

関連情報

DROP CERTIFICATE 文 [1029 ページ]

CREATE PROCEDURE 文 [Web サービス] [878 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
SYSCERTIFICATE システムビュー [1795 ページ]
sa_certificate_info システムプロシージャ [1450 ページ]

1.4.4.55 CREATE DATABASE 文

データベースを作成します。

構文

```
CREATE DATABASE db-filename-string

DBA USER userid-string
DBA PASSWORD password-string
[ create-option ... ]
```

```
create-option :
    [ ACCENT { RESPECT | IGNORE | FRENCH } ]
    [ ASE [ COMPATIBLE ] ]
    [ BLANK PADDING { ON | OFF } ]
    [ CASE { RESPECT | IGNORE } ]
    [ CHECKSUM { ON | OFF } ]
    [ COLLATION collation-label[ ( collation-tailoring-string ) ] ]
    [ DATABASE SIZE size { KB | MB | GB | PAGES | BYTES } ]
    [ ENCODING encoding-label ]
    [ ENCRYPTED [ TABLE ] { algorithm-key-spec | OFF } ]
    [ JCONNECT { ON | OFF } ]
    [ MINIMUM PASSWORD LENGTH positive-integer ]
    [ PAGE SIZE page-size ]
    [ NCHAR COLLATION nchar-collation-label[ ( collation-tailoring-string ) ] ]
    [ SYSTEM PROCEDURE AS DEFINER { ON | OFF } ]
    [ [ TRANSACTION ] LOG { OFF |
                                                    ON [ log-filename-
string ] [ MIRROR mirror-filename-string ] } ]
```

```
page-size :
2048 | 4096 | 8192 | 16384 | 32768
```

```
algorithm-key-spec :
ON
| [ ON ] KEY key [ ALGORITHM AES-algorithm ]
| [ ON ] ALGORITHM AES-algorithm KEY key
| [ ON ] ALGORITHM 'SIMPLE'
```

```
AES-algorithm :
'AES' | 'AES256' | 'AES_FIPS' | 'AES256_FIPS'
```

パラメータ

CREATE DATABASE

db-filename-string、log-filename-string、mirror-filename-string は、いずれもファイル名の前にパスを含めることができます。リテラル文字列なので、一重引用符で囲んでください。

- パスを指定する場合、後に n または x が続くすべての円記号 (¥) は、2 つ重ねます。このようにエスケープすることによって、SQL の文字列のルールに従って、改行文字 (¥n) または 16 進数字 (¥x) として解釈されるのを回避できます。
  このことが重要な場合の例を示します。

  ```
  CREATE DATABASE 'c:¥¥temp¥¥¥x41¥x42¥x43xyz.db'
  DBA USER 'DBA' DBA PASSWORD 'passwd';
  ```

  最初の ¥¥ エスケープシーケンスは円記号を表します。¥x エスケープシーケンスは、それぞれ文字 A、B、C を表します。このファイル名は ABCxyz.db です。

  ```
  CREATE DATABASE 'c:¥temp¥¥nest.db'
  DBA USER 'DBA' DBA PASSWORD 'passwd';
  ```

  ¥n がエスケープシーケンスの改行文字として解釈されないように、円記号を 2 つ連続で使用します。
  常に円記号をエスケープすると安全です。例:

  ```
  CREATE DATABASE 'c:¥¥my_db.db'
  DBA USER 'DBA' DBA PASSWORD 'passwd'
  LOG ON 'e:¥¥logdrive¥¥my_db.log';
  ```

- パスを指定しない場合、または相対パスで指定する場合、データベースファイルはデータベースサーバの作業ディレクトリを基準に作成されます。トランザクションログファイルのパスを指定しないと、データベースファイルと同じディレクトリにファイルが作成されます。データベースファイルとトランザクションログは、コンピュータの別々のディスクに保存してください。
- ファイル拡張子を指定しない場合、データベースファイルは .db 拡張子、トランザクションログは .log 拡張子、トランザクションログミラーは .mlg 拡張子を付けて作成されます。
- ディレクトリパスはデータベースサーバからの相対パスです。

db-filename-string には utility_db を指定できません。この名前は、ユーティリティデータベースのために予約されています。

ACCENT clause

この句は、データベースでのアクセント記号の区別を指定するときに使用します。この句のサポートは廃止されました。アクセント記号の区別を指定するには、COLLATION 句と NCHAR COLLATION 句に提供されている照合の適合化オプションを使用してください。

ACCENT 句は、COLLATION 句または NCHAR COLLATION 句で指定した照合に UCA (Unicode 照合アルゴリズム) を使用する場合にのみ適用されます。ACCENT RESPECT 句を使用すると、文字のアクセント記号を考慮して UCA 文字列が比較されます。たとえば、e は é よりも小さいと扱われます。ACCENT FRENCH は ACCENT RESPECT と似ていますが、フランス語の規則に合わせてアクセントが右から左の方向で比較される点が異なります。ACCENT IGNORE 句を使用すると、アクセント記号を無視して文字列が比較されます。たとえば、e と é は同等です。

データベースの作成時にアクセント記号の区別が指定されていない場合、比較とソートでのアクセント記号の区別は、デフォルトでは区別しないになります。ただし、UCA 照合を使用して作成された日本語のデータベースは例外であり、デフォルトのアクセント記号の区別は区別するになります。

ASE COMPATIBLE clause

SYS.SYSCOLUMNS ビューと SYS.SYSINDEXES ビューを作成しません。デフォルトでは、これらのビューは Watcom SQL で使用可能なシステムテーブルとの互換性を保つために作成されます (このソフトウェアのバージョン 4 以前)。これらのビューは、Adaptive Server Enterprise の互換ビュー dbo.syscolumns および dbo.sysindexes と競合します。

BLANK PADDING clause

データベースサーバは、すべての文字列を、可変長であり VARCHAR ドメインを使用して格納されている文字列として比較します。これには、固定長の CHAR カラムまたは NCHAR カラムの文字列比較も含まれます。また、データベースへの値の格納時に、データベースサーバは値の末尾への空白のパディングやこの削除は行いません。

デフォルトでは、データベースサーバは空白を意味のある文字として扱います。たとえば、値 'a ' (文字 'a' と、後続の 1 つのブランク) は、単一文字の文字列 'a' とは等しくありません。不等号比較の照合でも、ブランクは、ブランク以外の文字として扱われます。

ブランク埋め込みが有効である場合 (BLANK PADDING ON を指定)、文字列比較のセマンティックは ANSI/ISO SQL 標準とさらに近づきます。ブランク埋め込みが有効であると、データベースサーバはどのような比較でも末尾の空白を無視します。

上に挙げた例では、ブランクを埋め込まれたデータベースで 'a ' を 'a' に対して等号比較すると、TRUE が返されます。ブランクを埋め込まれたデータベースでは、固定長文字列の値は、アプリケーションによってフェッチされたときにブランクで埋め込まれます。このような文字の割り当てが行われたときにアプリケーションが文字列のトランケーション警告を受け取るかどうかは、ansi_blanks 接続オプションによって制御されます。

CASE clause

この句は、データベースでの大文字と小文字の区別を指定するときに使用します。この句のサポートは廃止されました。大文字と小文字の区別を指定するには、COLLATION 句と NCHAR COLLATION 句に提供されている照合の適合化オプションを使用してください。

CASE RESPECT 文を使用すると、すべての CHAR データ型と NCHAR データ型を比較するときに大文字と小文字が区別されます。UCA を使用した比較では、元の文字とアクセント記号がすべて同じ場合にのみ、大文字と小文字の違いが考慮されます。その他の照合の場合、大文字と小文字が区別されます。たとえば、a は A よりも小さく、A は b よりも小さいというように扱われます。CASE IGNORE 句を使用すると、大文字と小文字を区別せずに文字列が比較されます。大文字と小文字は同一と見なされます。

データベースの作成時に大文字と小文字の区別が指定されていない場合、比較とソートでの大文字と小文字の区別は、デフォルトでは区別しないになります。ただし、UCA 照合を使用して作成された日本語のデータベースは例外であり、デフォルトの大文字と小文字の区別は区別するになります。

CASE RESPECT は、ISO/ANSI SQL 標準との互換性を保つために用意されています。大文字と小文字を区別するデータベースであっても、データベースの識別子については大文字と小文字は常に区別されません。

CHECKSUM clause

チェックサムは、データベースページがディスク上で変更されたかどうかを判断するために使用します。グローバルチェックサムを有効にしてデータベースを作成した場合、チェックサムはページがディスクに書き込まれる直前に計算されます。そのページが次にディスクから読み出されるときに、ページのチェックサムが再計算されて、ページに保存されているチェックサムと比較されます。チェックサムが異なる場合は、ディスク上でページが変更されており、エラーが発生します。グローバルチェックサムを有効にして作成されたデータベースも、チェックサムを使用して検証されます。次の文を実行することによって、データベースがグローバルチェックサムを有効にして作成されたかどうかをチェックできます。

```
SELECT DB_PROPERTY ( 'Checksum' );
```

グローバルチェックサムがオンの場合、このクエリは ON を返します。それ以外の場合は OFF を返します。デフォルトでグローバルチェックサムはオンです。そのため、CHECKSUM 句を省略すると ON が適用されます。

リムーバブルドライブなどのストレージデバイス上で実行されているデータベースでは、データベースファイルの破損を速やかに検出できるように、この句の設定に関係なく、データベースサーバによって常に書き込みチェックサムが有効になります。また、検証アクティビティの実行時に重要なページのチェックサムも計算されます。

グローバルチェックサムが有効になっていないデータベースでは、-wc オプションを使用して書き込みチェックサムを有効にすることができます。

COLLATION clause

COLLATION 句で指定した照合は、文字データ型 (CHAR、VARCHAR、LONG VARCHAR) のソートと比較に使用されます。照合は、使用されるエンコード (文字セット) に文字の比較と順序付けに関する情報をもたらすものです。
COLLATION 句が指定されていない場合、データベースサーバはオペレーティングシステムの言語とエンコードに基づいて照合を選択します。

照合は、SQL Anywhere 照合アルゴリズム (SACA) を使用する照合リストから選択するか、Unicode 照合アルゴリズム (UCA) にすることができます。UCA が指定されている場合は、ENCODING 句も指定します。

照合は慎重に選択してください。データベースの作成後に照合は変更できません。

必要に応じて、文字列のソートや比較を詳細に制御することために、照合の調整オプション (`collation-tailoring-string`) を指定することもできます。これらのオプションは、"キーワード=値" のペアの形式で、括弧で囲んで指定して、その後ろに照合名を記述します。たとえば、`... CHAR COLLATION 'UCA(locale=es;case=respect;accent=respect)'` のように記述します。

DATABASE SIZE clause

このオプションの句を使用して、データベースファイルの初期サイズを設定します。単位をキロバイト、メガバイト、ギガバイト、またはページで指定するには、それぞれ KB、MB、GB、または PAGES を使用します。

作成時にファイルサイズを指定することは、ファイル用の領域を事前に割り付ける手段となります。このようにすると、データベースがあるドライブの空き領域が不足する危険性を小さくすることができます。また、データベースサイズを拡大する操作は時間を要するため、それが必要となる前にデータベースに保存できるデータの量を増やすことがパフォーマンスの向上につながります。

DBA USER and DBA PASSWORD clauses

これらの句は、データベースの DBA ユーザ ID およびパスワードを指定するときに使用します。

デフォルトでは、MINIMUM PASSWORD LENGTH 句が指定されていて、異なる値に設定されていないかぎり、パスワードは 6 文字以上である必要があります。パスワードには 7 ビット ASCII 文字が含まれている必要があります。それ以外の文字を使用すると、サーバがクライアントの文字セットを UTF-8 に変換できない場合、パスワードが機能しないことがあります。

ENCODING clause

COLLATION 句で指定するほとんどの照合には、エンコード (文字セット) と順序付けの両方が定義されています。そのような照合については、ENCODING 句を指定する必要はありません。ただし、COLLATION 句に指定されている値が UCA (Unicode 照合アルゴリズム) の場合、ENCODING を使用してロケール固有のエンコードを指定し、比較と順序指定に UCA を活用できます。ENCODING 句では、CHAR データ型に UTF-8 または任意のシングルバイトエンコードを指定できます。ENCODING は、UTF-8 以外のマルチバイトエンコードを指定できません。

UCA 照合を選択した場合は、オプションで照合の適合化オプションを指定できます。

COLLATION が UCA に設定され、ENCODING が指定されていない場合、データベースサーバでは UTF-8 が使用されます。

ENCRYPTED or ENCRYPTED TABLE clause

暗号化すると、格納データを解読できなくなります。データベース全体を暗号化するには、ENCRYPTED キーワード (TABLE なし) を使用します。テーブルの暗号化のみを有効にするときは、ENCRYPTED TABLE 句を使用します。テーブルの暗号化を有効化すると、以降に ENCRYPTED 句を使用して作成されるテーブルや変更されるテーブルがデータベースの作成時に指定した設定を使用して暗号化されることになります。

データベースとテーブルエンコーディングには、単純難読化と強力な暗号化という 2 つのレベルがあります。難読化は暗号化ではありません。暗号に関する知識を持ったユーザはデータを解読できます。強力な暗号化を使用すると、データは判読不能になり、事実上は解読できません。

単純難読化の場合は、ENCRYPTED ON ALGORITHM SIMPLE または ENCRYPTED ALGORITHM SIMPLE を指定するか、アルゴリズムやキーを指定せずに ENCRYPTED ON 句を指定します。

強力な暗号化では、128 ビットまたは 256 ビットの AES アルゴリズムを使用する ENCRYPTED ON ALGORITHM と KEY 句を指定して暗号化キーを指定します。キーの値には、少なくとも 16 文字の長さを持ち、大文字と小文字を含み、数字、特殊文字を使用したものを選びます。キーは文字列または変数名で指定できます。

> ⚠ 警告
>
> 強力な暗号化が適用されたデータベースの場合、キーのコピーは必ず安全な場所に保管してください。暗号化キーがわからなくなったら、たとえテクニカルサポートに依頼したとしても、決してデータにアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

既存のデータベースの暗号化されたコピーは、CREATE ENCRYPTED DATABASE 文を使用して作成することもできます。

JCONNECT clause

jConnect JDBC ドライバからシステムカタログ情報にアクセスできるようにするには、JCONNECT ON を指定します。この句によって jConnect をサポートするシステムオブジェクトがインストールされます。jConnect システムオブジェクトを除外するには、JCONNECT OFF を指定します。その場合でも、システム情報にアクセスしないかぎり、JDBC を使用できます。JCONNECT はデフォルトで ON です。

MINIMUM PASSWORD LENGTH clause

最小パスワード長を設定するには、この句を使用します。この句を指定しない場合、新しいデータベースに対する最小パスワード長は 6 です。

PAGE SIZE clause

データベースのページサイズは、2048、4096、8192、16384、または 32768 バイトです。デフォルトのページサイズは 4096 バイトです。2048 ページサイズは廃止予定です。ページサイズを大きくすると、一般に大規模なデータベースのパフォーマンスは向上しますが、オーバヘッドは増大します。

例:

```
CREATE DATABASE 'c:¥¥temp¥¥my_db.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
PAGE SIZE 4096;
```

> ℹ 注記
>
> 現在のサーバで使用しているページサイズより大きいページサイズは指定できません。サーバページサイズは、最初に起動されたデータベースのセットから取得されるか、サーバコマンドラインで -gp オプションを使用して設定されます。

NCHAR COLLATION clause

NCHAR COLLATION 句で指定した照合は、各国の文字データ型 (NCHAR、NVARCHAR、LONG NVARCHAR) のソートと比較に使用されます。照合は、各国の文字に使用される UTF-8 エンコード (文字セット) に文字の順序付けに関する情報をもたらすものです。NCHAR COLLATION 句が指定されない場合、データベースサーバは Unicode 照合アルゴリズム (UCA) を使用します。その他に使用できる照合は UTF8BIN のみです。UTF8BIN は、エンコードが 0x7E を超えるすべての文字のバイナリ順を規定します。

必要に応じて、文字列のソートや比較を詳細に制御することために、照合の調整オプション (`collation-tailoring-string`) を指定することもできます。これらのオプションは、キーワード=値の形式で、照合名の後ろに、引用符で囲んで記述します。たとえば、`... NCHAR COLLATION 'UCA(locale=es;case=respect;accent=respect)'` のように記述します。ACCENT 句または CASE 句と、大文字小文字やアクセント記号の区別の設定を含む照合の適合化文字列とを、併せて指定すると、ACCENT 句と CASE 句の値はデフォルトとしてのみ使用されます。

> **注記**
>
> UCA 照合を指定すると、照合の適合化のすべてのオプションがサポートされます。その他の照合の場合、大文字小文字の区別の適合化オプションのみがサポートされます。
>
> 照合の適合化オプションを使用して作成したデータベースは、10.0.1 より前のデータベースサーバでは起動できません。

SYSTEM PROCEDURE AS DEFINER { ON | OFF } clause

SYSTEM PROCEDURE AS DEFINER 句は、権限付き作業を実行する 16.0 より前のシステムプロシージャを、invoker または definer (所有者) の権限で実行するかどうかを指定します。ON の場合、これらのシステムプロシージャは definer (所有者) で実行されます。OFF の場合、これらのシステムプロシージャは invoker の権限で実行されます。

この句を指定しない場合、デフォルトではこれらのプロシージャは invoker (呼び出したユーザ) の権限で実行されます。

この設定は、ユーザ定義プロシージャまたはバージョン 16.0 以降に導入されたシステムプロシージャには影響しません。

[TRANSACTION] LOG clause

トランザクションログは、データベースサーバがデータベースに対するすべての変更を記録するファイルです。トランザクションログはバックアップとリカバリ、データレプリケーションで重要な役割を果たします。デフォルトは LOG ON です。

トランザクションログのミラーを使用する場合、LOG 句の MIRROR オプションを使用するとファイル名を指定できます。トランザクションログミラーはトランザクションログと同一のコピーで、通常は別のデバイスで管理され、データを確実に保護しています。デフォルトでは、データベースサーバはトランザクションログミラーを使用されません。

備考

指定された名前と属性でデータベースファイルを作成します。データベースはオペレーティングシステムファイルとして格納されます。この文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

別のデータベースを作成するには、データベースに接続されている必要があります。たとえば、ユーティリティデータベースに接続します。

データベースサーバを実行中のアカウントには、ファイルが作成されたディレクトリの書き込みパーミッションが必要です。

クライアントに送信されるメッセージは、どのタイプのデータベース暗号化がデータベースに使用されるのかを示します。暗号化を使用している場合は、使用されているアルゴリズムも表示されます。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

オペレーティングシステムファイルが作成されます。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

CREATE DATABASE 文は Adaptive Server Enterprise でサポートされていますが、使用する句は異なります。

例

次の文は、C:¥temp ディレクトリにデータベースファイル temp.db を作成します。

```
CREATE DATABASE 'c:¥¥temp¥¥temp.db'
DBA USER 'DBA' DBA PASSWORD 'passwd';
```

次の文は、C:¥temp ディレクトリにデータベースファイル mydb.db を作成します。

```
CREATE DATABASE 'C:¥¥temp¥¥mydb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
TRANSACTION LOG ON
CASE IGNORE
PAGE SIZE 4096
ENCRYPTED OFF
BLANK PADDING OFF;
```

次の文は、コードページ 1252 を使用してデータベースを作成し、CHAR と NCHAR のデータ型の両方に UCA を使用します。比較とソート時に、アクセント記号と大文字と小文字の区別が考慮されます。

```
CREATE DATABASE 'c:¥¥temp¥¥uca.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
COLLATION 'UCA'
ENCODING 'CP1252'
NCHAR COLLATION 'UCA'
ACCENT RESPECT
CASE RESPECT;
```

次の文はデータベース myencrypteddb.db を作成します。これは、単純難読化によって暗号化されます。

```
CREATE DATABASE 'c:¥¥temp¥¥myencrypteddb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
ENCRYPTED ON;
```

次の文はデータベース mystrongencryptdb.db を作成します。また、キー gh67AB2 を使用して暗号化されます (強力な暗号化)。

```
CREATE DATABASE 'c:¥¥temp¥¥mystrongencryptdb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
ENCRYPTED ON KEY 'gh67AB2';
```

次の文はデータベース mytableencryptdb.db を作成します。これは、単純難読化を使用してテーブルの暗号化が有効にされます。ENCRYPTED の後にキーワード TABLE を挿入した場合、データベースの暗号化ではなくテーブルの暗号化を示すことに注意してください。

```
CREATE DATABASE 'c:¥¥temp¥¥mytableencryptdb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
ENCRYPTED TABLE ON;
```

次の文はデータベース mystrongencrypttabledb.db を作成します。これは、単純暗号化を使用してテーブルの暗号化が有効にされます。

```
CREATE DATABASE 'c:¥¥temp¥¥mystrongencrypttabledb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
ENCRYPTED TABLE ON 'SIMPLE';
```

次の文は、照合 1252LATIN1 を使用するデータベースファイル mydb.db を作成します。NCHAR 照合を UCA に設定し、ロケールセットを es に設定して、大文字と小文字の区別とアクセント記号の区別を有効にします。

```
CREATE DATABASE 'c:¥¥temp¥¥my2.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
COLLATION '1252LATIN1(case=respect)'
NCHAR COLLATION 'UCA(locale=es;case=respect;accent=respect)';
```

次の文は、ギリシャ語の照合を設定したデータベースを作成します。

```
CREATE DATABASE 'c:¥¥temp¥¥mydb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
COLLATION '1253ELL';
```

次の文は、トランザクションログミラーを使用するデータベース mydb.db を作成します。

```
CREATE DATABASE 'c:¥¥mydb.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
TRANSACTION LOG ON 'mydb.log'
MIRROR 'd:¥¥mydb.mlg';
```

関連情報

エスケープシーケンス [13 ページ]
ALTER DATABASE 文 [630 ページ]
CREATE ENCRYPTED DATABASE 文 [798 ページ]

1.4.4.56 CREATE DBSPACE 文

新しいデータベース領域を定義し、関連するデータベースファイルを作成します。

構文

```
CREATE DBSPACE dbspace-name AS filename
```

パラメータ

dbspace-name

DB 領域の名前を指定します。これは、filename を使用して指定する実際のデータベースファイル名ではありません。dbspace-name は、文やプロシージャなどで参照可能な内部名です。system、temporary、temp、translog、translogmirror は事前定義の DB 領域用に予約されているため、これらの名前を DB 領域の名前として使用することはできません。

ピリオド (.) を含む値を指定すると、エラーが返されます。

filename

データベースファイルの名前を指定します。ファイルへのパスを含めることもできます。パスが指定されていない場合は、メインデータベースファイルと同じロケーション (ディレクトリ) にデータベースファイルが作成されます。別のロケーションを指定する場合、パスはデータベースサーバを基準にした相対パスになります。円記号 (¥) は、SQL 文字列のエスケープ文字であるため、2 つ重ねます。

filename パラメータには、文字列リテラルまたは変数のどちらかを指定してください。

備考

CREATE DBSPACE 文は、新しいデータベースファイルを作成します。データベースの作成時には、ファイルは 1 つしかありません。作成されるすべてのテーブルとインデックスは、このファイルの中に入ります。CREATE DBSPACE は、新しいファイルをデータベースに追加します。このファイルはメインファイルではなく、別のディスクドライブ上にあることもあり、1 つの物理的なデバイスよりも大きなデータベースを作成できるようになります。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

各データベースには、メインファイル以外に、最大で 12 の DB 領域という制限があります。

テーブル、インデックスなどの各オブジェクトは、1 つの DB 領域内にすべてが格納されます。CREATE 文の IN 句は、オブジェクトを入れる DB 領域を指定します。デフォルトでは、オブジェクトはシステムデータベースファイルの中に入ります。テーブルを作成する前に、default_dbspace オプションを設定することで、テーブルが作成される DB 領域を指定することもできます。

権限

MANAGE ANY DBSPACE システム権限が必要です。

関連する動作

オートコミット。自動チェックポイント。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、c:¥ ディレクトリに libbooks という DB 領域を作成します。後続の CREATE TABLE 文は、libbooks DB 領域にテーブル LibraryBooks を作成します。

```
CREATE DBSPACE libbooks
AS 'c:¥¥library.db';
CREATE TABLE LibraryBooks (
  title char(100),
  author char(50),
  isbn char(30),
) IN libbooks;
```

関連情報

文字列 [10 ページ]
DROP DBSPACE 文 [1034 ページ]

1.4.4.57 CREATE DECRYPTED DATABASE 文

すべてのトランザクションログおよび DB 領域を含む既存のデータベースの、復号化されたコピーを作成します。

構文

```
CREATE DECRYPTED DATABASE newfile
FROM oldfile
[ KEY key ]
```

パラメータ

FROM clause

この句は、コピーするデータベースの名前 (oldfile) を指定するために使用します。

KEY clause

この句は、データベースの復号化に必要な暗号化キーを指定するために使用します。キーには文字列または変数名を指定できます。既存のデータベースのエンコードに使用されたのが、キーが不要な単純難読化である場合は、KEY 句を指定しないでください。

備考

CREATE DECRYPTED DATABASE 文は、新しいデータベースファイル (newfile) を作成し、元のデータベースファイル (oldfile) の置換または削除はしません。

oldfile に含まれるすべての暗号化されたテーブルは、newfile では暗号化されず、テーブルの暗号化は有効になりません。

> **注記**
>
> SQL Anywhere 12 以降を使用して作成されたデータベースでは、不正なアクセスからデータを保護するため、ISYSCOLSTAT、ISYSUSER、ISYSEXTERNLOGIN の各システムテーブルは常に暗号化されたままになります。

oldfile でトランザクションログまたはトランザクションログミラーが使用されている場合は、ファイル名がそれぞれ newfile.log および newfile.mlg に変更されます。

oldfile に DB 領域ファイルが含まれている場合は、復号化を示す D がファイル名に付加されます。たとえば、CREATE DECRYPTED DATABASE 文を実行すると、ファイル mydbspace.dbs は mydbspace.dbsD に変更されます。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

リカバリが必要なデータベースに対してこの文を実行することはできません。この文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

復号化中のデータベースには接続できません。別のデータベースに接続されている必要があります。たとえば、ユーティリティデータベースに接続します。実行中のデータベースは暗号化できません。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の最初の文では、*demo.db* の AES256 で暗号化されたコピー `demoEncrypted.db` を作成します。2 番目の文では、`demoEncrypted.db` の復号化されたコピー `demoDecrypted.db` を作成します。

```
CREATE ENCRYPTED DATABASE 'demoEncrypted.db'
   FROM 'demo.db'
   KEY 'Sd8f6654*Mnn'
   ALGORITHM 'AES256';
CREATE DECRYPTED DATABASE 'demoDecrypted.db'
   FROM 'demoEncrypted.db'
   KEY 'Sd8f6654*Mnn';
```

関連情報

CREATE ENCRYPTED DATABASE 文 [798 ページ]
CREATE ENCRYPTED FILE 文 [801 ページ]
CREATE DECRYPTED FILE 文 [792 ページ]

1.4.4.58 CREATE DECRYPTED FILE 文

強力な暗号化が施されているデータベースの復号化されたコピーを作成します。この文を使用して、トランザクションログ、トランザクションログミラー、および DB 領域の復号化されたコピーを作成できます。

構文

```
CREATE DECRYPTED FILE newfile
FROM oldfile KEY key
```

パラメータ

FROM clause

暗号化ファイル名の一覧が表示されます。

KEY clause

暗号化されたファイルへのアクセスに必要なキーをリストします。このキーには文字列または変数名を使用できます。

備考

CREATE DECRYPTED DATABASE 文は、データベースの最暗号化に推奨されるメソッドです。CREATE DECRYPTED DATABASE 文が失敗したら、次は CREATE DECRYPTED FILE 文を使用します。CREATE DECRYPTED FILE 文は、技術サポートの目的でデータベースを暗号化する必要があるときに、頻繁に使用されます。この文は、トランザクションログファイル、トランザクションログミラーファイル、DB 領域ファイルなど、関連付けられたデータベースファイルの復号化にも使用できます。

データベースファイルの再暗号化に加え、CREATE DECRYPTED DATABASE 文は、トランザクションログ、トランザクションログミラー、および DB 領域ファイルといった関連ファイルを自動的に再暗号化します。CREATE DECRYPTED FILE 文を使用する場合、関連ファイルを個別に複合化する必要があります。

元のデータベースファイルは、暗号化キーを使用して強力な暗号化が施されている必要があります。復号化後のファイルは、暗号化されたファイルの正確なコピーですが、暗号化されていないため、暗号化キーは必要ありません。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

この文を使用してデータベースを復号化する場合は、対応するトランザクションログファイル (と DB 領域) も復号化しなければ、データベースを使用できません。

リカバリを必要とするデータベースを復号化する場合は、対応するトランザクションログファイルの復号化および新しいデータベースのリカバリも必要です。このプロセスでトランザクションログファイルの名前が変わることはありません。したがって、データベースとトランザクションログファイルの名前を変更した場合は、復号化後のデータベースに対して dblog -t を実行する必要があります。

テーブルの暗号化が有効なデータベースでは、この文を使用できません。テーブルを復号化する場合、ALTER TABLE 文の NOT ENCRYPTED 句を使用して復号化します。

> **注記**
>
> SQL Anywhere 12 以降を使用して作成されたデータベースでは、データベースファイルへの不正なアクセスからデータを保護するため、ISYSCOLSTAT、ISYSUSER、ISYSEXTERNLOGIN の各システムテーブルは常に暗号化されたままになります。

この文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

復号化中のデータベースには接続できません。別のデータベースに接続されている必要があります。たとえば、ユーティリティデータベースに接続します。実行中のデータベースは暗号化できません。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、架空の暗号化されたデータベース encContacts を復号化して、新しい暗号化されていないデータベース contacts を作成します。

```
CREATE DECRYPTED FILE 'contacts.db'
FROM 'encContacts.db'
KEY 'Sd8f6654*Mnn';
```

関連情報

ALTER TABLE 文 [705 ページ]
CREATE ENCRYPTED FILE 文 [801 ページ]
CREATE DECRYPTED DATABASE 文 [790 ページ]
CREATE ENCRYPTED DATABASE 文 [798 ページ]

1.4.4.59 CREATE DOMAIN 文

データベース内にドメインを作成します。

構文

```
CREATE { DOMAIN | DATATYPE } domain-name [ AS ] data-type
[ [ NOT ] NULL ]
[ DEFAULT default-value ]
[ CHECK( condition ) ]
```

```
[ AS USER user-name ]

domain-name : identifier

data-type :  built-in data type, with precision and scale, or another domain
```

パラメータ

DOMAIN | DATATYPE clause

CREATE DOMAIN は ANSI/ISO SQL 標準で定義されているため、CREATE DATATYPE ではなく、CREATE DOMAIN を使用することをおすすめします。

data-type

データ型に、組み込みデータ型の 1 つを設定するか、またはドメイン名を指定して別のドメインのデータ型を設定します。

例:

```
CREATE DOMAIN a INT;
CREATE DOMAIN b a;
```

%TYPE または %ROWTYPE 属性を指定し、データ型にテーブルまたはビューのカラムまたはローのデータ型を設定することもできます。ただし %ROWTYPE (TABLE REF (`table-reference-variable`) %ROWTYPE) のテーブル参照変数を指定することはできません。

NULL clause

この句を使用すると、ドメインに NULL 入力可かどうかを指定できます。ドメインを使用してカラムを定義する場合、NULL 入力可かどうかは次のようにして決まります。

- カラム定義で指定された Null 入力可能。
- ドメイン定義で指定された Null 入力可能。
- NULL 入力属性がカラム定義とドメイン定義のどちらでも明示的に指定されていない場合、allow_nulls_by_default オプションの設定が使用されます。

CHECK clause

検査制約を使用してドメインを作成する場合は、検査制約の検索条件の中に @ 記号のプレフィクスを持つ変数名を使用できます。データ型をカラムの定義内で使う場合、このような変数をカラム名に置き換えます。これにより、そのドメインで定義された各テーブルにドメインの検査制約を適用することができます。

AS USER clause

オブジェクトの所有者を指定します。

備考

ドメインは、必要に応じて精度と小数点以下の桁数を含めた組み込みデータ型のエイリアスです。データベース内の使いやすさを改善し、一貫性を高めます。

ドメインはデータベース内のオブジェクトです。名前を付けるには識別子のルールに従います。ドメイン名は、組み込みデータ型名の場合と同じで常に大文字と小文字を区別しません。ただし、照合を区別しないわけではありません。たとえば、トルコ語

照合の場合は、ドメイン名 "image" は使用できますが (小文字を使用して作成されたため)、ドメイン "IMAGE" は使用できません。動作する文字形式 (大文字/小文字) は作成したときの文字形式で、これは SYS.SYSUSERTYPE テーブルの type_name カラムで確認できます。

データ型を作成するユーザは、自動的にそのデータ型の所有者となります。CREATE DATATYPE 文の中では、所有者を指定できません。ドメイン名はユニークにします。また、すべてのユーザはプレフィクスとして所有者を使わなくてもデータ型にアクセスできます。

ドメインは、CHECK 条件と DEFAULT 値を持つことができ、ユーザ側でそのデータ型が NULL 値を使えるかどうかを指定できます。これらの条件と値は、ドメインで定義するカラムによって継承されます。カラム定義で明示的に指定された条件または値は、ドメイン型に指定された条件または値を上書きします。

SYSUSERTYPE システムビューに作成者が記録されると、ドメインの作成者に注釈をつけるための AS USER 句がデータベースアンロードスクリプトに生成されます。それ以外の重要性はありません。

権限

所有するドメインを作成するには、CREATE DATATYPE または CREATE ANY OBJECT システム権限が必要です。他のユーザが所有するドメインは作成できません。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

ドメインのサポートは、オプションの ANSI/ISO SQL 言語機能 F251 です。

例

データベーステーブルのカラムは、人の名前を入れたり、住所を格納したりできます。次のようなドメインを定義できます。

```
CREATE DOMAIN person CHAR(30) NOT NULL;
CREATE DOMAIN address CHAR(35);
```

住所ドメインと人名ドメインでは、含むことができる文字数が異なります。また、人名タイプのカラムには NULL 値を含めることができません。

整数値は一般的にテーブル内のローに対する一意の識別子として使用されます。次の文は、NULL 値使用不可の符号なし整数であるドメイン名識別子を作成します。この識別子はデフォルトでオートインクリメントに設定されます。

```
CREATE DOMAIN identifier UNSIGNED INT
NOT NULL
DEFAULT AUTOINCREMENT;
```

こうして定義されたドメインは、既存のデータ型と同様に使用できます。次のようにしてテーブルを定義できます。次の文を実行するには、CREATE TABLE 権限が必要です。

```
CREATE TABLE myCustomers (
   ID        identifier  PRIMARY KEY,
   Name      person,
   Street    address
);
```

上の例では、テーブルのプライマリキー ID は、非 NULL の符号なし整数値を自動的にインクリメントします (もちろんプライマリキーを NULL にすることはできません)。

テーブルの多くは、このような識別子カラムを必要とします。毎回同じ属性セットを指定する代わりに、識別子ドメインを作成してさまざまな個所で使用する方が便利です。同じことが、人物名やアドレスにも当てはまります。ドメインを使用することで、テーブルスキーマのメンテナンスが容易になります。

ドメインは他のドメインの観点から定義できます。次はその例です。

```
CREATE DOMAIN simple_identifier UNSIGNED INT;
CREATE DOMAIN identifier simple_identifier NOT NULL DEFAULT AUTOINCREMENT;
```

次の例の CREATE DOMAIN 文は、myCustomers テーブルの Name および Street カラムのデータ型に基づいてドメインを作成します。これらの新しいドメインは、myCustomers2 テーブルのカラムの定義に使用されます。

```
CREATE DOMAIN customers_name myCustomers.Name%TYPE NOT NULL;
CREATE DOMAIN customers_street myCustomers.Street%TYPE;

CREATE TABLE myCustomers2 (
   ID        identifier  PRIMARY KEY,
   Name      customers_name,
   Street    customers_street
);
```

この例の場合、myCustomers.Name と myCustomers.Street からのみデータ型が選択されているため、customers_name ドメインに対して NOT NULL 属性は指定されません。

ドメインとテーブルの作成順序が重要です。当然ながら、myCustomers テーブルの前に customers_name ドメインと customers_street ドメインを作成することはできません。また、ドメインが定義された後に myCustomers テーブル内のカラムの定義を調整しても、変更内容はドメイン定義へと自動的に継承されません。

ドメインを作成するとき、CHECK 制約を指定することで、不適切な値がこの型のカラムに入力されるのを防止できます。次の文は、phone_number という名前のドメインを作成します。ここでは CHECK 制約内の正規表現を使用して、先頭に 0 または 1 を持たない 3 桁の市外局番、先頭に 0 または 1 を持たない 3 桁の局番、および 4 桁の番号をダッシュまたは空白で区切った、正しいフォーマットの 12 文字の北米地域の電話番号であることが確認されます。

```
CREATE DOMAIN phone_number CHAR(12) NULL
CHECK( @phone_number REGEXP '([2-9][0-9]{2}-[2-9][0-9]{2}-[0-9]{4})|([2-9][0-9]
{2}¥s[2-9][0-9]{2}¥s[0-9]{4})');
```

次の文は、ローのデータを保持できる MyRow という名前の ROW ドメインを作成します。

```
CREATE DOMAIN MyRow ROW( a INT, b INT );
```

新しい MyRow ドメインは、SQL 文内で参照されます。例:

```
CREATE FUNCTION Swap( @parm MyRow )
   RETURNS (MyRow)
```

```
    BEGIN
       DECLARE @ret MyRow;
       SET @ret = ROW( @parm.b, @parm.a );
       RETURN @ret;
     END;
```

次の文は、ローの配列を保持できる MyArray という名前の ARRAY ドメインを作成します。

```
CREATE DOMAIN MyArray ARRAY( 10 ) OF ROW( a INT, b INT );
```

MyRow に対する前述の宣言を前提として、この例は次のように書くこともできます。

```
CREATE DOMAIN MyArray ARRAY( 10 ) OF MyRow;
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
DROP DOMAIN 文 [1035 ページ]
SQL データ型 [124 ページ]

1.4.4.60 CREATE ENCRYPTED DATABASE 文

すべてのトランザクションログおよび DB 領域を含む既存のデータベースの暗号化されたコピーを作成するか、テーブル暗号化が有効にされた既存のデータベースのコピーを作成します。

構文

データベースの暗号化されたコピーの作成

```
CREATE ENCRYPTED DATABASE newfile
FROM oldfile
[ KEY newkey ]
[ ALGORITHM algorithm ]
[ KEY DERIVATION ITERATIONS number ]
[ OLD KEY oldkey ]
```

```
algorithm :
    'SIMPLE'
  | 'AES'
  | 'AES256'
  | 'AES_FIPS'
  | 'AES256_FIPS'
```

テーブル暗号化が有効にされたデータベースのコピーの作成

```
CREATE ENCRYPTED TABLE DATABASE newfile
FROM oldfile
[ KEY newkey ]
[ ALGORITHM algorithm ]
[ KEY DERIVATION ITERATIONS number ]
```

```
[ OLD KEY oldkey ]
```

パラメータ

CREATE ENCRYPTED DATABASE clause

新しいデータベースの名前を指定します。

CREATE ENCRYPTED TABLE DATABASE clause

新しいデータベースの名前を指定します。新しいデータベースは暗号化されませんが、テーブルの暗号化が有効になります。

FROM clause

エンコードされていない元のデータベースファイルの名前 (`oldfile`) を指定します。

KEY clause

`newfile` の暗号化キーを指定します。このキーには文字列または変数名を使用できます。ALGORITHM 'SIMPLE' には不要です。

ALGORITHM clause

`newfile` で使用するエンコードアルゴリズムを指定します。単純難読化または何らかの AES 暗号化を選択できます。KEY 句を指定し、ALGORITHM 句を指定しない場合は、デフォルトで AES (128 ビット暗号化) が使用されます。`algorithm` に 'SIMPLE' を指定する場合は、KEY 句を指定しません。

KEY DERIVATION ITERATIONS

暗号化キーのハッシュ回数を指定します。1 から 1000 の間の整数を指定してください。デフォルト値は 2 で、これは 2000 回の繰り返しを意味します。回数が多いほど、セキュリティは高くなります。データベースを速度の遅いコンピュータ上で稼働させている場合、繰り返し回数が多すぎるとデータベースの起動が遅くなる可能性があります (いったんデータベースが稼働を始めれば、パフォーマンスへの影響はありません)。

OLD KEY clause

この句は、`oldfile` の暗号化キーを指定するために使用します。このキーには文字列または変数名を使用できます。この句が必要なのは、`oldfile` が何らかの AES 暗号化を使用して暗号化されている場合のみです。

備考

この文を使用して、既存のデータベースの暗号化されたコピー (すべてのトランザクションログおよび DB 領域を含む) を作成できます。

また、この文を使用して、データベースのコピーを作成し、このコピーでテーブルの暗号化を有効にすることもできます。

データベースファイル `oldfile` には、暗号化されていないデータベース、暗号化されたデータベース、またはテーブル暗号化が有効なデータベースを指定できます。

データベースの暗号化されたコピーの作成では、既存のデータベース `oldfile` から、その暗号化されたコピーである `newfile` を作成します。

テーブル暗号化が有効なデータベースのコピーの作成では、既存のデータベース oldfile から、そのコピー newfile を作成し、このコピーのテーブル暗号化を有効にします。この構文を使用する場合、oldfile で暗号化されていたすべてのテーブルは、newfile でも暗号化されます。oldfile に暗号化されているテーブルがない場合にテーブルを暗号化するには、暗号化するテーブルごとに ALTER TABLE...ENCRYPTED 文を実行してください。

どちらの構文を使用しても、oldfile が置換されたり削除されたりすることはありません。

oldfile でトランザクションログファイルまたはトランザクションログミラーファイルが使用されている場合は、ファイル名がそれぞれ newfile.log および newfile.mlg に変更されます。

oldfile に DB 領域ファイルが含まれている場合は、暗号化を示す E がファイル名に付加されます。たとえば、CREATE ENCRYPTED DATABASE 文を実行すると、ファイル mydbspace.dbs は mydbspace.dbsE に変更されます。

この文を使用して、データベースの暗号化アルゴリズムとキーを変更できます。ただし、CREATE ENCRYPTED DATABASE 文は、新しいファイル (newfile) を作成しますが、前バージョンのファイル (oldfile) を置換または削除することはありません。

データベースを暗号化するときや、データベース内のテーブル暗号化を有効にするときには、暗号化キーを指定する必要があります。本ソフトウェアではパスワードベースの鍵導出関数 2 (PBKDF2) を使用します。これは、総当たり攻撃からキーを保護する PKCS#5 標準の一部です。本ソフトウェアでは暗号化キーに対して暗号化ハッシュを繰り返し適用します。KEY DERIVATION ITERATIONS 句を使用してハッシュの適用回数を指定してください。

CREATE ENCRYPTED DATABASE 文と CREATE ENCRYPTED TABLE DATABASE 文は、リカバリを必要とするデータベースに対しては実行できません。代わりに CREATE ENCRYPTED FILE 文を使用する必要があります。

これらの文は、プロシージャ、トリガ、イベント、バッチではサポートされていません。

暗号化中のデータベースには接続できません。別のデータベースに接続されている必要があります。たとえば、ユーティリティデータベースに接続します。実行中のデータベースは暗号化できません。

dbunload -an オプションに -ek または -ep を使用してデータベースのアンロードと再ロードを行うと、既存のデータベースを暗号化したり、既存の暗号化キーを変更したりできます。

暗号化されたデータベースや、テーブル暗号化が有効にされたデータベースは、CREATE DATABASE 文を使用して作成することもできます。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、サンプルデータベースの暗号化されたコピー demoEnc.db を作成します。新しいデータベースの暗号化には、AES256 暗号化が使用されます。

```
CREATE ENCRYPTED DATABASE 'demoEnc.db'
   FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥sample.db'
   KEY 'Sd8f6654*Mnn'
   ALGORITHM 'AES256';
```

次の例は、サンプルデータベースのコピー demoTableEnc.db を作成します。新しいデータベースでは、テーブル暗号化が有効になります。アルゴリズムの指定なしでキーが指定されているため、AES 暗号化が使用されます。

```
CREATE ENCRYPTED TABLE DATABASE 'demoTableEnc.db'
   FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥sample.db'
   KEY 'Sd8f6654';
```

関連情報

CREATE DECRYPTED DATABASE 文 [790 ページ]
CREATE ENCRYPTED FILE 文 [801 ページ]
CREATE DECRYPTED FILE 文 [792 ページ]
CREATE DATABASE 文 [781 ページ]
ALTER TABLE 文 [705 ページ]

1.4.4.61 CREATE ENCRYPTED FILE 文

CREATE ENCRYPTED DATABASE 文を使えない場合に、強力に暗号化されたデータベースファイルのコピーを作成します。また、トランザクションログファイル、トランザクションログミラーファイル、DB 領域ファイルの暗号化されたコピーも作成します。

構文

```
CREATE ENCRYPTED FILE newfile
FROM oldfile
KEY newkey
[ ALGORITHM algorithm ]
[ KEY DERIVATION ITERATIONS number ]
[ OLD KEY oldkey ]
```

```
algorithm :
    'AES'
    | 'AES256'
    | 'AES_FIPS'
    | 'AES256_FIPS'
```

パラメータ

FROM clause

エンコードされていない元のデータベースファイルの名前 (oldfile) を指定します。

KEY clause

newfile に使用する暗号化キーを指定します。このキーには文字列または変数名を使用できます。このキーは必須です。

ALGORITHM clause

newfile の暗号化に使用されるアルゴリズムを指定します。アルゴリズムを指定しない場合は、デフォルトで AES (128 ビットの暗号化) が使用されます。

KEY DERIVATION ITERATIONS

暗号化キーのハッシュ回数を指定します。1 から 1000 の間の整数を指定してください。デフォルト値は 2 で、これは 2000 回の繰り返しを意味します。回数が多いほど、セキュリティは高くなります。データベースを速度の遅いマシン上で稼働させている場合、繰り返し回数が多すぎるとデータベースの起動が遅くなる可能性があります (いったんデータベースが稼働を始めれば、パフォーマンスへの影響はありません)。

OLD KEY clause

oldfile が暗号化されている場合、その暗号化キーを指定します。このキーには文字列または変数名を使用できます。

備考

CREATE ENCRYPTED FILE 文は、テクニカルサポートの目的でデータベースを暗号化する必要があるけれども CREATE ENCRYPTED DATABASE 文を使えない場合のために提供されています。CREATE ENCRYPTED DATABASE 文は、データベースの暗号化のために推奨されている文です。けれども、CREATE ENCRYPTED DATABASE 文がうまくいかない場合は CREATE ENCRYPTED FILE 文を使用できます。また、CREATE ENCRYPTED FILE 文は、トランザクションログファイル、トランザクションログミラーファイル、DB 領域ファイルの暗号化されたコピーを作成する場合にも使用できます。暗号化されたデータベースに対して CREATE ENCRYPTED FILE 文を実行する場合は、そのデータベースとは異なる暗号化キーと暗号化アルゴリズムによって暗号化されたコピーを作成します。

データベースを暗号化するとき、データベースファイルだけでなく、関連するファイル (トランザクションログファイル、トランザクションログミラーファイル、DB 領域ファイル) が存在する場合はそれらのファイルについても個別に CREATE ENCRYPTED FILE 文を実行する必要があります。

CREATE ENCRYPTED FILE 文を実行すると新しいファイル (newfile) が作成され、前バージョンのファイル (oldfile) が置換または削除されることはありません。

テーブル暗号化が有効化されているデータベースの場合は、CREATE ENCRYPTED FILE 文を使用できませんので、CREATE ENCRYPTED DATABASE 文を使用してください。

CREATE ENCRYPTED FILE 文は、プロシージャ、トリガ、イベント、バッチではサポートされていません。

データベースを暗号化するときや、データベース内のテーブル暗号化を有効にするときには、暗号化キーを指定します。データベース関連ファイルを暗号化するときは、そのデータベースに関連するすべてのファイルに同じアルゴリズム、キー、繰り返し回数を指定してください。本ソフトウェアではパスワードベースの鍵導出関数 2 (PBKDF2) を使用します。これは、総当たり攻撃からキーを保護する PKCS#5 標準の一部です。本ソフトウェアでは暗号化キーに対して暗号化ハッシュを繰り返し適用します。KEY DERIVATION ITERATIONS 句を使用してハッシュの適用回数を指定してください。

-kdi オプションを使用すると、暗号化キーに適用される繰り返し回数を変更できます。繰り返し回数の最小値は 1000 で、最大値は 1,000,000 です。適用する繰り返し回数が多いほど暗号化キーの作成にかかる時間が長くなり、総当たり攻撃によってパスワードの候補を試すのにかかる時間も長くなります。

`oldfile` に関連する DB 領域ファイルまたはトランザクションログファイルが存在し、それらのファイルも暗号化する場合は、それらのファイルの新しい名前とロケーションを新しいデータベースに格納する必要があります。そのためには、次の手順に従います。

- 新しいデータベースに対して dblog -t を実行して、トランザクションログの名前とロケーションを変更します。
- 新しいデータベースに対して dblog -m を実行して、トランザクションログミラーの名前とロケーションを変更します。
- 新しいデータベースに対して ALTER DBSPACE 文を実行して、DB 領域ファイルのロケーションと名前を変更します。

CREATE ENCRYPTED FILE 文を実行する場合、暗号化しようとしているデータベースとは別のデータベースに接続する必要があります。たとえば、ユーティリティデータベースに接続します。稼働中のデータベースは暗号化できません。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、サンプルデータベース *demo.db* を暗号化し、AES_FIPS で暗号化された新しいデータベース `demo2.db` を作成します。新しいデータベースファイルはサーバの現在の作業ディレクトリに保存されます。

```
CREATE ENCRYPTED FILE 'demo2.db'
FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥demo.db'
   KEY 'Sd8f6654*Mnn'
   ALGORITHM 'AES_FIPS';
```

次の例は、サンプルデータベース *demo.db* とそのトランザクションログファイル `demo.log` を暗号化します。

```
CREATE ENCRYPTED FILE 'demo3.db'
   FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥demo.db'
   KEY 'Sd8f6654*Mnn';
CREATE ENCRYPTED FILE 'demo3.log'
   FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥demo.log'
   KEY 'Sd8f6654*Mnn';
```

新しいデータベースファイルとトランザクションログは、サーバの現在の作業ディレクトリに保存されます。新しいデータベースファイル `demo3.db` でいまだ古いトランザクションログファイルを参照しているため、コマンドプロンプトでトランザクションログユーティリティ (dblog) を使用して、新しいトランザクションログの名前を設定します。

```
dblog -ek Sd8f6654*Mnn -t demo3.log demo3.db
```

データベースの暗号化キーを変更するには、新しいキーを使用してデータベースファイルとトランザクションログのコピーを作成します。次に例を示します。

```
CREATE ENCRYPTED FILE 'c:¥¥temp¥¥demo.db'
   FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥demo.db'
   KEY     'Sd251072*Mnn'
   OLD KEY 'Sd8f6654*Mnn';
CREATE ENCRYPTED FILE 'C:¥¥temp¥¥demo.log'
   FROM 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥demo.log'
   KEY     'Sd251072*Mnn'
   OLD KEY 'Sd8f6654*Mnn';
```

新しいデータベースファイルとトランザクションログは、指定されたディレクトリに保存されます。古いデータベースファイルとそのトランザクションログをアーカイブし、その後新しいデータベースファイルとトランザクションログを古いファイルが保存されていたのと同じディレクトリに移動できるようになりました。

関連情報

CREATE ENCRYPTED DATABASE 文 [798 ページ]
CREATE ENCRYPTED DATABASE 文 [798 ページ]
CREATE DECRYPTED FILE 文 [792 ページ]
CREATE DECRYPTED DATABASE 文 [790 ページ]

1.4.4.62 CREATE EVENT 文

イベントとイベントに関連付けて定義済みアクションを自動化するイベントハンドラを定義し、スケジュールされたアクションを定義します。

構文

```
CREATE [ OR REPLACE ] EVENT [user-name.]event-name
[ TYPE event-type
          [ WHERE trigger-condition [ AND trigger-condition ] ... ]
    | SCHEDULE schedule-spec, ... ]
[ ENABLE | DISABLE ]
[ AT { CONSOLIDATED | REMOTE | ALL } ]
[ FOR { PRIMARY | ALL } ]
[ HANDLER
      BEGIN
...
     END ]
```

```
event-type :
  BackupEnd
| Connect
| ConnectFailed
| DatabaseStart
| DBDiskSpace
| Deadlock
| "Disconnect"
| GlobalAutoincrement
| GrowDB
| GrowLog
| GrowTemp
| LogDiskSpace
| MirrorFailover
| MirrorServerDisconnect
| RAISERROR
| ServerIdle
| TempDiskSpace
```

```
trigger-condition :
event_condition( condition-name ) {
=
| <
| >
| !=
| <=
| >=
} value
```

```
schedule-spec :
[ schedule-name ]
   { START TIME start-time | BETWEEN start-time AND end-time }
   [ EVERY period { HOURS | MINUTES | SECONDS } ]
   [ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]
   [ START DATE start-date ]
```

```
event-name : identifier
```

```
schedule-name : identifier
```

```
day-of-week : string

day-of-month : integer

value : integer

period : integer

start-time : time

end-time : time

start-date : date
```

パラメータ

CREATE EVENT clause

イベント名は識別子です。イベントには作成者が関連付けられます。これはイベントを作成したユーザであり、イベントハンドラはその作成者の権限で実行されます。これはストアドプロシージャの実行と同じです。他のユーザが所有するイベントを作成することはできません。

user-name

オプションで、システム内のユーザの名前を指定します。イベントの実行時に、`user-name` の権限で実行されます。このパラメータを指定しない場合、イベントは、そのイベントを作成したユーザの権限で実行されます。`user-name` とイベントの所有者を混同しないでください。イベントは所有者を持ちません。

OR REPLACE clause

OR REPLACE (CREATE OR REPLACE EVENT) を指定すると、イベントが作成されるか、同じ名前のイベントが置き換えられます。イベントが存在する場合、OR REPLACE 句の使用時にすべてのコメントが保持されますが、イベントのすべての既存の属性は削除されます。

TYPE clause

オプションの WHERE 句と一緒に TYPE 句を指定するか、または SCHEDULE を指定できます。

`event-type` には、リストで示したシステム定義のイベントタイプのうちいずれかを指定します。イベントタイプでは大文字と小文字を区別しません。この `event-type` がイベントをトリガする条件を指定するには、WHERE 句を使用します。

DiskSpace event types

データベースに DiskSpace タイプの 1 つに対応するイベントハンドラがある場合、データベースサーバは、使用するファイルに対応する各デバイスの空き領域を 30 秒おきにチェックします。

データベースが別々のドライブに複数の DB 領域を持っている場合、DBDiskSpace は各ドライブをチェックし、その中で最小の空き領域に基づいて動作します。

LogDiskSpace イベントタイプは、トランザクションログのロケーションと、トランザクションログミラーがあればそのロケーションをチェックし、最小の空き領域に基づいてレポートします。

TempDiskSpace イベントタイプは、テンポラリディスク領域の容量をチェックします。

適切なイベントハンドラが定義されている場合 (DBDiskSpace、LogDiskSpace、または TempDiskSpace)、データベースサーバは、データベースファイルに対応する各デバイスの空き領域を 30 秒おきにチェックします。同様に、システムイベントタイプ ServerIdle を処理するようにイベントが定義されている場合、データベースサーバは、前の 30 秒間に要求が処理されなかったときにハンドラを通知します。

データベースサーバの起動時に -fc オプションを指定して、データベースサーバでファイルシステムがいっぱいになった場合のコールバック関数を実装できます。

GlobalAutoincrement event type

このイベントは、GLOBAL AUTOINCREMENT の残りの値の数がその範囲の終わり 1% 未満になったときに、それぞれの挿入に対して発生します。このハンドラの一般的なアクションは、このイベントのパラメータとして指定されたテーブルと残りの値の数に基づいて、global_database_id オプションの新しい値を要求することです。

event_condition 関数と RemainingValues をこのイベントタイプの引数として使用できます。

ServerIdle event type

データベースに ServerIdle タイプのイベントハンドラがある場合、データベースサーバは 30 秒おきにサーバのアクティビティをチェックします。

Database mirroring event types

MirrorServerDisconnect イベントは、プライマリデータベースサーバからミラーサーバまたは監視サーバへの接続が失われたときに発生します。MirrorFailover イベントは、サーバがデータベースの所有権を取得したときに発生します。

WHERE clause

トリガ条件は、イベントが起動する条件を決定します。たとえば、トランザクションログが書き込まれるディスクの使用率が 80% を超えたときにアクションを実行する場合は、次のようなトリガ条件を使用します。

```
...
WHERE event_condition( 'LogFreePercent' ) < 20
...
```

event_condition 関数には、イベントタイプに有効な引数を指定してください。

複数の AND 条件を使って WHERE 句を構成できますが、OR 条件やその他の条件は使用できません。

event_condition 値の変数名を指定できます。

SCHEDULE clause

この句は、スケジュールされたアクションをいつ実行するかを指定します。時刻のシーケンスは、イベントハンドラに定義された関連するアクションのトリガ条件セットとして動作します。

1 つのイベントとそのハンドラに対し、複数のスケジュールを作成できます。これにより、複雑なスケジュールを実装できます。複数のスケジュールがある場合は、`schedule-name` を指定しなければなりませんが、スケジュールが 1 つしかない場合は、`schedule-name` を省略できます。

スケジュールされたイベントの定義に EVERY または ON が含まれる場合、そのイベントは反復されます。このどちらの予約語も使用されていない場合、イベントは最大でも 1 回しか実行されません。反復されないスケジュールイベントを作成する場合に、そのスケジュールの開始時刻が過ぎているときは、エラーになります。反復されないスケジュールされたイベントが経過すると、スケジュールは削除されますが、イベントハンドラは削除されません。

スケジュールされたイベントの時刻は、スケジュールの作成時に計算され、イベントハンドラの実行が完了したときに再計算されます。次のイベント時刻を計算するときには、イベントのスケジュールが調べられ、起動時刻が現在以降のスケジュールから次のスケジュール時刻が決定されます。9:00 から 5:00 の間に 1 時間ごとに実行され、実行に 65 分を要す

るイベントハンドラは、9:00、11:00、1:00、3:00、5:00 に実行されます。実行を重複させたい場合は、複数のイベントを作成する必要があります。

スケジュール定義に使用するサブ句は次のとおりです。

START TIME 句

イベントがスケジュールされた各日の最初のスケジュール時刻。`start-time` パラメータは文字列であり、`NOW()` などの式にはできません。START DATE を指定した場合、START TIME はその日付とそれ以降の毎日 (スケジュールに EVERY または ON が含まれる場合) を表します。START DATE を指定しない場合、START TIME は現在の日付 (時刻が経過していない場合) とそれ以降の毎日 (スケジュールに EVERY または ON が含まれる場合) となります。START TIME `start-time` 句は BETWEEN `start-time` AND '23:59:59' と同義です。

`start-time` の変数名を指定できます。

BETWEEN...AND 句

1 日のうち、スケジュールされた時刻が発生する範囲。`start-time` および `end-time` パラメータは文字列であり、`NOW()` などの式にはできません。START DATE を指定した場合、スケジュールされた時刻はその日が来るまで発生しません。

`start-time` および `end-time` の変数名を指定できます。

EVERY 句

連続してスケジュールするときのイベント発生の間隔。スケジュールされたイベントは、その日の START TIME より後、または BETWEEN...AND で指定した範囲内でのみ発生します。

`period` の変数名を指定できます。

ON 句

スケジュールされたイベントが発生する日のリスト。EVERY を指定した場合、デフォルトは毎日です。これらは曜日または日付として指定できます。

曜日は、Mon、Tues のようになります。Monday のような完全形も使用できます。使用言語が、英語でない場合、接続文字列でクライアントによって要求された言語でない場合、データベースサーバメッセージウィンドウに表示される言語でない場合は、完全形を使用します。

月の日数は 0 ～ 31 の性数値です。値 0 は月の末日を表します。

START DATE 句

スケジュールされたイベントが開始される日付。この値は文字列であり、`TODAY()` などの式にはできません。デフォルトは現在の日付です。

`start-date` の変数名を指定できます。

スケジュールされたイベントハンドラが完了するたびに、イベントに対してスケジュールされた次の時刻と日付を計算するために次のアクションが実行されます。

1. EVERY 句を使用した場合は、次のスケジュールされた時刻が現在の日付にあり、BETWEEN...AND 句で指定された範囲の終わりより前であるかどうかを調べます (範囲が指定された場合)。そうであれば、これが次のスケジュールされた時刻となります。
2. 次のスケジュールされた時刻が現在の日付にない場合は、イベントが実行される次の日付を調べ、その日付の START TIME、または BETWEEN...AND で指定された範囲の始まりを確認します。

ENABLE | DISABLE 句

イベントハンドラはデフォルトで有効になっています。DISABLE を指定すると、スケジュールされた時刻に達したりトリガ条件が発生しても、イベントハンドラは起動しません。TRIGGER EVENT 文は、無効に設定されているイベントハンドラを起動しません。

AT 句

SQL Remote の設定時に、リモートデータベースまたは統合データベースで AT 句を使用してイベントを処理するデータベースを制限する場合にのみ、この句を使用してください。

SQL Remote のイベント作成時に AT 句を使用しなかった場合は、すべてのデータベースでイベントが実行されます。統合データベースで実行される場合、この文はすでに抽出されているリモートデータベースには影響しません。

FOR 句

データベースミラーリングまたは読み込み専用スケールアウトシステムで、FOR 句を使用してイベントを処理するデータベースを制限する場合にのみ、この句を使用してください。

ミラーリングまたは読み込み専用スケールアウトシステムで、データベースのイベントを作成するときに FOR 句を使用しない場合は、プライマリサーバで実行されているデータベースだけが、そのイベントを実行します。次のサブ句がサポートされています。

FOR PRIMARY

イベントはプライマリサーバとして現在稼働しているサーバでのみ実行されます。デフォルトは PRIMARY サブ句です。

DatabaseStart イベントタイプで FOR PRIMARY 句を (または FOR 句を指定しないで) 使用すると、イベントはサーバがデータベースのプライマリサーバになったときに実行されます。

FOR ALL

イベントはシステムのすべてのサーバで実行されます。

DatabaseStart イベントタイプで FOR ALL 句を使用すると、イベントはいずれかのデータベースが起動すると実行されます。ミラーリングシステムでデータベースが起動したときにイベントが実行されなかった場合 (たとえば、イベントが作成される前にデータベースが実行されていた場合など)、そのイベントはフェイルオーバーのときに実行できます。たとえば、データベースミラーリングシステムを起動し、FOR ALL 句を使用して DatabaseStart イベントを作成して、プライマリサーバを停止すると、フェイルオーバーが発生します。この例では、新しいプライマリサーバでイベントが実行されます。以降のフェイルオーバーのときには DatabaseStart イベントは実行されません。

HANDLER 句

各イベントが 1 つのハンドラを持ちます。

備考

イベントは次の操作に使用できます。

アクションのスケジュール

データベースサーバは、時刻指定されたスケジュールに沿ってアクションを実行します。この機能を使用すると、バックアップ、妥当性検査、レポートテーブルへのデータ追加に使用するクエリなどのタスクをスケジュールして実行できます。

イベント処理アクション

データベースサーバは、事前に定義されたイベントが発生するとアクションを実行します。この機能を使用すると、ディスク領域の使用量が指定の割合を超えたときのディスク領域の制限などのタスクを、スケジュールして実行できます。実行中

にエラーが検出されなかった場合は、イベントハンドラアクションがコミットされ、エラーが検出された場合は、ロールバックされます。

イベント定義は 2 つの部分からなります。トリガ条件とは、ディスク領域の使用量が指定のスレッショルドを超えるなどの出来事をいいます。スケジュールとは、時刻のセットのことで、それぞれの時刻がトリガ条件の役割を果たします。トリガ条件が満たされると、イベントハンドラが実行されます。イベントハンドラには、複合文 (BEGIN ...END) の中に指定された 1 つまたは複数のアクションが含まれています。

トリガ条件やスケジュールを指定しない場合は、明示的な TRIGGER EVENT 文だけがイベントをトリガします。開発段階では TRIGGER EVENT を使ってイベントハンドラをテストし、テストの完了後にスケジュールまたは WHERE 句を追加することができます。

イベントエラーはデータベースサーバメッセージログに出力されます。

各イベントハンドラの実行後、エラーが発生しない場合、COMMIT が発生します。エラーが発生した場合は、ROLLBACK が発生します。

イベントハンドラがトリガされると、データベースサーバは event_parameter 関数を使用して、イベントをトリガした接続 ID などのコンテキスト情報をイベントハンドラに渡します。

イベントハンドラは別の接続で実行されますが、その接続はパーソナルデータベースサーバの最大接続数 10 には含められません。

所有者を指定することはできますが無視されます。イベントは所有者を持ちません。イベントは所有者を持たないため、2 つのイベントで同じ名前を持つことはできません。

注記

変数名を受け付けるパラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

新しいイベントを作成する場合は、MANAGE ANY EVENT または CREATE ANY OBJECT システム権限が必要です。

既存のイベントを置き換えるには、次のいずれかのシステム権限が必要です。

- MANAGE ANY EVENT システム権限
- ALTER ANY OBJECT システム権限
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

データベースサーバに対して、インクリメンタルバックアップを毎日午前 1 時に実行するように指示します。

```
CREATE EVENT IncrementalBackup
SCHEDULE
 START TIME '1:00 AM' EVERY 24 HOURS
HANDLER
BEGIN
 BACKUP DATABASE DIRECTORY 'c:¥¥backup'
 TRANSACTION LOG ONLY
 TRANSACTION LOG RENAME MATCH
END;
```

データベースサーバに対して、月曜日から金曜日までの午前 8 時から午後 6 時まで、1 時間ごとにトランザクションログのみの自動バックアップを実行するように指示します。

```
CREATE EVENT HourlyLogBackup
SCHEDULE hourly_log_backup
BETWEEN '8:00AM' AND '6:00PM'
EVERY 1 HOURS ON
   ('Monday','Tuesday','Wednesday','Thursday','Friday')
HANDLER
   BEGIN
      BACKUP DATABASE DIRECTORY 'c:¥¥database¥¥backup'
      TRANSACTION LOG ONLY
      TRANSACTION LOG RENAME
   END;
```

次にスケジュールされているイベントの実行時期を決定します。

```
SELECT DB_EXTENDED_PROPERTY( 'NextScheduleTime', 'HourlyLogBackup');
```

次の例は、event_condition 値の 1 つの変数を使用するイベントを作成し、次に、最初の event_condition 値に変数 @i1 を使用するイベントを作成します。

```
CREATE VARIABLE @i1 INTEGER;
SET @i1 = 10000;
```

```
CREATE EVENT LogNotifier
TYPE RAISERROR
WHERE event_condition ( 'ErrorNumber' ) <> @i1 AND event_condition
( 'ErrorNumber' ) <> 7
   HANDLER
      BEGIN
         MESSAGE 'LogNotifier message'
      END;
```

関連情報

BEGIN 文 [745 ページ]
ALTER EVENT 文 [643 ページ]
TRIGGER EVENT 文 [1365 ページ]
EVENT_PARAMETER 関数 [システム] [356 ページ]
EVENT_CONDITION 関数 [システム] [353 ページ]

1.4.4.63 CREATE EXISTING TABLE 文

リモートサーバ上の既存のオブジェクトを表す新しいプロキシテーブルを作成します。

構文

```
CREATE { EXISTING | VIRTUAL } TABLE [owner.]table-name
[ column-definition, ... ]
AT location-string [ ESCAPE CHARACTER character ]
```

```
column-definition :
column-name data-type NOT NULL
```

```
location-string :
  remote-server-name.[db-name].[owner].object-name
| remote-server-name;[db-name];[owner];object-name
```

パラメータ

CREATE { EXISTING | VIRTUAL } TABLE 句

CREATE EXISTING TABLE と CREATE VIRTUAL TABLE はセマンティック上同じです。CREATE VIRTUAL TABLE は、SAP HANA との互換性を保つために用意されています。

AT 句

AT 句は、リモートオブジェクトのロケーションを指定します。AT 句は、デリミタとしてセミコロン (;) をサポートします。セミコロンが `location-string` 文字列のどこかにある場合、そのセミコロンはフィールドデリミタです。セミコロンがない場合は、ピリオドがフィールドデリミタになります。この動作により、データベースと所有者の各フィールドにファイル名と拡張子を使用できます。ESCAPE CHARACTER 句を使用すると、アプリケーションはロケーション文字列内のこれらの区切り文字をエスケープできます。

CREATE TABLE または CREATE EXISTING 文のいずれかを使用してプロキシテーブルを作成するとき、AT 句に次の要素で構成されるロケーション文字列を含めます。

- リモートサーバの名前
- リモートカタログ

- リモート所有者またはスキーマ
- リモートテーブル名

ピリオドまたはセミコロンを使用して、ロケーション文字列を区切ります。ロケーション文字列はまた、データベースサーバがロケーション文字列を評価する際に展開される変数名を含むことができます。ロケーション文字列内の変数名は波括弧で囲みます。リモートサーバ名、カタログ名、所有者名、スキーマ名、またはテーブル名の一部として、ピリオド、セミコロン、波括弧などを用いることは非常にまれです。しかしながら、ロケーション文字列の中で、これらの区切り文字の 1 つまたはすべてを文字どおりに解釈すべき状況もあります。

> **注記**
>
> ESCAPE 句は、ロケーション句内の区切り文字をエスケープする場合のみ必要です。一般的に、プロキシテーブルの作成時は ESCAPE 句を省略できます。エスケープ文字には、任意の 1 バイト文字を指定できます。

AT 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます (たとえば、{variable-name})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、'access;{@myfile};;a1' を含む AT 句は、@myfile が SQL 変数で、@myfile 変数の現在の内容がプロキシテーブルの作成時に置き換えられることを示します。

備考

CREATE EXISTING TABLE 文は、外部のロケーションにあるテーブルに対応する、新しいローカルのプロキシテーブルを作成します。CREATE EXISTING TABLE 文は、CREATE TABLE 文の変形です。EXISTING キーワードを CREATE TABLE 文とともに使用すると、テーブルがすでにリモートに存在していることを指定して、そのメタデータをインポートするように指定できます。この構文によって、ユーザにとって可視のエンティティであるように、リモートテーブルが設定されます。このソフトウェアは、テーブルを作成する前に、外部ロケーションにテーブルが存在するかどうかを確認します。

オブジェクトが (ホストデータファイルまたはリモートサーバオブジェクトのどちらかとして) 存在しない場合、この文は拒否されてエラーメッセージが出力されます。

ホストデータファイルまたはリモートサーバテーブルのインデックス情報が抽出され、システムテーブル ISYSIDX のローを作成するために使用されます。この情報により、サーバ関係のインデックスとキーが定義されて、クエリオプティマイザがこのテーブルに存在する可能性のあるすべてのインデックスを考慮できるようになります。

参照整合性制約は、必要に応じてリモートロケーションに渡されます。

column-definitions が指定されていない場合、データベースサーバは、リモートテーブルから取得するメタデータからカラムリストを生成します。column-definitions が指定された場合、データベースサーバは column-definitions を検証します。カラム名、データ型、長さ、IDENTITY プロパティ、NULL プロパティについて、次の条件がチェックされます。

- カラム名が一致しなければなりません (大文字小文字は無視されます)。
- CREATE EXISTING TABLE 文のデータ型は、リモートロケーションのカラムのデータ型と一致するか、またはそのデータ型に変換可能でなければなりません。たとえば、ローカルカラムのデータ型が通貨として定義されているのに対して、リモートカラムのデータ型が数値である場合があります。
- 各カラムの NULL プロパティがチェックされます。ローカルカラムの NULL プロパティがリモートカラムの NULL プロパティと同じでない場合、警告メッセージが出力されますが、文はアボートしません。
- 各カラムの長さがチェックされます。CHAR、VARCHAR、BINARY、VARBINARY、DECIMAL、NUMERIC の各カラムの長さが一致しない場合は、警告メッセージが出力されますが、コマンドはアボートしません。
  CREATE EXISTING 文には、実際のリモートカラムリストのサブセットだけをインクルードすることができます。

権限

ユーザ本人が所有するプロキシテーブルを作成するには、CREATE PROXY TABLE システム権限が必要です。他のユーザが所有するプロキシテーブルを作成するには、CREATE ANY TABLE または CREATE ANY OBJECT のシステム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

Adaptive Server Enterprise によってサポートされます。`location-string` のフォーマットは実装依存です。

例

リモートサーバ server_a にある blurbs テーブルのプロキシテーブル blurbs を作成します。

```
CREATE EXISTING TABLE blurbs
( author_id ID not null,
copy text not null)
AT 'server_a.db1.joe.blurbs';
```

リモートサーバ server_a にある blurbs テーブルのプロキシテーブル blurbs を作成します。データベースサーバは、リモートテーブルから取得するメタデータからカラムリストを生成します。

```
CREATE EXISTING TABLE blurbs
AT 'server_a.db1.joe.blurbs';
```

リモートサーバ rda にある Employees テーブルのプロキシテーブル rda_employees を作成します。

```
CREATE EXISTING TABLE rda_employees
AT 'rda...Employees';
```

リモートサーバ server_a に存在する SQL 変数 table_name によって指定されたテーブルのプロキシテーブル rda_employees を作成します。

```
CREATE EXISTING TABLE rda_employees
AT 'rda...{table_name}';
```

ESCAPE CHARACTER 句を活用するため、以下の例を考えてみましょう。

1. test1.db と test2.db という名前の 2 つの SQL Anywhere データベースを作成します。

2. 両方のデータベースを同じサーバで起動します。

```
dbsrv17 -n escape_test test1.db test2.db
```

3. test2 に接続し、次のテーブルを作成します。

```
CREATE TABLE "table.with;fun{characters}"(c int);
INSERT INTO "table.with;fun{characters}" VALUES(100);
COMMIT;
```

4. 接続を切断し、test1 に接続します。
5. 次のとおり、test2 に対してリモートサーバを作成します。

```
CREATE SERVER test2_server CLASS 'saodbc' USING 'driver=SQL Anywhere
Native;eng=escape_test;dbn=test2';
CREATE EXTERNLOGIN localuser TO test2_server REMOTE LOGIN remoteuser
IDENTIFIED BY remotepwd;
```

> **注記**
>
> localuser は test1 へのログインに使用されたユーザ ID、一方で remotepwd は test2 へのログインに必要なリモートユーザ ID とパスワードにします。

6. ESCAPE CHARACTER 句を使用して、リモートの "table.with;fun{characters}" に対してプロキシテーブルを作成します。

```
CREATE EXISTING TABLE remtab AT 'test2_server;;;table.with!;fun!
{characters!}'ESCAPE CHARACTER'!';
```

または

```
CREATE EXISTING TABLE remtab AT 'test2_server...table!.with!;fun!
{characters!}'ESCAPE CHARACTER'!';
```

オプションで、プロキシテーブルに関するクエリを実行し、必要な結果セットを取得します。

```
SELECT c FROM remtab;
```

関連情報

CREATE TABLE 文 [952 ページ]

1.4.4.64 CREATE EXTERNLOGIN 文

リモートサーバとの通信に使用される代替ログイン名とパスワードを割り当てます。

構文

リモートサーバ用の外部ログインの作成

```
CREATE EXTERNLOGIN login-name
TO remote-server
[ REMOTE LOGIN remote-user [ IDENTIFIED BY remote-password ] ]
```

リモートサーバ用の外部ログインの作成 (構文の変数を含む)

```
CREATE EXTERNLOGIN USER string | variable
SERVER string | variable
[ REMOTE USER string | variable [ IDENTIFIED BY string | variable ] ]
```

ディレクトリアクセスサーバ用の外部ログインの作成

```
CREATE EXTERNLOGIN login-name
TO remote-server
```

ディレクトリアクセスサーバ用の外部ログインの作成 (構文の変数を含む)

```
CREATE EXTERNLOGIN USER string | variable
SERVER string | variable
```

パラメータ

login-name ローカルユーザログイン名を指定します。統合化ログインを使用する場合、`login-name` は Windows ユーザまたはグループのマッピング先となるデータベースユーザです。

TO 句 リモートサーバの名前を指定します。

REMOTE LOGIN 句 REMOTE LOGIN 句は、ローカルユーザ login-name に対して、remote-server 上にユーザアカウントを指定します。REMOTE LOGIN 句の値は 128 バイトまでに制限されます。

user-name

データベースユーザ名を指定します。リモートサーバでは、統合化ログインを使用する場合、`user-name` は Windows ユーザまたはグループのマッピング先となるデータベースユーザです。この値には文字列または変数を使用できます。

SERVER 句

リモートサーバまたはディレクトリアクセスサーバの名前を指定します。

REMOTE USER 句 (リモートサーバ)

REMOTE USER 句は、データベースユーザ名としてリモートサーバ上のユーザアカウントを指定します。REMOTE USER 句の値は 128 バイトまでに制限されます。

IDENTIFIED BY 句

リモートユーザのリモートパスワードを指定します。リモートユーザとリモートパスワードの組み合わせはリモートサーバで有効でなければなりません。この句はリモートサーバにのみ適用され、ディレクトリアクセスサーバには適用されません。

IDENTIFIED BY 句を省略すると、NULL のパスワードがリモートサーバに送信されます。ただし、IDENTIFIED BY "" (空の文字列) を指定すると、空の文字列がパスワードとして送信されます。

備考:リモートサーバ

CREATE EXTERNLOGIN は、リモートサーバとの通信に使用される代替ログイン名とパスワードを割り当てます。

まず、現在の既存ユーザの外部ログインを使用して、リモートサーバに接続しようとします。このユーザに外部ログインがない場合、DEFAULT LOGIN 認証を使用して接続しようとします。リモートサーバの作成時に DEFAULT LOGIN が設定されておらず、ユーザに外部ログインが定義されていない場合、現在の既存ユーザの ID とパスワードを使用して接続が試みられます。

REMOTE LOGIN 句が必要なのは、リモートサーバが接続にユーザ ID とパスワードを必要とする場合のみです。リモートログインなしの外部ログインがあると、DBA はリモートサーバにアクセスできるユーザを制御できます。また、リモートアクセスレイヤに対し、リモートサーバへのログインにユーザ ID とパスワードが不要であることを指示できます。

パスワードは内部的に暗号化された形式で保存されます。`remote-server` は、ISYSSERVER テーブルのエントリによってローカルサーバに認識させる必要があります。

パスワードの自動有効期限を使用するサイトでは、外部ログインのために、定期的なパスワードの更新を計画してください。

CREATE EXTERNLOGIN は、トランザクション内からは使用できません。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合は、データベースサーバからエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

備考 (ディレクトリアクセスサーバ)

デフォルトでは、データベースユーザがディレクトリアクセスサーバにアクセスするには、外部ログインが必要です。ただし、すべてのユーザが使用できるデフォルトの外部ログインを作成することで、この要件をなくすようディレクトリアクセスサーバを設定できます。

CREATE EXTERNLOGIN は、ディレクトリアクセスサーバへのアクセスに使用される外部ログインを割り当てます。

CREATE EXTERNLOGIN は、トランザクション内からは使用できません。

権限

MANAGE ANY USER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

この架空の例は、サーバ server1 に接続するときに、ローカルユーザ DBA を、パスワード Plankton を持つ sa ユーザにマッピングします。

```
CREATE EXTERNLOGIN DBA
TO server1
REMOTE LOGIN sa
IDENTIFIED BY Plankton;
```

関連情報

DROP EXTERNLOGIN 文 [1038 ページ]
CREATE SERVER 文 [914 ページ]

1.4.4.65 CREATE FUNCTION 文 [外部呼び出し]

ネイティブ関数または外部関数へのインタフェースを作成します。

構文

```
CREATE [ OR REPLACE ] FUNCTION [ owner.]function-name
( [ parameter, ... ] )
RETURNS data-type
[ SQL SECURITY { INVOKER | DEFINER } ]
[ [ NOT ] DETERMINISTIC ]
{ EXTERNAL NAME 'native-call'
```

```
  | EXTERNAL NAME 'c-call' LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32 |
C_ODBC64 }
  | EXTERNAL NAME 'clr-call' LANGUAGE CLR
  | EXTERNAL NAME 'perl-call' LANGUAGE PERL
  | EXTERNAL NAME 'php-call' LANGUAGE PHP
  | EXTERNAL NAME 'java-call' LANGUAGE JAVA
  | EXTERNAL NAME 'js-call' LANGUAGE JS }
```

```
parameter :
   [ IN ] parameter-name data-type [ DEFAULT expression ]
```

```
result-column :
column-name data-type
```

```
native-call :
[ system-configuration:]function-name@library-file-prefix[.{ so | dll} ]
```

```
system-configuration :
{ generic-operating-system | specific-operating-system } [ (processor-
architecture) ]
```

```
generic-operating-system :
{ Unix | Windows }
```

```
specific-operating-system :
{ AIX | HPUX | Linux | OSX | Solaris | WindowsNT }
```

```
processor-architecture :
{ 32 | 64 | ARM | IA64 | PPC | SPARC | X86 | X86_64 }
```

```
c-call :
[ operating-system:]function-name@library; ...
```

```
operating-system :
Unix
```

```
clr-call :
dll-name::function-name( param-type-1[, ... ] )
```

```
perl-call :
<file=perl-file> $sa_perl_return = perl-subroutine( $sa_perl_arg0[, ... ] )
```

```
php-call :
<file=php-file> print php-func( $argv[1][, ... ] )
```

```
java-call :
[package-name.]class-name.method-name java-method-signature
```

```
java-method-signature :
( [ java-field-descriptor, ... ] ) java-return-descriptor
```

```
java-field-descriptor および java-return-descriptor :
{ Z
  | B
  | S
```

```
  | I
  | J
  | F
  | D
  | C
  | V
  | [descriptor
  | Lclass-name;
}
```

```
js-call :
        <js-return-descriptor><file=js-object> js-func( js-field-descriptor[ ...])
```

```
js-field-descriptor および js-return-descriptor :
{ S
  | B
  | I
  | U
  | D
  | [descriptor
}
```

パラメータ

パラメータ名は、データベース識別子に対するルールに従って付けてください。パラメータ名は、有効な SQL データ型にします。また、キーワード IN のプレフィクスを付けて、引数が関数に値を提供する式であることを示してください。ただし、関数パラメータはデフォルトで IN です。

関数を実行するときに、すべてのパラメータを指定する必要はありません。CREATE FUNCTION 文の中に DEFAULT 値がある場合、不明のパラメータにデフォルト値を割り当てます。呼び出すときに引数を指定せず、デフォルトも設定されていない場合には、エラーが発生します。

OR REPLACE 句

CREATE OR REPLACE FUNCTION を指定すると、新しい関数が作成されるか、同じ名前の既存の関数が置き換えられます。この句によって、関数の定義は変更されますが、既存の権限は保持されます。

RETURNS 句

RETURNS 句は、関数の結果のデータ型を指定するときに使用します。RETURNS 句は、文の最初の句である必要があります。

SQL SECURITY 句

SQL SECURITY 句は、INVOKER (関数を呼び出すユーザ) または DEFINER (関数を所有するユーザ) として関数が実行されるかどうかを定義します。デフォルトは DEFINER です。外部呼び出しの場合、この句は、外部環境での修飾されていないオブジェクト参照に対して所有者のコンテキストを確立します。

SQL SECURITY INVOKER が指定されている場合、関数を呼び出すユーザごとに注釈を行う必要があるため、メモリ使用量が増えます。また、SQL SECURITY INVOKER が指定されている場合は、呼び出し側としても名前の決定が行われます。そのため、適切な所有者名で、すべてのオブジェクト名 (テーブル、プロシージャなど) を修飾します。たとえば、user1 が次の関数を作成するとします。

```
CREATE FUNCTION user1.myFunc()
   RETURNS INT
```

```
    SQL SECURITY INVOKER
    BEGIN
      DECLARE res INT;
      SELECT COUNT(*) INTO res FROM table1;
      RETURN res;
    END;
```

user2 がこの関数を実行しようとし、テーブル user2.table1 が存在しない場合、テーブルルックアップエラーが生じます。さらに、user2.table1 が存在する場合は、意図する user1.table1 の代わりにこのテーブルが使用されます。このような状況を防ぐには、文においてテーブル参照を修飾します (単なる table1 ではなく、user1.table1 とします)。

[NOT] DETERMINISTIC 句

この句は、関数が決定的か非決定的かを示すために使用します。この句が省略されると、関数の決定的な動作は指定されません (デフォルト)。

DETERMINISTIC と宣言された関数は、同じパラメータセットで呼び出されるたび、同じ値を返します。

NOT DETERMINISTIC と宣言された関数は、同じパラメータセットに対して同じ値を返すとはかぎりません。NOT DETERMINISTIC として宣言された関数は、クエリで呼び出されるたびに再評価されます。特定のパラメータセットに対して関数が返す結果が変化するとわかっている場合は、この句は必須です。

また、基本となるデータの修正などの関連する動作を伴う関数は、NOT DETERMINISTIC として宣言してください。たとえば、プライマリキー値を生成し、INSERT...SELECT 文で使用する関数は、次のように NOT DETERMINISTIC として宣言してください。

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
NOT DETERMINISTIC
BEGIN
  DECLARE keyval INTEGER;
  UPDATE counter SET x = x + increment;
  SELECT counter.x INTO keyval FROM counter;
  RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table;
```

特定の入力パラメータに対して常に同じ値を返す関数は、DETERMINISTIC として宣言できます。

EXTERNAL NAME native-call 句

EXTERNAL NAME 句は TEMPORARY 関数ではサポートされていませんので、注意してください。

LANGUAGE 属性なしで EXTERNAL NAME 句を使用する関数は、C などのプログラミング言語で記述されたネイティブ関数へのインタフェースを定義します。ネイティブ関数は、データベースサーバによってそのアドレス領域にロードされます。

`native-call` にはオペレーティングシステム、プロセッサ、ライブラリ、および関数の複数のセットを指定できるため、より精密に指定された設定が、それよりも精密さに欠けて定義された設定よりも優先されます。たとえば、`Solaris(X86_64):myfunc64@mylib.so` は `Solaris:myfunc64@mylib.so` よりも優先されます。

`system-configuration` がサポートされている構文で `system-configuration` を指定しないと、プロシージャがすべてのプラットフォームで稼働すると見なされます。UNIX は、AIX、HPUX、Linux、OS X、および Solaris の UNIX ベースオペレーティングシステムを意味します。Windows は、Windows オペレーティングシステムのすべてのバージョンを意味します。

specific-operating-system および processor-architecture の値は、SQL Anywhere サーバでサポートされるオペレーティングシステムおよびプロセッサです。

ライブラリ名 (library-file-prefix) にはファイル拡張子が付きます。この拡張子は通常、Windows では .dll、UNIX では .so です。拡張子がない場合、ライブラリに対するプラットフォーム固有のデフォルトのファイル拡張子が追加されます。例:

```
CREATE FUNCTION mystring( IN instr LONG VARCHAR )
RETURNS LONG VARCHAR
EXTERNAL NAME 'mystring@mylib.dll;Unix:mystring@mylib.so';
```

EXTERNAL NAME 句を、プラットフォーム固有のデフォルト値を使用して簡単に記述すると、次のようになります。

```
CREATE FUNCTION mystring( IN instr LONG VARCHAR )
RETURNS LONG VARCHAR
EXTERNAL NAME 'mystring@mylib';
```

呼び出されると、関数を含むライブラリがデータベースサーバの動作のアドレス領域にロードされます。ネイティブ関数は、データベースサーバの一部として実行されます。この場合、関数によって障害が引き起こされると、データベースサーバは停止します。このような動作のため、外部環境での関数のロードと実行には、LANGUAGE 属性を使用することをおすすめします。外部環境で関数によって障害が引き起こされた場合でも、データベースサーバは動作し続けます。

EXTERNAL NAME c-call 句

コンパイルされたネイティブ C 関数をデータベースサーバ内ではなく外部環境で呼び出すには、EXTERNAL NAME 句の後に LANGUAGE 属性を続けて、ストアドプロシージャまたは関数を定義します。

LANGUAGE 属性が指定されると、その関数を含むライブラリが外部プロセスによってロードされ、外部関数がその外部プロセスの一部として実行されます。この場合、関数が原因で障害が発生しても、データベースサーバは実行し続けます。

```
CREATE FUNCTION ODBCinsert(
  IN ProductName CHAR(30),
  IN ProductDescription CHAR(50)
)
RETURNS INT
EXTERNAL NAME 'ODBCexternalInsert@extodbc.dll'
LANGUAGE C_ODBC32;
```

EXTERNAL NAME clr-call 句

Microsoft .NET 関数を外部環境で呼び出すには、関数インタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE CLR 属性を指定します。

CLR ストアドプロシージャまたはファンクションの動作は、SQL ストアドプロシージャまたはファンクションと同じです。ただし、プロシージャまたはファンクションのコードは Microsoft C# または Visual Basic などの Microsoft .NET 言語で記述され、その実行はデータベースサーバの外側 (つまり別の Microsoft .NET 実行ファイル内) で行われます。

```
CREATE FUNCTION clr_interface(
    IN p1 INT,
    IN p2 UNSIGNED SMALLINT,
    IN p3 LONG VARCHAR)
RETURNS INT
EXTERNAL NAME 'CLRlib.dll::CLRproc.Run( int, ushort, string ) int'
LANGUAGE CLR;
```

EXTERNAL NAME perl-call 句

Perl 関数を外部環境で呼び出すには、関数インタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE PERL 属性を指定します。

Perl ストアドプロシージャまたは関数の動作は、SQL ストアドプロシージャまたは関数と同じです。ただし、プロシージャまたは関数のコードは Perl で記述され、その実行はデータベースサーバの外側 (つまり Perl 実行インスタンス内) で行われます。

```
CREATE FUNCTION PerlWriteToConsole( IN str LONG VARCHAR)
RETURNS INT
EXTERNAL NAME '<file=PerlConsoleExample>
    WriteToServerConsole( $sa_perl_arg0 )'
LANGUAGE PERL;
```

EXTERNAL NAME php-call 句

PHP 関数を外部環境で呼び出すには、関数インタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE PHP 属性を指定します。

PHP ストアドプロシージャまたは関数の動作は、SQL ストアドプロシージャまたは関数と同じです。ただし、プロシージャまたは関数のコードは PHP で記述され、その実行はデータベースサーバの外側 (つまり PHP 実行インスタンス内) で行われます。

```
CREATE FUNCTION PHPPopulateTable()
RETURNS INT
EXTERNAL NAME '<file=ServerSidePHPExample> ServerSidePHPSub()'
LANGUAGE PHP;
```

EXTERNAL NAME java-call 句

Java メソッドを外部環境で呼び出すには、関数インタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE JAVA 属性を指定します。

Java とのインタフェースとなるストアドプロシージャまたはファンクションの動作は、SQL ストアドプロシージャまたはファンクションと同じです。ただし、プロシージャまたはファンクションのコードは Java で記述され、その実行はデータベースサーバの外側 (つまり Java VM 内) で行われます。

```
CREATE FUNCTION HelloDemo( IN name LONG VARCHAR )
RETURNS INT
EXTERNAL NAME 'Hello.main([Ljava/lang/String;)I'
LANGUAGE JAVA;
```

Java メソッドの引数と戻り値の記述子には次の意味があります。

フィールドタイプ	Java データ型
B	byte
C	char
D	double
F	float
I	int
J	long

フィールドタイプ	Java データ型
L class-name;	クラス class-name のインスタンス。クラス名は、完全に修飾された名前です。また、名前内のドットは / に置き換える必要があります。例: **java/lang/String**。
S	short
V	void
Z	Boolean
[	配列の各次元ごとに 1 つ使用

EXTERNAL NAME js-call 句

JavaScript 関数を外部環境で呼び出すには、プロシージャインタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE JS 属性を指定します。

JavaScript ストアドプロシージャまたはファンクションの動作は、SQL ストアドプロシージャまたはファンクションと同じです。ただし、コードは JavaScript で記述され、コードの実行はデータベースサーバの外側 (つまり Node.js 実行インスタンス内) で行われます。

山括弧内の EXTERNAL NAME 文字列の先頭に、JavaScript 関数の戻り値のタイプを指定します。JavaScript では関数内の単純変数の参照による受け渡しが許可されていないため、左角括弧文字 ([) は、1 つの要素の配列が JavaScript ストアドプロシージャに渡されていることを示すために S、B、I、U、または D の文字を処理できます。この構文は、ストアドプロシージャ内の INOUT および OUT パラメータをサポートするために提供されます。

次に、関数定義の例を示します。

```
CREATE FUNCTION SimpleJSDemo(
    IN thousands INT,
    IN hundreds INT,
    IN tens INT,
    IN ones INT)
  RETURNS INT
  EXTERNAL NAME '<I><file=SimpleJSExample> SimpleJSFunction(IIII)'
  LANGUAGE JS;
```

JavaScript メソッドの引数と戻り値の記述子には次の意味があります。

フィールドタイプ	JavaScript データ型
S	文字列
B	ブール値
I	整数
U	符号なし整数
D	Double

備考

CREATE FUNCTION 文はデータベースに関数を作成します。所有者を指定することによって他のユーザの関数を作成できます。関数は、SQL 式の一部として呼び出されます。

複数の関数からテンポラリテーブルを参照する場合、テンポラリテーブル定義が矛盾していたり、テーブルを参照する文がキャッシュされていたりすると、問題が発生する可能性があります。

Mac OS X 10.11 上での実行時に EXTERNAL NAME 句を指定する場合、ロードする必要がある `.dylib` へのフルパスを指定するか、SQL Anywhere インストールの `lib64` ディレクトリに `.dylib` ファイルを配置する必要があります。

権限

ユーザ本人が所有する外部関数を作成するには、CREATE ANY PROCEDURE および CREATE EXTERNAL REFERENCE システム権限が必要です。

他のユーザが所有する外部関数を作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT システム権限および CREATE EXTERNAL REFERENCE システム権限が必要です。

既存の関数を置き換えるには、そのプロシージャを所有するか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

外部言語環境の CREATE FUNCTION は ANSI/ISO SQL 標準のコア機能ですが、ソフトウェアでサポートされている一部のコンポーネントはオプションの ANSI/ISO SQL 言語機能です。これらの機能のサブセットを次に示します。

- SQL SECURITY 句は、オプションの ANSI/ISO SQL 言語機能 T324 です。
- SQL 関数に LONG VARCHAR、LONG NVARCHAR、または LONG BINARY の値を渡す機能は、ANSI/ISO SQL 言語機能 T041 です。
- LANGUAGE JAVA のサポートは、オプションの ANSI/ISO SQL 言語機能 J621 です。
- CREATE TABLE または DROP TRIGGER などの文を使用して外部関数内でスキーマオブジェクトを作成または変更する機能は、言語機能 T653 です。
- CONNECT、EXECUTE IMMEDIATE、PREPARE、および DESCRIBE などの文を使用して外部関数内で動的 SQL 文を使用する機能は、言語機能 T654 です。

CREATE FUNCTION 文のいくつかの句は、標準にありません。これらを以下に示します。

- LANGUAGES 句での C_ESQL32、C_ESQL64、C_ODBC32、C_ODBC64、CLR、PERL、PHP のサポートは、標準にありません。

- external-call のフォーマットは実装依存です。
- 特定のルーチンパラメータのオプションの DEFAULT 句は、標準にありません。
- オプションの OR REPLACE 句は標準にありません。

Transact-SQL

外部ルーチンに対する CREATE FUNCTION は、Adaptive Server Enterprise でサポートされています。Adaptive Server Enterprise では、外部関数の外部環境 (SQL 言語機能 J621) として LANGUAGE JAVA のみがサポートされています。

関連情報

ALTER FUNCTION 文 [649 ページ]
CALL 文 [756 ページ]
CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
DROP FUNCTION 文 [1039 ページ]
GRANT 文 [1131 ページ]
SQL データ型 [124 ページ]

1.4.4.66 CREATE FUNCTION 文 [Web サービス]

HTTP 要求または SOAP over HTTP 要求を行う Web クライアント関数を作成します。

構文

```
CREATE [ OR REPLACE ] FUNCTION [ owner.]function-name ( [ parameter, ... ] )
RETURNS data-type
URL url-string
[ TYPE { http-type-spec-string | soap-type-spec-string } ]
[ HEADER header-string ]
[ CERTIFICATE certificate-string ]
[ CLIENTPORT clientport-string ]
[ PROXY proxy-string ]
[ SET protocol-option-string ]
[ SOAPHEADER soap-header-string ]
[ NAMESPACE namespace-string ]
```

```
http-type-spec-string :
HTTP[: { GET
   | POST[:MIME-type ]
   | PUT[:MIME-type ]
   | DELETE
   | HEAD
   | OPTIONS } ]
```

```
soap-type-spec-string :
```

```
SOAP[:{ RPC | DOC }
```

```
parameter :
  [ IN ] parameter-name datatype [ DEFAULT expression ]
```

```
url-string :
{ HTTP | HTTPS | HTTPS_FIPS }://[user:password@]hostname[:port][/path]
```

```
protocol-option-string : option-list [, option-list ...]
```

```
option-list :
  HTTP( http-option [ ;http-option ...] )
| SOAP( soap-option [ ;soap-option ...] )
| REDIR( redir-option [ ;redir-option ...] )
```

```
http-option :
  CHUNK={ ON | OFF | AUTO }
| EXCEPTIONS={ ON | OFF | AUTO }
| VERSION={ 1.0 | 1.1 }
| KTIMEOUT=number-of-seconds
```

```
soap-option :
  OPERATION=soap-operation-name
```

```
redir-option :
  COUNT=count
| STATUS=status-list
```

パラメータ

OR REPLACE clause

CREATE OR REPLACE FUNCTION を指定すると、新しい関数が作成されるか、同じ名前の既存の関数が置き換えられます。この句によって、関数の定義は変更されますが、既存の権限は保持されます。OR REPLACE 句をテンポラリ関数で使用することはできません。

function-name

関数の名前。

parameter-name

パラメータ名は、データベース識別子に対するルールに従って付けてください。これらは有効な SQL データ型を持つ必要があります。

パラメータがデフォルト値を持つ場合、これを指定する必要はありません。デフォルト値を持たないパラメータを指定する必要があります。

パラメータの先頭にキーワード IN を付けることで、この引数が関数に値を渡す式であることを示すことができます。ただし、関数パラメータはデフォルトで IN です。

data-type

パラメータのユーザ型。データ型を明示的に設定するか、%TYPE または %ROWTYPE 属性を指定し、データ型をデータベース内の別のオブジェクトのデータ型に設定します。%TYPE を使用して、テーブルまたはビュー内のカラムのデータ

型に設定します。%ROWTYPE を使用して、テーブルまたはビュー内のローから派生する複合データ型に設定します。ただし、テーブル参照変数に設定されている %ROWTYPE (`TABLE REF (table-reference-variable) %ROWTYPE`) を使用したデータ型の定義は許可されていません。

FLOAT、INT などの型指定のあるデータの転送をサポートするのは SOAP 要求のみです。HTTP 要求は文字列の転送のみサポートしているため、CHAR 型に制限されます。

RETURNS clause

次のいずれかを指定して、SOAP 関数または HTTP 関数の戻り値のデータ型を定義します。

- CHAR
- VARCHAR
- LONG VARCHAR
- TEXT
- NCHAR
- NVARCHAR
- LONG NVARCHAR
- NTEXT
- XML
- BINARY
- VARBINARY
- LONG BINARY

返される値は、HTTP 応答の本文です。HTTP ヘッダ情報は含まれません。ステータス情報などの他の情報が必要な場合は、関数ではなくプロシージャを使用します。

データ型は HTTP 応答の処理方法には影響しません。

URL clause

Web サービスの URI を指定します。オプションのユーザ名とパスワードのパラメータは、HTTP 基本認証に必要なクレデンシャルとして機能します。HTTP 基本認証は、ユーザとパスワードの情報を base-64 でエンコードし、HTTP 要求の Authentication ヘッダに渡します。この方法で指定すると、ユーザ名とパスワードは URL の一部として暗号化されずに渡されます。

HTTP:GET タイプの関数の場合、URL 句内および (自動的に生成されて) 関数に渡されたパラメータからの両方でクエリパラメータを指定できるようになりました。

```
URL 'http://localhost/service?parm=1
```

HTTPS_FIPS を指定すると、強制的に FIPS 認定ライブラリが使用されます。HTTPS_FIPS を指定したときに、FIPS 認定ライブラリがない場合は、代わりに FIPS 認定ではないライブラリが使用されます。

オペレーティングシステムの証明書ストアの証明書を使用するには、**`https://`** で始まる URL を指定します。

TYPE clause

Web サービス要求を行う場合に使用するフォーマットを指定します。SOAP が指定されている場合、または TYPE 句が含まれていない場合は、SOAP:RPC が使用されます。HTTP が指定されている場合は、HTTP:POST が使用されます。

TYPE 句では、HTTP:POST および HTTP:PUT の各タイプに MIME タイプを指定できます。HTTP:PUT が使用されている場合は、MIME タイプを指定する必要があります。`MIME-type` 指定は、Content-Type 要求ヘッダや操作モードの設定に使用することにより、1 つの呼び出しパラメータのみで要求の本文を設定できます。パラメータの置換後に Web サービスの関数呼び出しを行う場合は、パラメータがまったくなくなるか、1 つだけ残される場合があります。Web サービス関

数を (置換後に) NULL 値またはパラメータなしで呼び出すと、本文がなくコンテンツ長が 0 の要求になります。MIME タイプが指定されている場合は、要求内で単一の本文パラメータが送信されます。このため、アプリケーションは、コンテンツのフォーマットを MIME タイプに一致させる必要があります。

一般的な MIME タイプの例を次に示します。

- text/plain
- text/html
- text/xml

MIME タイプが指定されていない場合は、パラメータの名前と値 (複数のパラメータが可能) は、HTTP 要求の本文内で URL にエンコードされます。

TYPE 句のキーワードには次の意味があります。

'HTTP:GET'

デフォルトでは、URL で指定されたパラメータのエンコード用に、このタイプで MIME タイプ application/x-www-form-urlencoded が使用されます。

たとえば、クライアントが URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` から要求を送信すると、次の要求が生成されます。

```
GET /WebServiceName?arg1=param1&arg2=param2 HTTP/1.1
// <End of Request - NO BODY>
```

'HTTP:POST'

デフォルトでは、POST 要求の本文で指定されたパラメータのエンコード用に、このタイプで MIME タイプ application/x-www-form-urlencoded が使用されます。URL パラメータは、要求の本文に格納されます。

たとえば、クライアントが URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` から要求を送信すると、次の要求が生成されます。

```
POST /WebServiceName HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 19
arg1=param1&arg2=param2
// <End of Request>
```

'HTTP:PUT'

HTTP:PUT は HTTP:POST に似ていますが、HTTP:PUT タイプにはデフォルトのメディアタイプはありません。

次の例は、*%SQLANYSAMP17%*`¥SQLAnywhere¥HTTP¥put_data.sql` サンプルを実行しているデータベースサーバにデータをアップロードする、汎用のクライアント関数を設定する方法を示します。

```
CREATE OR REPLACE FUNCTION CPUT([data] LONG VARCHAR, resnm LONG VARCHAR,
mediatype LONG VARCHAR)
    RETURNS LONG BINARY
    URL 'http://localhost/resource/!resnm'
    TYPE 'HTTP:PUT:!mediatype';
SELECT CPUT('hello world', 'hello', 'text/plain' );
```

'HTTP:DELETE'

サーバにあるリソースを削除する Web サービスクライアント関数を設定できます。メディアタイプの指定はオプションです。

次の例は、put_data.sql サンプルを実行しているデータベースサーバからリソースを削除する、汎用のクライアント関数を設定する方法を示します。

```
CREATE OR REPLACE FUNCTION CDEL(resnm LONG VARCHAR, mediatype LONG VARCHAR)
    RETURNS LONG BINARY
    URL 'http://localhost/resource/!resnm'
    TYPE 'HTTP:DELETE:!mediatype';
SELECT CDEL('hello', 'text/plain' );
```

'HTTP:HEAD'

HEAD メソッドは GET メソッドと同じですが、サーバは本文を返しません。メディアタイプを指定できます。

```
CREATE OR REPLACE FUNCTION CHEAD(resnm LONG VARCHAR)
    RETURNS LONG BINARY
    URL 'http://localhost/resource/!resnm'
    TYPE 'HTTP:HEAD';
SELECT CHEAD( 'hello' );
```

'HTTP:OPTIONS'

OPTIONS メソッドは GET メソッドと同じですが、サーバは本文を返しません。メディアタイプを指定できます。このメソッドは CORS (クロスオリジンリソース共有) が可能です。

'SOAP:RPC'

このタイプは、Content-Type ヘッダを 'text/xml' に設定します。SOAP 操作と SOAP パラメータは、SOAP エンベロープの XML ドキュメントにカプセル化されます。

'SOAP:DOC'

このタイプは、Content-Type ヘッダを 'text/xml' に設定します。SOAP:RPC タイプに似ていますが、より多くのデータ型を送信できます。SOAP 操作と SOAP パラメータは、SOAP エンベロープの XML ドキュメントにカプセル化されます。

TYPE 句に MIME タイプを指定すると、その MIME タイプに Content-Type ヘッダが自動的に設定されます。

HEADER clause

HTTP Web サービスクライアント関数を作成する場合は、この句を使用して、HTTP 要求ヘッダのエントリを追加、変更、または削除します。ヘッダの仕様は、RFC2616 Hypertext Transfer Protocol の HTTP/1.1 および RFC822 Standard for ARPA Internet Text Messages に非常に近いものになっています。たとえば、HTTP ヘッダに指定できるのは印字可能な ASCII 文字のみで、大文字と小文字は区別されません。

ヘッダは、`header-name:value-name` ペアとして定義できます。各ヘッダとその値は、コロン (:) を使用して区切ります。このため、ヘッダにコロンは使用できません。各ペアを ¥n、¥x0d¥n、<LF> (改行)、または <CR><LF> (キャリッジリターンに続く改行) で区切ることで、複数のヘッダを定義できます。

ヘッダ内の連続する複数のスペースは、単一の空白文字に変換されます。

CERTIFICATE clause

安全な (HTTPS) 要求を行うには、HTTPS サーバの証明書の正味に使用される証明書 (または署名チェイン内の上位にある証明書) にクライアントがアクセスできる必要があります。必要な情報は、セミコロンで区切られたキーワード/値のペアの文字列で指定されます。次のキーワードを使用できます。

キーワード	省略形	説明
file		証明書のファイル名。オペレーティングシステムの証明書ストアにある証明書を使用する場合は * を指定します。certificate または certificate_name キーワードが指定されているときは指定できません。
certificate	cert	証明書自体 file または certificate_name キーワードが指定されているときは指定できません。
certificate_name	cert_name	データベースに保存される証明書の名前 file または certificate キーワードが指定されているときは指定できません。
company	co	証明書で指定された会社
unit		証明書で指定された会社の部署
name		証明書で指定された通称
skip_certificate_name_check		データベースサーバ証明書のチェックしないようにするには、ON を指定します。 **注記** この設定を使用すると、データベースサーバにより HTTP サーバが十分に認証されなくなるため、このオプションに ON を指定することはおすすめしません。

証明書は、HTTPS サーバに対する要求、または安全でないサーバから安全なサーバにリダイレクトされる可能性がある要求に対してのみ必要です。PEM でフォーマットされた証明書のみがサポートされています。

CLIENTPORT clause

HTTP クライアント関数が TCP/IP を使用して通信するポート番号を示します。これは "送信" TCP/IP 接続をフィルタするファイアウォール経由で接続するためのもので、この用途にかぎり推奨されています。単一のポート番号、ポート番号の範囲、または両方の組み合わせを指定できます。たとえば、`CLIENTPORT '85,90-97'` を指定できます。

PROXY clause

プロキシサーバの URI を指定します。クライアントがプロキシを介してネットワークにアクセスする場合に使用します。`proxy-string` は、通常 HTTP または HTTPS の url-string です。これは、通常ネットワーク管理者から取得する必要があるサイト固有の情報です。この句は、関数がプロキシサーバに接続し、そのプロキシサーバを介して Web サービスに要求を送信することを示します。たとえば、次の PROXY 句はプロキシサーバを `proxy.example.com` に設定します。

```
PROXY http://proxy.example.com
```

SET clause

HTTP、SOAP、および REDIR (リダイレクト) のプロトコル固有動作オプションを指定します。SET 句は 1 つしか使用できません。次の表に、サポートされている SET オプションを示します。CHUNK、EXCEPTIONS、VERSION、KTIMEOUT

は HTTP プロトコルに適用され、OPERATION は SOAP プロトコルに適用され、COUNT と STATUS は REDIR オプションに適用されます。REDIR オプションは、HTTP または SOAP プロトコルオプションとともに含めることができます。

CHUNK={ ON | OFF | AUTO }

(省略形 CH) この HTTP オプションでは、チャンクを使用するかどうかを指定します。チャンクを使用すると、HTTP メッセージを複数の部分に分割できます。可能な値は、ON (常にチャンクを使用)、OFF (チャンクは使用しない)、AUTO (内容が 8196 バイトを超えた場合にのみチャンクを使用、ただし自動生成されたマークアップを除く) です。たとえば、次の SET 句を使用すると、チャンクは有効になります。

```
SET 'HTTP(CHUNK=ON)'
```

CHUNK オプションを指定しないときの、デフォルトの動作は AUTO です。チャンクされた要求が、AUTO モードのときに、ステータスが 505 `HTTP Version Not Supported`、501 `Not Implemented`、または 411 `Length Required` で失敗した場合、クライアントはチャンクされた転送コーディングなしで要求をリトライします。

チャンクされた転送コーディングによる要求を HTTP サーバがサポートしない場合は、CHUNK オプションを OFF (チャンクを使用しない) に設定してください。

CHUNK モードは HTTP バージョン 1.1 からサポートされる転送エンコードです。CHUNK を ON に設定する場合、バージョン (VER) が 1.1 に設定される必要がありますが、デフォルトのバージョンとして 1.1 を使用する場合は、VER に何も設定しないでください。

EXCEPTIONS={ ON | OFF | AUTO }

(省略形 EX) この HTTP オプションは、ステータスコードの処理を制御することを可能にします。デフォルトは ON です。

ON または AUTO に設定すると、HTTP クライアント関数は HTTP 成功ステータスコード (1XX や 2XX) に対する応答を返し、すべてのコードは例外として SQLE_HTTP_REQUEST_FAILED を発生します。

```
SET 'HTTP(EXCEPTIONS=AUTO)'
```

OFF に設定すると、HTTP クライアント関数は、HTTP ステータスコードとは関係なく常に応答を返します。HTTP ステータスコードは使用できません。

HTTP ステータスコードと関連のない例外 (たとえば SQLE_UNABLE_TO_CONNECT_TO_HOST) は、EXCEPTIONS 設定とは関係なく、適宜発生します。

VERSION={ 1.0 | 1.1 }

(省略形 VER) この HTTP オプションでは、HTTP メッセージのフォーマットに使用する HTTP プロトコルのバージョンを指定できます。たとえば、次の SET 句は HTTP のバージョンを 1.1 に設定します。

```
SET 'HTTP(VERSION=1.1)'
```

可能な値は 1.0 と 1.1 です。VERSION が指定されていない場合は、次のようになります。

- CHUNK が ON に設定された場合、1.1 が HTTP バージョンとして使用される
- CHUNK が OFF に設定された場合、1.0 が HTTP バージョンとして使用される
- CHUNK が AUTO に設定された場合、クライアントが CHUNK モードで送信しているかどうかによって 1.0 か 1.1 が使用される

KTIMEOUT=number-of-seconds

(省略形 KTO) この HTTP オプションは、キープアライブタイムアウト条件を指定して、Web クライアント関数がキープアライブ HTTP/HTTPS 接続をインスタンス化し、一定期間キャッシュできるようにします。HTTP キープアライブ

接続をキャッシュするには、HTTP バージョンは 1.1 に設定し、KTIMEOUT は非ゼロ値に設定する必要があります。HTTP と HTTPS との間でパフォーマンスに大きな違いを感じている場合、HTTPS 接続にとって KTIMEOUT は特に役立ちます。データベース接続でキャッシュできるのは 1 つのキープアライブ HTTP 接続だけです。同じ URI を使用する Web クライアント関数へのその後の呼び出しでは、キープアライブ接続が再利用されます。このため、実行する Web クライアント呼び出しに含まれる URI のスキーマ、宛先ホスト、ポートが、キャッシュされた URI と一致することが必要です。また、HEADER 句で Connection: close を指定しない必要があります。KTIMEOUT が指定されないか、ゼロに設定されている場合、HTTP/HTTPS 接続はキャッシュされません。

OPERATION=soap-operation-name

(省略形 OP) この SOAP オプションでは、SOAP 操作の名前を指定します (作成している関数の名前と異なる場合)。OPERATION の値は、リモートファンクションコールの名前に似ています。たとえば、login という SOAP 操作を呼び出す accounts_login という関数を作成する場合は、次のように指定します。

```
CREATE FUNCTION accounts_login( name LONG VARCHAR, pwd LONG VARCHAR )
   RETURNS LONG BINARY
   SET 'SOAP(OPERATION=login)'
```

OPERATION オプションを指定しない場合、SOAP 操作の名前は、作成する関数の名前と一致する必要があります。

COUNT=count

(省略形 CNT) この REDIR オプションはリダイレクトの制御を可能にします。以下の "ステータス" を参照してください。

STATUS=status-list

(省略形 STAT) この REDIR オプションはリダイレクトの制御を可能にします。*302 Found* および *303 See Other* などの HTTP 応答ステータスコードを使用して、特に HTTP POST を実行した後に Web アプリケーションを新しい URI にリダイレクトします。たとえば、クライアント要求は次のようにすることができます。

```
GET /people/alice HTTP/1.1
Host: www.example.com
Accept: text/html, application/xhtml+xml
Accept-Language: en, de
```

web サーバ応答は次のようにすることができます。

```
HTTP/1.1 302 Found
Location: http://www.example.com/people/alice.en.html
```

応答では、クライアントは別の HTTP 要求を新しい URI に送信します。REDIR オプションを使用して、許可されるリダイレクションの最大数および自動的にリダイレクトする HTTP 応答ステータスコードを制御できます。

たとえば、`SET 'REDIR(COUNT=3; STATUS=301,307)'` では、最大制限の 3 リダイレクションおよび 301 と 307 ステータスのリダイレクションが可能です。302 や 303 などの他のリダイレクションステータスコードが受信されると、エラーが発行されます (SQLE_HTTP_REQUEST_FAILED)。

デフォルトのリダイレクション制限 `count` は 5 です。デフォルトでは、HTTP クライアント関数がすべての HTTP リダイレクションステータスコード (301、302、303、307) に応答して自動的にリダイレクトします。すべてのリダイレクションステータスコードを許可しないようにするには、`SET 'REDIR(COUNT=0)'` を使用します。このモードでは、リダイレクション応答はエラーになりません (SQLE_HTTP_REQUEST_FAILED)。その代わりに、HTTP ステータスおよび応答ヘッダによって結果セットが返されます。これにより、呼び出し側は *Location* ヘッダに含まれる URI に基づいて、条件付きで要求を再発行できます。

303 See Other ステータスを受信する POST HTTP メソッドを指定する Web サービス関数は、GET HTTP メソッドを使用してリダイレクト要求を発行します。

Location ヘッダには、絶対パスまたは相対パスのいずれかを含むことができます。HTTP クライアント関数がいずれかを処理します。このヘッダには、クエリパラメータを含めることもでき、これらのパラメータはリダイレクトされたロケーションに転送されます。たとえば、ヘッダに次のようなパラメータが含まれる場合、後続の GET または POST にこれらのパラメータが含まれます。

```
Location: alternate_service?a=1&b=2
```

上の例では、クエリパラメータは `a=1&b=2` です。

複数のオプション設定を 1 つの SET 句で組み合わせると、次の例のようになります。

```
CREATE FUNCTION accounts_login( name LONG VARCHAR, pwd LONG VARCHAR )
   RETURNS LONG BINARY
   SET 'HTTP( CHUNK=ON; VERSION=1.1 ), REDIR(COUNT=5;STATUS=302,303)'
   ...
```

次の例は、大文字と小文字から成る省略形の使用を示します。

```
CREATE FUNCTION accounts_login( name LONG VARCHAR, pwd LONG VARCHAR )
   RETURNS LONG BINARY
   SET 'HTTP( CH=ON; Ver=1.1 ), REDIR(CNT=5;Stat=302,303)'
   ...
```

SOAPHEADER clause

(SOAP フォーマットのみ) SOAP Web サービスを関数として宣言する場合は、この句を使用して 1 つ以上の SOAP 要求ヘッダエントリを指定します。SOAP ヘッダは、静的定数として宣言したり、代入パラメータメカニズムを使用して動的に設定したりできます (hd1、hd2 などに IN、OUT、または INOUT パラメータを宣言)。Web サービス関数では、1 つ以上の IN モード代入パラメータを定義できますが、INOUT または OUT 代入パラメータは定義できません。

次の例は、クライアントが、いくつかのヘッダエントリを代入パラメータを使用して送信し、応答 SOAP ヘッダデータを受信するよう指定する方法を示しています。

```
CREATE FUNCTION soap_client( IN hd1 LONG VARCHAR, IN hd2 LONG VARCHAR, IN hd3
LONG VARCHAR)
   RETURNS LONG BINARY
   URL 'localhost/some_endpoint'
   SOAPHEADER '!hd1!hd2!hd3';
```

NAMESPACE clause

(SOAP フォーマットのみ) この句は、SOAP:RPC 要求と SOAP:DOC 要求の両方に通常必要なメソッドネームスペースを示します。要求を処理する SOAP サーバは、このネームスペースを使用して、SOAP 要求メッセージ本文内のエンティティの名前を解釈します。ネームスペースは、Web サービスサーバから使用できる SOAP サービスの WSDL (Web Services Description Language) から取得できます。デフォルト値は、関数の URL のオプションのパスコンポーネントの直前までです。

`namespace-string` の変数名を指定できます。変数が NULL の場合、ネームスペースプロパティは無視されます。

備考

CREATE FUNCTION 文は、データベースに Web サービス関数を作成します。所有者名を指定すれば、別のユーザに対する関数を作成できます。

関数を実行するときに、すべてのパラメータを指定する必要はありません。CREATE FUNCTION 文の中に DEFAULT 値がある場合、不明のパラメータにデフォルト値を割り当てます。呼び出すときに引数を指定せず、デフォルトも設定されていない場合には、エラーが発生します。

パラメータ値は、要求の一部として渡されます。使用される構文は、要求のタイプによって決まります。HTTP:GET の場合、パラメータは URL の一部として渡されます。HTTP:POST 要求の場合、値は要求の本文に置かれます。SOAP 要求へのパラメータは、常に要求本文にバンドルされます。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

ユーザ本人が所有する関数を作成するには、CREATE PROCEDURE システム権限が必要です。

他のユーザが所有する関数を作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT のシステム権限が必要です。

既存の関数を置き換えるには、そのプロシージャを所有するか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

Adaptive Server Enterprise によってサポートされません。

例

1. 次の文は、localhost 上で実行されている get_picture サービスからイメージを返す cli_test1 という名前の関数を作成します。

```
CREATE FUNCTION cli_test1( image LONG VARCHAR )
RETURNS LONG BINARY
URL 'http://localhost/get_picture'
TYPE 'HTTP:GET';
```

2. 次の文は、URL `http://localhost/get_picture?image=widget` の HTTP 要求を作成します。

```
 SELECT cli_test1( 'widget' );
```

3. 次の文は代入パラメータを使用して、入力パラメータとして要求 URL を渡すことができるようにします。セキュア HTTPS 要求では、データベースに格納されている証明書を使用します。CHUNK モードの転送エンコードをオフにするために、SET 句が使用されます。

```
CREATE CERTIFICATE client_cert
FROM FILE 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥Certificates
¥¥rsaroot.crt';
CREATE FUNCTION cli_test2( image LONG VARCHAR, myurl LONG VARCHAR )
RETURNS LONG BINARY
URL '!myurl'
CERTIFICATE 'certificate_name=client_cert'
TYPE 'HTTPS:GET'
SET 'HTTP(CH=OFF)'
HEADER 'ASA-ID';
```

4. 次の文は、URL `http://localhost/get_picture?image=widget` の HTTP 要求を作成します。

```
CREATE VARIABLE a_binary LONG BINARY;
SET a_binary = cli_test2( 'widget', 'https://localhost/get_picture' );
SELECT a_binary;
```

5. 次の例は、NAMESPACE 句の変数を使用して関数を作成します。
   1. 次の文では、NAMESPACE 句の変数を作成します。

```
CREATE VARIABLE @ns LONG VARCHAR
SET @ns = 'http://wsdl.domain.com/';
```

   2. 次の文は、NAMESPACE 句の変数を使用する FtoC という名前の関数を作成します。

```
CREATE FUNCTION FtoC ( IN temperature LONG VARCHAR )
RETURNS LONG BINARY
URL 'http://localhost:8082/FtoCService'
TYPE 'SOAP:DOC'
NAMESPACE @ns;
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]

SQL データ型 [124 ページ]
ALTER FUNCTION 文 [649 ページ]
CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
DROP FUNCTION 文 [1039 ページ]
RETURN 文 [1272 ページ]

1.4.4.67 CREATE FUNCTION 文

データベースにユーザ定義 SQL 関数を作成します。

構文

```
CREATE [ OR REPLACE | TEMPORARY ] FUNCTION [ owner.]function-name
( [ parameter, ... ] )
RETURNS data-type
[ SQL SECURITY { INVOKER | DEFINER } ]
[ ON EXCEPTION RESUME ]
[ [ NOT ] DETERMINISTIC ]
compound-statement | AS tsql-compound-statement | AT location-string
```

```
parameter :
   [ IN ] parameter-name data-type [ DEFAULT expression ]
```

```
tsql-compound-statement :
sql-statement
sql-statement
 ...
```

パラメータ

OR REPLACE 句

CREATE OR REPLACE FUNCTION を指定すると、新しい関数が作成されるか、同じ名前の既存の関数が置き換えられます。関数が置き換えられた場合、関数の定義は変更されますが、既存の権限は保持されます。

OR REPLACE 句をテンポラリ関数で使用することはできません。

TEMPORARY キーワード

CREATE TEMPORARY FUNCTION を指定すると、作成した接続でのみ参照できる関数になり、接続を削除すると関数も自動的に削除されます。テンポラリ関数を明示的に削除することもできます。テンポラリ関数に対して ALTER、GRANT、または REVOKE は実行できません。また他の関数とは異なり、テンポラリ関数はカタログやトランザクションログに記録されていません。

テンポラリ関数は、作成者 (現在のユーザ) または指定された所有者の権限で実行されます。テンポラリ関数に所有者を指定できるのは次の場合です。

- テンポラリ関数が永続的なストアドプロシージャ内に作成された場合
- テンポラリ関数と永続的なストアドプロシージャとで所有者が同じ場合

テンポラリ関数の所有者を削除するには、テンポラリ関数を先に削除する必要があります。

読み込み専用のデータベースに接続するときに、テンポラリ関数の作成と削除を行うことができます。

OR REPLACE 句をテンポラリ関数で使用することはできません。

parameter-name

パラメータ名は、データベース識別子に対するルールに従って付けてください。パラメータ名は、有効な SQL データ型にします。また、キーワード IN のプレフィクスを付けて、引数が関数に値を提供する式であることを示してください。ただし、関数パラメータはデフォルトで IN です。

data-type

パラメータのユーザ型。データ型を明示的に設定するか、%TYPE または %ROWTYPE 属性を指定し、データ型をデータベース内の別のオブジェクトのデータ型に設定します。%TYPE を使用して、テーブルまたはビュー内のカラムのデータ型に設定します。%ROWTYPE を使用して、テーブルまたはビュー内のローから派生する複合データ型に設定します。

RETURNS 句

RETURNS 句は、関数の結果のデータ型を指定するときに使用します。RETURNS 句は、文の最初の句である必要があります。

SQL SECURITY 句

SQL SECURITY 句は、INVOKER (関数を呼び出すユーザ) または DEFINER (関数を所有するユーザ) として関数が実行されるかどうかを定義します。デフォルトは DEFINER です。

ON EXCEPTION RESUME 句

Transact-SQL のようなエラー処理を使用します。

[NOT] DETERMINISTIC 句

この句は、関数が決定的か非決定的かを示すために使用します。この句が省略されると、関数の決定的な動作は指定されません (デフォルト)。

DETERMINISTIC と宣言された関数は、同じパラメータセットで呼び出されるたび、同じ値を返します。

NOT DETERMINISTIC と宣言された関数は、同じパラメータセットに対して同じ値を返すとはかぎりません。NOT DETERMINISTIC として宣言された関数は、クエリで呼び出されるたびに再評価されます。特定のパラメータセットに対して関数が返す結果が変化するとわかっている場合は、この句を使用してください。

また、基本となるデータの修正などの関連する動作を伴う関数は、NOT DETERMINISTIC として宣言してください。たとえば、プライマリキー値を生成し、INSERT...SELECT 文で使用される関数の例を示します。この関数は NOT DETERMINISTIC として宣言されます。参照されるテーブルは架空のものです。

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
NOT DETERMINISTIC
BEGIN
  DECLARE keyval INTEGER;
  UPDATE counter SET x = x + increment;
  SELECT counter.x INTO keyval FROM counter;
  RETURN keyval
END
```

```
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table;
```

特定の入力パラメータに対して常に同じ値を返す関数は、DETERMINISTIC として宣言できます。

compound-statement

BEGIN と END で囲まれ、セミコロンで区切られた SQL 文のセット。

AS 句

`tsql-compound-statement` は Transact-SQL 文のバッチです。

AT 句

`location-string` で指定されたリモート関数に対して、現在のデータベース上にプロキシ関数を作成します。AT 句では、`location-string` 内のフィールドデリミタとしてセミコロン (;) を使用できます。セミコロンがない場合は、ピリオドがフィールドデリミタになります。セミコロンを使用すると、データベースフィールドと所有者フィールドでファイル名と拡張子を使用できます。

AT 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます ({`variable-name`})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、`'bostonase.master.dbo.{@myfunction}'` を含む AT 句は、`@myfunction` が SQL 変数で、`@myfunction` 変数の現在の内容がリモートプロシージャの使用時に置き換えられることを示します。

プロキシ関数は、DECIMAL、NUMERIC、LONG VARCHAR、LONG NVARCHAR、LONG BINARY、XML、または空間データ型を除くすべてのデータ型を返すことができます。

備考

CREATE FUNCTION 文はデータベースに関数を作成します。所有者名を指定すれば、別のユーザに対する関数を作成できます。権限による制御はありますが、関数は他の非集合関数とまったく同様に使用できます。

関数を実行するときに、すべてのパラメータを指定する必要はありません。CREATE FUNCTION 文の中に DEFAULT 値がある場合、不明のパラメータにデフォルト値を割り当てます。呼び出すときに引数を指定せず、デフォルトも設定されていない場合には、エラーが発生します。

SQL SECURITY INVOKER が指定されている場合は、プロシージャを呼び出すユーザごとに注釈を行う必要があるためメモリ使用量が増えます。また、SQL SECURITY INVOKER が指定されている場合は、呼び出し側として名前の決定が行われます。このため、必ずすべてのオブジェクト名 (テーブル、プロシージャなど) を該当する所有者で修飾してください。

NOT DETERMINISTIC と宣言されないかぎり、すべての関数は決定的として扱われます。決定的関数は、同じパラメータに対して一貫した結果を返し、副次効果はありません。つまり、データベースサーバは、同じパラメータを持つ同じ関数が連続して 2 回呼び出されている場合に、どちらの呼び出しでも同じ結果が返され、クエリのセマンティックに不要な弊害は生じないものと見なします。

関数が結果セットを返す場合、出力パラメータを設定したり戻り値を返したりすることはできません。

権限

ユーザ本人が所有する関数を作成するには、CREATE PROCEDURE システム権限が必要です。

他のユーザが所有する関数を作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT のシステム権限が必要です。

外部関数を作成するには、CREATE EXTERNAL REFERENCE システム権限も必要です。

テンポラリ関数の作成には権限は必要ありません。

既存の関数を置き換えるには、その関数の所有者であるか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット、テンポラリ関数にも適用。

標準

ANSI/ISO SQL 標準

CREATE FUNCTION は ANSI/ISO SQL 標準のコア機能ですが、ソフトウェアでサポートされている一部のコンポーネントはオプションの SQL 言語機能です。これらの機能のサブセットを次に示します。

- SQL SECURITY 句は、オプションの言語機能 T324 です。
- SQL 関数に LONG VARCHAR、LONG NVARCHAR、または LONG BINARY の値を渡す機能は、言語機能 T041 です。
- CREATE TABLE または DROP TRIGGER などの文を使用して SQL 関数内でスキーマオブジェクトを作成または変更する機能は、言語機能 T651 です。
- EXECUTE IMMEDIATE、PREPARE、および DESCRIBE などの文を使用して SQL 関数内で動的 SQL 文を使用する機能は、言語機能 T652 です。

CREATE FUNCTION 文のいくつかの句は、標準にありません。これらを以下に示します。

- TEMPORARY 句。
- ON EXCEPTION RESUME 句。
- 特定のルーチンパラメータ用のオプションの DEFAULT 句。
- AS 句を使用した Transact-SQL 関数の指定。
- オプションの OR REPLACE 句。

Transact-SQL

CREATE FUNCTION は、Adaptive Server Enterprise でサポートされています。Adaptive Server Enterprise では、関数パラメータにオプションの IN キーワードはサポートされていません。

例

次の関数は、firstname 文字列と lastname 文字列を連結します。

```
CREATE FUNCTION fullname(
   firstname CHAR(30),
   lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
   DECLARE name CHAR(61);
   SET name = firstname || ' ' || lastname;
   RETURN (name);
END;
```

次の例は、最初の例で作成された fullname 関数を置き換えます。この関数が置き換えられると、ローカル変数 `name` が削除されます。

```
CREATE OR REPLACE FUNCTION fullname(
   firstname CHAR(30),
   lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
   RETURN ( firstname || ' ' || lastname );
END;
```

次の例は、fullname 関数の使用法を示します。

2 つの提供された文字列からフルネームを戻します。

```
SELECT fullname ( 'joe', 'smith' );
```

fullname('joe', 'smith')
joe smith

全従業員の名前をリストします。

```
SELECT fullname ( GivenName, Surname )
FROM GROUPO.Employees;
```

fullname (GivenName, Surname)
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
...

次の文は、Transact-SQL 構文を使用します。

```
CREATE FUNCTION DoubleIt( @Input INT )
RETURNS INT
AS
BEGIN
  DECLARE @Result INT
  SELECT @Result = @Input * 2
  RETURN @Result
```

```
END;
```

文 `SELECT DoubleIt( 5 );` は、`10` の値を返します。

次の例は、fullname と呼ばれる関数を作成し、%TYPE 属性を使用して firstname および lastname パラメータのデータ型を Employees テーブルの Surname および Givenname カラムのデータ型に設定します。

```
CREATE OR REPLACE FUNCTION fullname(
   firstname Employees.Surname%TYPE,
   lastname Employees.GivenName%TYPE )
RETURNS LONG VARCHAR
BEGIN
   RETURN ( firstname || ' ' || lastname );
END;
SELECT fullname ( Surname, GivenName )FROM GROUPO.Employees;
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
ALTER FUNCTION 文 [649 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
BEGIN 文 [745 ページ]
CREATE PROCEDURE 文 [891 ページ]
DROP FUNCTION 文 [1039 ページ]
RETURN 文 [1272 ページ]
SQL データ型 [124 ページ]

1.4.4.68 CREATE INDEX 文

指定したテーブル、またはマテリアライズドビューにインデックスを作成します。

構文

テーブルへのインデックスの作成

```
CREATE [ VIRTUAL ] [ UNIQUE ] [ CLUSTERED ] INDEX [ IF NOT EXISTS ] index-name
ON [ owner.]table-name
    ( column-name [ ASC | DESC ], ...
      | function-name ( argument, [ ... ] )AS column-name )
| [WITH NULLS [ NOT ] DISTINCT ]
[ { IN | ON } dbspace-name ]
[ FOR OLAP WORKLOAD ]
```

マテリアライズドビューへのインデックスの作成

```
CREATE [ VIRTUAL ] [ UNIQUE ] [ CLUSTERED ] INDEX [ IF NOT EXISTS ] index-name
  ON [ owner.]materialized-view-name
```

```
    ( column-name [ ASC | DESC ], ...)
| [ WITH NULLS NOT DISTINCT ]
[ { IN | ON } dbspace-name ]
[ FOR OLAP WORKLOAD ]
```

パラメータ

VIRTUAL 句

VIRTUAL キーワードは主に、インデックスコンサルタントによって使用されます。実行プランがインデックスコンサルタントによって評価されている間と、PLAN 関数が使用されている場合に、仮想インデックスは実際の物理的なインデックスのプロパティを模倣します。仮想インデックスを PLAN 関数と一緒に使用すると、実際のインデックスを作成するという時間とリソースを消費する作業をせずに、インデックスのパフォーマンスへの影響を調べることができます。

仮想インデックスは、他の接続からは参照できず、その接続が閉じたときに削除されます。仮想インデックスは、クエリの実際の実行プランを評価しているときは使用されないため、パフォーマンスの妨げにはなりません。

仮想インデックスは、最大 4 カラムまでを扱えます。

仮想インデックスの作成 (CREATE VIRTUAL INDEX 文) は、BEGIN PARALLEL WORK 文の中ではサポートされません。

UNIQUE 句

UNIQUE 属性によって、インデックス内のすべてのカラムで同じ値を持つローがテーブルまたはマテリアライズドビュー内に複数存在しないようにします。UNIQUE を指定して、WITH NULLS NOT DISTINCT を指定しない場合は、各インデックスキーがユニークであるか、少なくとも 1 つのカラムで NULL を持つ必要があります。たとえば、2 つのエントリ ('a', NULL) と ('a', NULL) は、それぞれユニークと見なされます。

UNIQUE...WITH NULLS NOT DISTINCT を指定する場合は、インデックスキーが NULL 値に関係なくユニークである必要があります。たとえば、2 つのエントリ ('a', NULL) と ('a', NULL) は、同等でありユニークとは見なされません。

一意性制約とユニークインデックスには違いがあります。ユニークインデックスのカラムは NULL を使用できますが、一意性制約のカラムには使用できません。外部キーは、プライマリキーまたは一意性制約は参照できますが、ユニークインデックスは参照できません。ユニークインデックスは NULL の複数のインスタンスを含むことがあるからです。

プライマリキーや、一意性制約のカラムには FLOAT や DOUBLE などの概数値データ型を使用しないことをおすすめします。概数値データ型は、算術演算後の丸め誤差がでます。

ユニークインデックスには空間カラムは設定できません。

CLUSTERED 句

CLUSTERED 属性を使用すると、ローはインデックスのキーの順序で格納されます。データベースサーバはキーの順序を保持しようとしますが、完全なクラスタは保証されません。

クラスタドインデックスが存在すると、LOAD TABLE 文はインデックスキーの順序でローを挿入し、INSERT 文はキーの順序の定義に従って、隣接するローがあるページに新しいローを配置しようとします。

IF NOT EXISTS 句

IF NOT EXISTS 属性を指定し、指定したインデックスがすでに存在する場合、変更は行われず、エラーも返されません。

ASC | DESC 句

降順 (DESC) が明示的に指定されている場合以外、カラムは昇 (増加) 順で格納されます。昇順と降順の ORDER BY の両方に対してインデックスを使います。これはインデックスが昇順であっても降順であっても同様です。しかし、昇順と

降順属性を混在させて ORDER BY を実行する場合、インデックスを使うのは同じ昇順と降順属性を使ってインデックスが作成されたときだけです。

function-name

function-name 句は、関数でインデックスを作成します。この句は、宣言済みのテンポラリテーブルまたはマテリアライズドビューでは使用できません。

この形式の CREATE INDEX 文は、次の操作を実行する際に便利です。

1. 計算カラム `column-name` をテーブルに追加します。カラムは、指定された関数である COMPUTE 句と指定された引数を使用して定義されます。指定できる関数の種類に関する制限については、CREATE TABLE 文の COMPUTE 句の説明を参照してください。カラムのデータ型は、関数の結果タイプに基づきます。
2. テーブルの既存のローに対して計算カラムを移植します。
3. カラムでインデックスを作成します。
   このインデックスを削除すると、関連する計算カラムも削除されます。

関数のインデックスの作成 (CREATE INDEX... ON `function-name` 文) は、BEGIN PARALLEL WORK 文の中ではサポートされません。

IN | ON 句

デフォルトで、インデックスはそのテーブルまたはマテリアライズドビューと同じデータベースファイルの中に置かれます。インデックスを入れる DB 領域名を指定して、別のデータベースファイルにインデックスを入れることができます。この機能が便利なのは、主に大規模なデータベースがファイルサイズの制限を回避する場合、または複数のディスクデバイスを使用してパフォーマンスの改善が望める場合です。

新しいインデックスで物理インデックスを既存の論理インデックスと共有できる場合、IN 句は無視されます。

WITH NULLS NOT DISTINCT 句

この句はインデックスを UNIQUE と宣言した場合にのみ指定でき、インデックスキーで NULL がユニークでないことを指定できます。詳細については、UNIQUE 句を参照してください。

FOR OLAP WORKLOAD 句

FOR OLAP WORKLOAD を指定すると、データベースサーバは特定の最適化を実行し、キーに関する統計情報を収集して、OLAP の負荷に関するパフォーマンスを向上させることができます。パフォーマンスの向上は、optimization_workload を OLAP に設定すると特に顕著になります。

備考

インデックスによってデータベースのパフォーマンスを改善できます。データベースサーバは物理インデックスと論理インデックスを使用します。物理インデックスは、インデックスがディスクに保存されるときの実際のインデックス構造です。論理インデックスは、物理インデックスへの参照です。既存のインデックスに対する物理属性と同じインデックスを作成する場合、データベースサーバは、既存の物理インデックスを共有する論理インデックスを作成します。通常、ユーザが作成したインデックスは論理インデックスと見なされます。論理インデックスの実装に必要な場合、データベースサーバは物理インデックスを作成します。また、同じ物理インデックスを複数の論理インデックスで共有できます。

CREATEINDEX 文は、指定されたテーブルまたはマテリアライズドビューの特定のカラムにソートされたインデックスを作成できます。インデックスは、データベースに対して発行されたクエリのパフォーマンスを向上させたり、ORDER BY 句を使用してクエリをソートするときに自動的に使用されます。一度インデックスを作成すると、そのインデックスを検証する (VALIDATE INDEX) とき、変更する (ALTER INDEX) とき、削除する (DROPINDEX) とき以外は、再び参照されることはありません。

インデックスの所有者

CREATE INDEX 文ではインデックスの所有者を指定できません。常にテーブルまたはマテリアライズドビューの所有者がインデックスの所有者となります。

テーブルのインデックス

インデックスは、ベーステーブルだけでなく、ローカルとグローバルの両方のテンポラリテーブルに対して作成できます。

CREATE INDEX 文を BEGIN PARALLEL WORK 文の中で使用する場合には、ベーステーブルのインデックスしか作成できません。

ビューのインデックス

マテリアライズドビュー上にはインデックスを作成できますが、通常のビューには作成できません。

インデックス名のスペース

各インデックスには、特定のテーブルまたはマテリアライズドビューに対してユニークな名前を与えます。

排他的使用

CREATE INDEX は、別の接続によって現在使用されているテーブルまたはマテリアライズドビューにこの文が影響する場合は常に阻止されます。

自動的に作成されたインデックス

データベースサーバは、プライマリキー、外部キー、一意性制約のインデックスを自動的に作成します。これら自動作成インデックスは、テーブルと同じデータベースファイルに保持されます。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

CREATE INDEX 文を BEGIN PARALLEL WORK 文の中で使用する場合、細分化レベルはテーブルレベルです。同じテーブルに対してインデックスを作成する CREATE INDEX 文は、BEGIN PARALLEL WORK の中であっても、逐次的に実行されます。

権限

テーブルのインデックスを作成するには、テーブルの所有者であるか、次のいずれかの権限を持っている必要があります。

- そのテーブルに対する REFERENCES 権限
- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限

マテリアライズドビューのインデックスを作成するには、マテリアライズドビューの所有者であるか、次のいずれかの権限を持っている必要があります。

- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限

関連する動作

ほとんどの場合はオートコミットです。auto_commit_on_create_local_temp_index オプションが Off に設定されている場合、ローカルテンポラリテーブルのインデックスを作成する前にコミットは実行されません。関数 (暗黙的な計算カラム) でインデックスを作成すると、チェックポイントが発生します。

カラムの統計情報が更新されます (統計情報が存在しない場合は作成されます)。

CREATE INDEX 文の開始時と終了時の限られた時間においてのみ、テーブルに排他ロックが適用されます。CREATE INDEX 文の実行の途中に、テーブルに共用ロックが適用されます。文の実行全体でテーブルへの排他アクセスを必要とする場合は、その排他ロックが一時的なものであっても、CREATE INDEX と同時には実行できません。ただし、共有アクセスのみを必要とする文は、CREATE INDEX 文の実行の途中にのみテーブル上で実行できます。

標準

ANSI/ISO SQL 標準

標準になし。

例

Employees テーブルで 2 カラムのインデックスを作成します。

```
CREATE INDEX employee_name_index
ON GROUPO.Employees
( Surname, GivenName );
```

ProductID カラム用に SalesOrderItems テーブル上でインデックスを作成します。

```
CREATE INDEX item_prod
ON GROUPO.SalesOrderItems
( ProductID );
```

SORTKEY 関数を使用して、Products テーブルの Description カラムにインデックスを作成し、Russian 照合に従ってソートします。関連動作として、この文は計算カラム desc_ru をテーブルに追加します。この例を成功させるには、Products テーブルに対する SELECT 権限も必要です。

```
CREATE INDEX ix_desc_ru
ON GROUPO.Products (
 SORTKEY( Description, 'rusdict' )
 AS desc_ru );
```

関連情報

DROP INDEX 文 [1040 ページ]
CREATE STATISTICS 文 [942 ページ]

1.4.4.69 CREATE LDAP SERVER 文

LDAP サーバ設定オブジェクトを作成します。

構文

```
CREATE LDAP SERVER ldapua-server-name
[ ldapua-server-attribs ... ]
[ WITH ACTIVATE ]
```

```
ldapua-server-attribs :
SEARCH DN search-dn-attributes ...
| AUTHENTICATION URL { 'url-string' | NULL }
| CONNECTION TIMEOUT timeout-value
| CONNECTION RETRIES retry-value
| TLS { ON | OFF }
```

```
search-dn-attributes :
URL { 'url-string' | NULL }
| ACCESS ACCOUNT { 'dn-string' | NULL }
| IDENTIFIED BY ( 'password' | NULL }
| IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
```

パラメータ

SEARCH DN 句

SEARCH DN 句のパラメータにはデフォルト値がありません。

URL

この句を使用して、ホストを (名前または IP アドレスで) 指定したり、ポート番号を指定したり、指定されたユーザ ID の LDAP 識別名 (DN) のルックアップを行うために実行する検索を指定します。`url-string` は正しい LDAP URL 構文であることが検証されてから ISYSLDAPSERVER に格納されます。この文字列の最大サイズは 1024 バイトです。

`url-string` のフォーマットは、LDAP URL 標準に準拠している必要があります。LDAP 標準仕様 を参照してください。

ACCESS ACCOUNT

この句を使用して、データベースサーバによって LDAP サーバに接続するために使用される DN を指定します。これは SQL Anywhere ユーザではなく、LDAP サーバへのログイン用に LDAP サーバで作成されたユーザです。このユーザは、SEARCH DN URL 句で指定される場所にあるユーザ ID によって DN を検索するために、LDAP サーバ内でパーミッションを得る必要があります。この文字列の最大サイズは 1024 バイトです。

IDENTIFIED BY

この句を使用して、ACCESS ACCOUNT で識別されるユーザに関連付けられたパスワードを指定します。最大サイズは 255 バイトで、NULL には設定できません。

IDENTIFIED BY ENCRYPTED

この句を使用して、暗号化形式で提供され、ディスクのいずれかの場所に保存されたバイナリ値である、ACCESS ACCOUNT で識別されるユーザに関連付けられたパスワードを指定します。バイナリの最大サイズは 289 バイト

で、NULL には設定できません。IDENTIFIED BY ENCRYPTED により、パスワードを既知にせずに取得および使用できます。

AUTHENTICATION URL 句

この句を使用して、ホストを名前または IP アドレスで識別したり、ユーザ認証に使用する LDAP サーバのポート番号を識別したりする `url-string` を指定します。以前の DN 検索およびユーザパスワードから取得したユーザの DN は、新しい接続を認証 URL にバインドするために使用されます。LDAP サーバへの正常な接続は、接続ユーザの ID の証明とみなされます。このパラメータにはデフォルト値はありません。この文字列に対するサイズ制限については、SYSLDAPSERVER システムビューを参照してください。

CONNECTION TIMEOUT 句

この句を使用して、DN の検索と認証の両方に対する LDAP サーバへの接続タイムアウトをミリ秒単位で指定します。デフォルト値は 10 秒です。

CONNECTION RETRIES 句

この句を使用して、DN の検索と認証の両方に対する LDAP サーバへの接続再試行回数をミリ秒単位で指定します。値の有効範囲は 1 ～ 60 です。デフォルトは 3 です。

TLS 句

この句を使用して、DN の検索と認証の両方に対する LDAP サーバへの接続時の TLS プロトコルの使用を指定します。有効な値は ON と OFF です。デフォルトは OFF です。セキュア LDAP プロトコルを使用するには、URL の先頭に `ldaps://` ではなく `ldap://` を使用します。セキュア LDAP を使用するときは、TLS オプションを OFF に設定する必要があります。

WITH ACTIVATE 句

この句を使用して、LDAP サーバをアクティブにしてすぐに使用できるようにします。この句によって、LDAP ユーザ認証の定義とアクティブ化を 1 つの文で行えるようにして、新しい LDAP サーバの状態を READY に変更します。

備考

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE ANY LDAP SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例は、検索パラメータ、認証 URL、3 秒のタイムアウトを設定し、LDAP サーバをアクティブにしてユーザの認証を開始できるようにします。TLS プロトコルまたは SECURE LDAP プロトコルを使用しない LDAP サーバへの接続が確立されます。次の例で login_mode オプションを設定するには、CREATE LDAP SERVER 文の実行に必要な権限に加えて、SET ANY SECURITY システム権限も必要です。

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA';
CREATE LDAP SERVER apps_primary
    SEARCH DN
        URL  'ldap://voyager:389/dc=MyCompany,dc=com??sub?cn=*'
        ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
        IDENTIFIED BY 'Secret99Password'
    AUTHENTICATION URL 'ldap://voyager:389/'
    CONNECTION TIMEOUT 3000
    WITH ACTIVATE;
```

この例は同じ検索パラメータを使用しますが、`ldaps://` を指定することにより、ホスト voyager、ポート 636 の LDAP サーバを使用してセキュア LDAP 接続が確立されるようにします。このポートにはセキュア LDAP プロトコルを使用している LDAP クライアントだけが接続できます。`'ldaps://voyager:636'` で LDAP サーバによって使用される証明書に署名する認証局 (CA) の証明書が含まれているファイル名を使用して、データベースセキュリティオプション Trusted_certificate_file が設定されている必要があります。LDAP サーバとのハンドシェイクの間、LDAP サーバから提示された証明書がデータベースサーバによって検証され、ファイル内にリストされているいずれかの証明書によって署名されているかどうか確認されます。LDAP サーバに対して指定された ACCESS ACCOUNT パラメータと IDENTIFIED BY パラメータも、LDAP サーバによって検証されます。

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA';
SET OPTION PUBLIC.trusted_certificates_file = '/opt/sap/shared/trusted.txt';
CREATE LDAP SERVER secure_primary
    SEARCH DN
        URL  'ldaps://voyager:636/dc=MyCompany,dc=com??sub?cn=*'
        ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
        IDENTIFIED BY 'Secret99Password'
    AUTHENTICATION URL 'ldaps://voyager:636/'
    CONNECTION TIMEOUT 3000
    WITH ACTIVATE;
```

関連情報

ALTER LDAP SERVER 文 [654 ページ]
DROP LDAP SERVER 文 [1042 ページ]
VALIDATE LDAP SERVER 文 [1398 ページ]

1.4.4.70 CREATE LOCAL TEMPORARY TABLE 文

プロシージャが完了した後も、明示的に削除されるか接続が終了するまで保持されるローカルテンポラリテーブルをプロシージャ内で作成します。

構文

```
CREATE LOCAL TEMPORARY TABLE [ IF NOT EXISTS ] [owner.]table-name
( { column-definition [ column-constraint ... ] | table-constraint |
pctfree }, ... )
[ ON COMMIT { DELETE | PRESERVE } ROWS | NOT TRANSACTIONAL ]
```

```
pctfree : PCTFREE percent-free-space
```

```
percent-free-space : integer
```

パラメータ

IF NOT EXISTS 句

指定された名前のテーブルがすでに存在する場合、変更は行われず、エラーも返されません。

ON COMMIT 句

デフォルトでは、テンポラリテーブルのローは COMMIT のときに削除されます。ON COMMIT 句を使用すると、COMMIT のときにローを保護できます。

NOT TRANSACTIONAL 句

状況によっては、NOT TRANSACTIONAL 句を使用するとパフォーマンスが向上します。これは、トランザクション単位でないテンポラリテーブルでの操作では、ロールバックログにエントリが作成されないためです。たとえば、テンポラリテーブルを使用するプロシージャが COMMIT や ROLLBACK の介入を受けずに繰り返し呼び出される場合、NOT TRANSACTIONAL が有用です。

備考

プロシージャの完了後も保持されるテーブルを作成する場合、DECLARE LOCAL TEMPORARY TABLE 文ではなく CREATE LOCAL TEMPORARY TABLE 文をプロシージャに使用します。CREATE LOCAL TEMPORARY TABLE 文を使用して作成されたローカルテンポラリテーブルは、明示的に削除するか接続が終了するまで保持されます。

CREATE LOCAL TEMPORARY TABLE を使用して作成されたテーブルは、システムカタログの SYSTABLE ビューには表示されません。

CREATE LOCAL TEMPORARY TABLE を使用して IF 文で作成されたローカルテンポラリテーブルは、IF 文が完了した後も保持されます。

同じスコープ内にある 2 つのローカルテンポラリテーブルは、同じ所有者や名前にはできません。ベーステーブルと同じ所有者と名前のローカルテンポラリテーブルを作成すると、ローカルテンポラリテーブルのスコープが終了した時点で、そのベース

テーブルは、その接続内でのみ参照できるようになります。接続では、既存のテンポラリテーブルと同じ所有者と名前のベーステーブルを作成できません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

CREATE LOCAL TEMPORARY TABLE は、オプションの ANSI/ISO SQL 言語機能 F531 の一部です。PCTFREE 句と NOT TRANSACTIONAL 句は、標準にはありません。文に定義されたカラム定義と制約定義には、標準にはない構文拡張を含めることもできます。ANSI/ISO SQL 標準では、CREATE LOCAL TEMPORARY TABLE 文によって作成されるテーブルをシステムカタログに表示することが規定されていますが、ソフトウェアでは表示されません。

Transact-SQL

CREATE LOCAL TEMPORARY TABLE は、Adaptive Server Enterprise ではサポートされていません。SAP Adaptive Server Enterprise では、特殊文字 # で始まるテーブル名を使用して CREATE TABLE 文でテンポラリテーブルを作成します。

例

次の例は、TempTab というローカルテンポラリテーブルを作成します。

```
CREATE LOCAL TEMPORARY TABLE TempTab ( number INT )
ON COMMIT PRESERVE ROWS;
```

関連情報

CREATE TABLE 文 [952 ページ]
DECLARE LOCAL TEMPORARY TABLE 文 [1007 ページ]

1.4.4.71 CREATE LOGIN POLICY 文

ログインポリシーを作成します。

構文

```
CREATE LOGIN POLICY policy-name policy-options
```

```
policy options :
policy-option [ policy-option ... ]
```

```
policy-option :
policy-option-name = policy-option-value
```

```
policy-option-value :
{ UNLIMITED | option-value }
```

パラメータ

policy-name

ログインポリシーの名前。

policy-option-name

ログインポリシーオプションの名前。

policy-option-value

ログインポリシーオプションに割り当てられている値。UNLIMITED を指定すると、制限は適用されません。

備考

ポリシーオプションを指定していない場合、対応するルートログインポリシーオプションが常に使用されます。ただし、新しいポリシーは、ルートポリシーのみのオプションである max_non_dba_connections および root_auto_unlock_time ポリシーオプションを継承しません。

すべての未指定の設定については、新しいポリシーはルートログインポリシーから静的コピーを作成しません。未指定の設定には常にルートログインポリシーのデフォルトが戻されます。つまり、ルートログインポリシーオプションへの変更は、オプションが指定されていないすべてのポリシーにも影響します。

すべての新しいデータベースには、ルートログインポリシーが含まれています。ルートログインポリシーの値を変更することはできますが、ポリシーは削除できません。ルートログインポリシーのデフォルト値の概要はパラメータの項に示されています。

権限

MANAGE ANY LOGIN POLICY システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Test1 ログインポリシーを作成します。この例では、パスワードは無期限で、アカウントがロックされるまでに許容されるユーザパスワードの入力回数が最大 5 回に設定されています。

```
CREATE LOGIN POLICY Test1
PASSWORD_LIFE_TIME=UNLIMITED
MAX_FAILED_LOGIN_ATTEMPTS=5;
```

次の例は、LDAP ユーザ認証を使用する新しいログインポリシー (ldap_user_policy) の代表的な設定を示します。プライマリとセカンダリの両方のサーバ設定オブジェクト (事前に作成されているもの) が指定されてセカンダリ LDAP サーバへのフェイルオーバーが可能になり、システムリソース、ネットワークリソース、またはプライマリとセカンダリの両方の LDAP サーバが反応しなくなったときに、標準認証にフェイルオーバーできる機能が有効になります。この例は、LDAP サーバが受信要求に対応できない場合に、キャッシュ済みの値で応答することを許可する認証オプションの組み合わせを提供します。この例は、login_mode データベースオプションに 'Standard' が含まれていることを前提にしています。この例で取り上げているプライマリサーバとセカンダリサーバは架空のものであるため、この例を使用して実行することはできません。

```
CREATE LOGIN POLICY ldap_user_policy
LDAP_PRIMARY_SERVER=ldapsrv1
LDAP_SECONDARY_SERVER=ldapsrv2
LDAP_FAILOVER_TO_STD=ON;
```

関連情報

ALTER LOGIN POLICY 文 [657 ページ]
ALTER USER 文 [731 ページ]
COMMENT 文 [769 ページ]
CREATE USER 文 [990 ページ]

DROP LOGIN POLICY 文 [1043 ページ]
DROP USER 文 [1084 ページ]

1.4.4.72 CREATE MATERIALIZED VIEW 文

マテリアライズドビューを作成します。

構文

```
CREATE MATERIALIZED VIEW
[ owner [,....]materialized-view-name [ ( alt-column-names, ... ) ]
[ IN dbspace-name ]
AS select-statement
[ CHECK { IMMEDIATE | MANUAL } REFRESH ]
```

```
alt-column-names :
( column-name)
```

パラメータ

alt-column-names 句

この句は、マテリアライズドビューでカラムの代替名を指定するときに使用します。代替カラム名を指定する場合、alt-column-names に含まれるカラム数が select-statement のカラム数に一致する必要があります。代替カラム名を指定しない場合、select-statement の名前が設定されます。

IN 句

この句は、マテリアライズドビューを作成する DB 領域を指定するときに使用します。この句が指定されない場合、default_dbspace オプションで指定された DB 領域にマテリアライズドビューが作成されます。指定されている場合は、システム DB 領域が使用されます。

AS 句

この句は、マテリアライズドビューの移植に使用するデータを SELECT 文の形式で指定するときに使用します。マテリアライズドビュー定義は、ベーステーブルのみを参照できます。ビュー、他のマテリアライズドビュー、またはテンポラリテーブルは参照できません。select-statement にはカラム名を含めるか、エイリアス名を指定します。alt-column-names を指定する場合、これらの名前が select-statement に指定されたエイリアスの代わりに使用されます。

SELECT 文のカラム名は明示的に指定される必要があります。SELECT * の構成は使用できません。たとえば、CREATE MATERIALIZED VIEW matview AS SELECT * FROM table-name とは指定できません。また、select-statement ではオブジェクト名を完全に修飾してください。

CHECK 句

この句は、実際にビューを作成しないで文を検証するときに使用します。CHECK 句を指定すると、次のようになります。

- データベースサーバは、CREATE MATERIALIZED VIEW がこの句なしで実行された場合に行う通常の言語チェックを実行し、通常どおり生成されたエラーが返されます。

- データベースサーバは実際のビューの作成を行わないため、作成時に発生する特定のエラーは生成されません。たとえば、指定されたビュー名がすでに存在することを示すエラーは生成されません。このため、CHECK 句を使用すると、ビューの命名で競合することなく、ビュー定義の変更をテストできます。
- CHECK IMMEDIATE REFRESH が使用される場合、データベースサーバは即時ビューで構文が有効であるかを検証し、エラーを返します。
- データベースへの変更は行われず、トランザクションログには何も記録されません。
- 文実行の開始時に暗黙的なコミットがあり、実行中に取得されたすべてのロックを解放するロールバックが最後に行われます。

備考

マテリアライズドビューを作成すると、これは手動ビューとなり、初期化されていない状態です。つまり、再表示タイプは手動であり、ビューは再表示されていません (データは移植されています)。このビューを初期化するには、REFRESH MATERIALIZED VIEW 文を実行するか、sa_refresh_materialized_views システムプロシージャを使用します。

マテリアライズドビューの暗号化、PCTFREE 設定の変更、再表示タイプの変更、さらにオプティマイザによる使用の有無を設定できます。ただし、これらの設定変更は、マテリアライズドビューを作成した後、ALTER MATERIALIZED VIEW 文を使用して実施します。マテリアライズドビュー作成時のデフォルト値は、次のとおりです。

- NOT ENCRYPTED
- ENABLE USE IN OPTIMIZATION
- PCTFREE は、データベースのページサイズに応じて、ページサイズが 4 KB の場合は 200 バイト、ページサイズが 2 KB の場合は 100 バイトに設定される
- MANUAL REFRESH

マテリアライズドビューを作成するには、いくつかのデータベースとサーバのオプションが有効になっている必要があります。

sa_recompile_views システムプロシージャは、マテリアライズドビューに影響しません。

権限

ユーザ本人が所有するマテリアライズドビューを作成するには、CREATE MATERIALIZED VIEW システム権限が必要です。また、マテリアライズドビューによって参照される基本オブジェクトを所有しているか、そのオブジェクトに対する SELECT 権限を持っている必要があります。

他のユーザが所有するマテリアライズドビューを作成するには、CREATE ANY MATERIALIZED VIEW または CREATE ANY OBJECT システム権限が必要です。

関連する動作

オートコミット。実行中に、CREATE MATERIALIZED VIEW 文は、マテリアライズドビューから参照されるすべてのテーブルに排他ロックをかけますが、ブロックは実行しません。参照テーブルのいずれかがロックできない場合、文は失敗し、エラーが返されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、SQL Anywhere サンプルデータベースに格納されている従業員に関する機密情報を含むマテリアライズドビューを作成します。この後、ビューを初期化して使用するには、例に示すように REFRESH MATERIALIZED VIEW 文を実行する必要があります。

```
CREATE MATERIALIZED VIEW EmployeeConfid2 AS
SELECT EmployeeID, Employees.DepartmentID,
    SocialSecurityNumber, Salary, ManagerID,
    Departments.DepartmentName, Departments.DepartmentHeadID
FROM GROUPO.Employees, GROUPO.Departments
WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid2;
```

関連情報

ALTER MATERIALIZED VIEW 文 [659 ページ]
DROP MATERIALIZED VIEW 文 [1045 ページ]
REFRESH MATERIALIZED VIEW 文 [1252 ページ]
CREATE VIEW 文 [995 ページ]
sa_refresh_materialized_views システムプロシージャ [1591 ページ]

1.4.4.73 CREATE MESSAGE 文 [T-SQL]

メッセージ番号/メッセージ文字列のペアを作成します。このメッセージ番号は PRINT 文と RAISERROR 文で使用できます。

構文

```
CREATE MESSAGE message-number AS message-text
```

```
message-number : integer
```

```
message-text : string
```

パラメータ

message-number

追加するメッセージのメッセージ番号。ユーザ定義メッセージのメッセージ番号は 20000 以上にしてください。

message-text

追加するメッセージのテキスト。最大長は、255 バイトです。PRINT と RAISERROR は、メッセージテキストのプレースホルダを認識します。1 つのメッセージには、ユニークなプレースホルダを任意の順序で 20 個まで含めることができます。これらのプレースホルダは、メッセージのテキストをクライアントに送信するときにメッセージの後に付けた任意の引数のフォーマット文字列に置換されます。

メッセージを文法的構文が異なる言語に翻訳するときに順番を変えられるように、プレースホルダには番号が付けられます。引数のプレースホルダの形式は "%nn!" です。パーセント記号 (%)、nn (1 ～ 20 の整数)、感嘆符 (!) の順になります。整数は引数の引数リスト内の位置を表します。"%1!" が最初の引数で、2 番目の引数は "%2!"、のようになります。

sp_addmessage の `language` 引数に対応するパラメータはありません。

備考

PRINT 文と RAISERROR 文によって使用される、ユーザ定義メッセージをシステムテーブル ISYSUSERMESSAGE に追加します。

権限

CREATE MESSAGE または CREATE ANY OBJECT システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

CREATE MESSAGE には、Adaptive Server Enterprise の sp_addmessage システムプロシージャで提供される機能が備えられています。

例

次の例は新しいメッセージを作成します。

```
CREATE MESSAGE 20000 AS 'End of line reached';
```

関連情報

PRINT 文 [T-SQL] [1243 ページ]
RAISERROR 文 [1246 ページ]
DROP MESSAGE 文 [1046 ページ]
SYSUSERMESSAGE システムビュー [1858 ページ]

1.4.4.74 CREATE MIRROR SERVER 文

データベースミラーリングまたは読み込み専用のスケールアウトに使用されているミラーサーバを作成するか、置き換えます。

構文

ミラーサーバの作成

```
CREATE [ OR REPLACE ] MIRROR SERVER mirror-server-name
AS { PRIMARY | MIRROR | ARBITER | PARTNER }
[ server-option = string [ ... ] ]
```

ミラーサーバをコピーとして作成

```
CREATE [ OR REPLACE ] MIRROR SERVER mirror-server-name
AS COPY
  { FROM SERVER parent-name [ OR SERVER server-name ] | USING AUTO PARENT }
[ server-option = string [ ... ] ]
```

```
server-option :
connection_string
logfile
preferred
state_file
```

```
parent-name :
server-name | PRIMARY
```

パラメータ

OR REPLACE 句

CREATE MIRROR SERVER は、ミラーサーバを作成します。指定された名前のミラーサーバがデータベースにすでに存在する場合は、エラーが返されます。

OR REPLACE を指定すると、ミラーサーバがデータベースに存在しない場合は作成され、存在する場合は置き換えられます。

AS 句

次のいずれかのサーバタイプを指定できます。

PRIMARY

PRIMARY タイプのミラーサーバは、実際のデータベースサーバではなく、仮想サーバまたは論理サーバを定義します。このサーバの名前は、データベースの代替サーバ名です。アプリケーションで代替サーバ名を使用して、現在プライマリサーバとして動作しているサーバに接続できます。PRIMARY としてマーク付けされているサーバの接続文字列

- コピーノードによってルートノードまたは PRIMARY の親に接続するために使用される接続文字列を定義します。
- 接続パラメータ NodeType の PRIMARY 値で使用される接続文字列を定義します。

データベースで使用できる PRIMARY サーバは 1 つのみです。

MIRROR

MIRROR タイプのミラーサーバは、実際のデータベースサーバではなく、仮想サーバまたは論理サーバを定義します。このサーバの名前は、データベースの代替ミラーサーバ名です。アプリケーションで代替ミラーサーバ名を使用して、現在読み込み専用ミラーとして動作しているサーバに接続できます。MIRROR としてマーク付けされたサーバも、NodeType 接続パラメータの MIRROR 値で使用される接続文字列を定義します。データベースで使用できる MIRROR サーバは 1 つのみです。

ARBITER

データベースミラーリングシステムでは、監視サーバは、いずれの PARTNER サーバがデータベースの所有権を取得するかを決定するのに役立ちます。監視サーバは、パートナサーバが監視サーバへの接続に使用できる接続文字列を使用して定義します。データベースで使用できる ARBITER サーバは 1 つのみです。

PARTNER

ミラーサーバ名は、-n サーバオプションで指定されたデータベースサーバ名に対応している必要があり、connection_string ミラーサーバオプションで指定された SERVER 接続文字列と一致している必要があります。

データベースミラーリングシステムでは、パートナは接続文字列の値を使用して相互に接続します。読み込み専用スケールアウトシステムでは、その親として現在のサーバを備えたコピーノードによって接続文字列が使用されます。

ミラーリングまたは読み込み専用スケールアウトシステムでのミラーリング

データベースミラーリングを行うには、2 つの PARTNER サーバを定義する必要があり、両方のサーバに接続文字列とステータスファイルが必要です。

データベースミラーリングシステムでは、PARTNER として定義されたサーバは、プライマリサーバとなってデータベースの所有権を取得できます。

ミラーリングのない読み込み専用スケールアウト

読み込み専用スケールアウトに対しては 1 つの PARTNER サーバを定義する必要があり、それが接続文字列を持っていて、ステータスファイルを持っていないことが必要です。このサーバはルートサーバであり、読み込みと書き込みの両方の処理が許可されたデータベースの唯一のコピーを実行します。

COPY

読み込み専用のスケールアウトシステムでは、この値は、データベースサーバがコピーノードであることを指定します。このサーバ上のデータベース接続はすべて、読み込み専用です。ミラーサーバ名は、-n サーバオプションで指定されたデータベースサーバ名に対応している必要があり、connection_string ミラーサーバオプションで指定された SERVER 接続文字列と一致している必要があります。

AS COPY が指定されている場合は、FROM SERVER 句または USING AUTO PARENT 句も指定する必要があります。

接続文字列は NodeType 接続パラメータの COPY 値で使用され、親としてこのサーバを持つ他のコピーノードでも使用されます。

読み込み専用スケールアウトシステムにコピーノードを追加する場合は、そのコピーノードに対して CREATE MIRROR SERVER 文を実行するか、ルートサーバにミラーサーバを自動的に定義させることができます。

FROM SERVER 句

この句を使用できるのは、AS COPY が指定されている場合だけです。この句はスケールアウトシステムのサーバのツリーを構成し、コピーノードがどのサーバからトランザクションログページを取得するのかを示します。

ミラーサーバ名または PRIMARY を使用して、親を指定できます。コピーノードの代替の親は、OR SERVER 句を使用して指定できます。

2 レベル (パートナノードとコピーノード) のみが存在するデータベースミラーリングシステムでは、コピーノードが、現在のプライマリサーバまたはミラーサーバからトランザクションログページを取得します。

コピーノードは、データベースに格納されているコピーノードのミラーサーバ定義を使用して、接続先となるサーバを判断します。コピーノードの定義から親の定義を探すことができ、親の定義から親に接続するための接続文字列を取得できます。

スケールアウトシステムにコピーノードを明示的に定義する必要はありません。接続時に、ルートノードによってコピーノードを定義できます。

OR SERVER 句

OR SERVER 句を使用して、コピーノードの代替の親を指定します。

USING AUTO PARENT 句

この句を使用できるのは、AS COPY が指定されている場合だけです。この句を指定すると、このサーバの親がプライマリサーバによって割り当てられます。この句を使用して既存のコピーノードサーバを置き換えても、そのコピーノードの親および代替の親に関する定義は変更されません。

server-option 句

`server-option` の変数名を指定できます。

次のオプションがサポートされています。

connection_string サーバオプション

サーバに接続するための接続文字列を指定します。1 つのミラーサーバを別のミラーサーバに接続するときにユーザ ID とパスワードは使用されないため、ミラーサーバの接続文字列にはユーザ ID とパスワードを含めないでください。

logfile サーバオプション

データベースミラーリングが使用されている場合に、ミラーサーバ間で送信されるファイル (1 行に 1 つの要求を含む) のロケーションを指定します。このファイルは、デバッグ専用です。

preferred サーバオプション

サーバがミラーリングシステムにおいて優先サーバであるかどうかを指定します。YES または NO を指定できます。優先サーバは、可能なかぎり、プライマリサーバのロールを引き受けます。このオプションは、PARTNER サーバを定義する場合に指定します。

state_file サーバオプション

ミラーリングシステムに関するステータス情報の管理に使用されるファイルのロケーションを指定します。データベースミラーリングには、このオプションは必須です。ミラーリングシステムでは、PARTNER タイプのサーバにはステータスファイルを指定してください。監視サーバの場合は、サーバの起動コマンドの一部としてロケーションが指定されます。

備考

この文は、ミラーサーバ定義を作成または置換するものであり、ミラーサーバ定義を変更するものではありません。ミラーサーバ定義を変更するには、ALTER MIRROR SERVER 文を使用します。

データベースのミラーリングシステムで使用されるミラーサーバタイプは PRIMARY、MIRROR、ARBITER、PARTNER のいずれかであり、3 つ以上のパートナサーバを作成することはできません。

読み込み専用のスケールアウトシステムでは、ミラーサーバのタイプに PRIMARY、PARTNER、または COPY を使用できます。

PARTNER または COPY タイプのサーバのミラーサーバ名は、ミラーリングシステムの一部であるデータベースサーバの名前 (-n サーバオプションで使用される名前) と一致している必要があります。この要件により、各データベースサーバで、サーバ自体の定義とその親の定義を見つけることができるようになります。また、すべてのコピーノードサーバの名前がユニークになるように指定します。前述の `mirror-server-name`、`parent-name`、および `server-name` は、7 ビット ASCII 文字でなければなりません。

データベースミラーリングシステムで、コピーしているデータベースの監視サーバとしてコピーノードを使用するには、高可用性システムのデータベースサーバのいずれのサーバ名とも一致しない名前で監視サーバを作成します。このように設定すると、ミラーサーバ定義において、監視サーバの名前が監視サーバの接続文字列が保持するプレースホルダとして使用されます。

注記

変数名を受け付けるパラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

読み込み専用のスケールアウトとデータベースミラーリングには、それぞれ別途ライセンスが必要です。

権限

MANAGE ANY MIRROR SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、データベースミラーリングシステムでプライマリサーバとして使用できるミラーサーバを作成します。

```
CREATE MIRROR SERVER "scaleout_primary"
 AS PRIMARY
connection_string = 'server=scaleout_primary;host=winxp-2:6871,winxp-3:6872';
```

次の文は、データベースミラーリングシステムでミラーサーバとして使用できるミラーサーバを作成します。

```
CREATE MIRROR SERVER "scaleout_mirror"
AS MIRROR
connection_string = 'server=scaleout_mirror;host=winxp-2:6871,winxp-3:6872';
```

次の文は、データベースミラーリングシステムで監視サーバとして使用できるミラーサーバを作成します。

```
CREATE MIRROR SERVER "scaleout_arbiter"
AS ARBITER
connection_string = 'server=scaleout_arbiter;host=winxp-4:6870';
```

次の文は、データベースミラーリングシステムでパートナサーバとして使用できる 2 つのミラーサーバを作成します。

```
CREATE MIRROR SERVER "scaleout_server1"
AS PARTNER
connection_string = 'server=scaleout_server1;HOST=winxp-2:6871'
state_file = 'c:¥¥server1¥¥server1.state';

CREATE MIRROR SERVER "scaleout_server2"
AS PARTNER
connection_string = 'server=scaleout_server2;HOST=winxp-3:6872'
state_file = 'c:¥¥server2¥¥server2.state';
```

次の文は、データベースミラーリングシステムで監視サーバとして稼働できるコピーノードを作成します。

```
CREATE MIRROR SERVER "scaleout_child"
AS COPY FROM SERVER  "scaleout_primary"
```

```
connection_string = 'server=scaleout_child;host=winxp-5:6878';
```

次の文は、異なるデータベースミラーリングシステムの監視サーバとしてコピーノードを定義します。

```
CREATE MIRROR SERVER "The Arbiter"
AS ARBITER
connection_string = 'server=scaleout_child;host=winxp-5:6878';
```

次の文は、server-name がすでに存在する場合に現在の親を保持します。ただし、新しい親は自動生成しません。

```
CREATE OR REPLACE MIRROR SERVER "server-name" AS COPY USING AUTO PARENT;
```

例

次の例は、connection_string パラメータの変数を使用してミラーサーバを作成します。

次の文は、接続文字列の変数を作成します。

```
CREATE VARIABLE @connstr_value LONG VARCHAR ;
SET @connstr_value = ' server=new_scaleout_primary;host=winxp-2:6871,winxp-3:6872
' ;
```

次の文は、connection_string パラメータの変数 @connstr_value を使用してミラーサーバを作成します。

```
CREATE MIRROR SERVER new_scaleout_primary AS PRIMARY
connection_string = @connstr_value ;
```

関連情報

SYSMIRRORSERVER システムビュー [1818 ページ]
SET MIRROR OPTION 文 [1305 ページ]
ALTER MIRROR SERVER 文 [662 ページ]
COMMENT 文 [769 ページ]
DROP MIRROR SERVER 文 [1047 ページ]

1.4.4.75 CREATE MUTEX 文

ファイルやプロシージャなどのリソースをロックするために使用できるミューテックス (ロック) を作成または置換します。

構文

```
CREATE [ OR REPLACE | TEMPORARY ] MUTEX [ IF NOT EXISTS ] [ owner.]mutex-name
[ SCOPE { CONNECTION | TRANSACTION } ]
```

パラメータ

owner

ミューテックスの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

mutex-name

ミューテックスの名前。CHAR データベース照合に有効な識別子を指定します。mutex-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

OR REPLACE clause

この句は、同じ名前の永続ミューテックスの定義 (存在する場合) を上書き (更新) するときに使用します。

OR REPLACE 句を指定し、その時点でこの名前のミューテックスが使用中の場合、文からエラーが返されます。

この句は、TEMPORARY 句または IF NOT EXISTS 句と一緒に使用しないでください。

TEMPORARY clause

この句は、一時ミューテックスを作成するときに使用します。

この句は、OR REPLACE 句と一緒に使用しないでください。

IF NOT EXISTS clause

この句は、存在しないミューテックスを作成するときに使用します。同じ名前のミューテックスが存在する場合は、何も実行されずエラーも返されません。

この句は、OR REPLACE 句と一緒に使用しないでください。

SCOPE clause

この句は、ミューテックスをトランザクション (TRANSACTION) に適用するか接続 (CONNECTION) に適用するかを指定するときに使用されます。SCOPE 句を指定しない場合、デフォルトの動作は CONNECTION です。

備考

永続および一時のミューテックスおよびセマフォは、同じネームスペースを共有するため、これらのオブジェクトは同じ名前および所有者で作成することはできません。OR REPLACE および IF NOT EXISTS 句を使用すると、命名に関する不適切なエラーが発生することがあります。たとえば、永続ミューテックスが存在していて、それと同じ名前の一時セマフォを作成しようとすると、IF NOT EXISTS を指定してもエラーが返されます。同様に、一時セマフォが存在していて、OR REPLACE を指定して同じ名前の永続セマフォで置き換えようとすると、エラーが返されます。これは、同じ名前で 2 つめのオブジェクトを作成しようとするのと同じためです。

永続ミューテックスの定義は、データベースの再起動後も保持されます。ただし、ステータス情報 (ロック済または解除済) は保持されません。

一時ミューテックスは、ミューテックスを作成した接続が終了するまで、または DROP MUTEX 文を使用してミューテックスが削除されるまで存続します。別の接続が一時ミューテックスを待機中で、一時ミューテックスを作成した接続が終了した場合は、待機中の接続にミューテックスが削除されたことを示すエラーが返されます。

CONNECTION スコープミューテックスは、接続が終了したときを除き自動的に解除されません。

権限

CREATE ANY MUTEX SEMAPHORE または CREATE ANY OBJECT のシステム権限が必要です。

関連する動作

オートコミット（永続ミューテックスの場合のみ）。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ストアドプロシージャのクリティカルセクションを保護する protect_my_cr_section という接続スコープミューテックスを作成します。

```
CREATE MUTEX protect_my_cr_section SCOPE CONNECTION;
```

関連情報

DROP MUTEX 文 [1048 ページ]
LOCK MUTEX 文 [1199 ページ]
RELEASE MUTEX 文 [1259 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.76 CREATE ODATA PRODUCER 文

OData プロデューサを作成します。

構文

```
CREATE [ OR REPLACE ] ODATA PRODUCER name [ producer-clause... ]
```

```
producer-clause:
[ ADMIN USER { NULL | user } ]
[ AUTHENTICATION [ DATABASE | USER user ] ]
```

```
[ [ NOT ] ENABLED ]
[ MODEL [ FROM ] { FILE string-or-variable [ ENCODING string ] | VALUE { NULL |
string-or-variable [ ENCODING string ] } } ]
  [ SERVICE ] ROOT string
[ USING string ]
```

パラメータ

ADMIN USER clause OData プロデューサがテーブルの作成に使用する管理ユーザ、プロデューサ、および繰返可能要求を管理するイベントを指定します。CREATE TABLE、CREATE PROCEDURE、VERIFY ODATA、および MANAGE ANY EVENT システム権限が必要です。このユーザは、AUTHENTICATION 句で指定したユーザと同じにすることはできません。

AUTHENTICATION clause

OData サーバのデータベースへの接続方法を指定します。デフォルト設定 DATABASE では、ユーザはパーソナル化クレデンシャルで接続できます。これらのクレデンシャルは、HTTP 基本認証を使用して要求されます。

指定したユーザのクレデンシャルを使用して接続するすべてのユーザに、USER を指定します。指定したユーザには、VERIFY ODATA システム権限が必要です。USER は、テストの場合や読み取り専用プロデューサに対して使用することを推奨します。詳細については、"OData サーバのセキュリティの考慮事項" を参照してください。

ENABLED clause OData プロデューサが有効か無効かを指定します。デフォルトでは、OData プロデューサは有効です。

MODEL clause

どのテーブルとビューが OData メタデータに公開されるかを示す OData プロデューサのサービスモデルを (ODSL に) 指定します。FILE 句を指定する場合は、値は変数または ODSL データを含むファイル名の文字列にする必要があります。ファイル名が完全修飾パスでない場合、パスはデータベースサーバの現在の作業ディレクトリからの相対パスです。ファイル名が指定される場合、ファイルが読み込まれ、ファイルのコンテンツがデータベースに格納されてトランザクションログに記録されます。デフォルトでは、テーブルとビューはユーザ権限に基づいて公開されます。プライマリキーのないテーブルとビューは公開されません。

ENCODING 句を使用してモデルデータまたはファイルの文字セットを指定します。デフォルトは UTF-8 です。

SERVICE ROOT clause

OData サーバで OData サービスのルートを指定します。この OData プロデューサのすべてのリソースには、次のフォーマットの URI を使用してアクセスします。`scheme:host:port/path-prefix/resource-path[query-options]`。

SERVICE ROOT 句は `/path-prefix` です。値の指定は必須で、NULL にすることはできません。また、`/` から開始する必要があります。

USING clause

追加の OData プロデューサオプションを指定します。オプションは、name=value のペアをセミコロンで区切ったリストで指定します。

オプション	説明
ConnectionPoolMaximum =`num-max-connections`	この OData プロデューサが接続プールで使用するために開き続ける同時接続の最大数を示します。 少ない接続は、サーバの負荷に応じて、接続プールによって使用されます。 デフォルトでは、接続プールのサイズは、データベースサーバでサポートされる同時接続の最大数の半分に制限されています。
CSRFTokenTimeout =`num-seconds-valid`	CSRF トークンチェックを有効にし、トークンを有効にする秒数を指定します。 デフォルトでは、この値は 0 です。つまり、CSRF トークンチェックは無効です。それ以外の場合は、秒数は 1 ～ 1800 の有効な整数である必要があります。
PageSize =`num-max-entities`	次のリンクを発行する前に、取得エンティティセット応答に含めるエンティティの最大数を指定します。 デフォルト設定は 100 です。
ReadOnly = { *true* | *false* }	変更要求を無視するかどうかを示します。 デフォルト設定は *false* です。
RepeatRequestForDays = { `days-number` }	繰返可能要求を有効にする長さを指定します。 この値は、1 ～ 31 の範囲内の整数である必要があります。 デフォルト設定は 2 です。 このオプションは、ADMIN USER 句が NULL ではない場合のみ有効です。
SecureOnly = { *true* | *false* }	プロデューサが HTTPS ポートでのみ要求のみを受信するかどうかを示します。 デフォルト設定は *false* です。
ServiceOperationColumnNames = { *generate* | *database* }	ReturnType で使用される ComplexType のプロパティ名を指定するときに、メタデータのカラム名を生成するか、データベースの結果セットから取得するかを指定します。 デフォルト設定は *generate* です。

備考

OData プロデューサを作成し、開始します。OData プロデューサが開始するのは、プロデューサが有効化されていて、かつサーバ上で -xs ODATA データベースサーバオプションを使用してデータベースが開始された場合のみとなります。

データベースが -xs ODATA データベースサーバオプションを使用して開始されたデータベースサーバ上で開始された場合、すべての有効な OData プロデューサがそのデータベースサーバ上で開始されます。

権限

MANAGE ODATA システム権限が必要です。

AUTHENTICATION USER 句または ADMIN USER 句を指定する場合は、指定したユーザには VERIFY ODATA システム権限が必要です。

例

次の例では、OData プロデューサの作成方法を示します。

```
CREATE ODATA PRODUCER OrderEntryProducer
    ADMIN USER SuperODataUser
    AUTHENTICATION USER ODataUser
    ENABLED
    MODEL FILE '../../ordermodel.osdl'
    SERVICE ROOT '/orders/'
    USING 'ConnectionPoolMaximum=10;CSRFTokenTimeout=600;PageSize=100;
      ReadOnly=false;RepeatRequestForDays=3;SecureOnly=true;
      ServiceOperationColumnNames=generate';
```

関連情報

ALTER ODATA PRODUCER 文 [666 ページ]
DROP ODATA PRODUCER 文 [1050 ページ]
COMMENT 文 [769 ページ]
SYSODATAPRODUCER システムビュー [1822 ページ]

1.4.4.77 CREATE PROCEDURE 文 [外部呼び出し]

ネイティブプロシージャまたは外部プロシージャへのインタフェースを作成します。

構文

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure-name
     ( [ parameter[, ... ] ] )
[ RESULT ( result-column [, ... ] )
     | NO RESULT SET
     | DYNAMIC RESULT SETS integer-expression ]
[ SQL SECURITY { INVOKER | DEFINER } ]
{ EXTERNAL NAME 'native-call'
   | EXTERNAL NAME 'c-call' LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32 |
C_ODBC64 }
   | EXTERNAL NAME 'clr-call' LANGUAGE CLR
   | EXTERNAL NAME 'perl-call' LANGUAGE PERL
   | EXTERNAL NAME 'php-call' LANGUAGE PHP
   | EXTERNAL NAME 'java-call' LANGUAGE JAVA
   | EXTERNAL NAME 'js-call' LANGUAGE JS }
```

```
parameter :
[ parameter-mode ] parameter-name data-type [ DEFAULT expression ]
| SQLCODE
| SQLSTATE
```

```
parameter-mode :
IN
| OUT
| INOUT
```

```
result-column :
column-name data-type
```

```
native-call :
[ system-configuration:]function-name@library-file-prefix[.{ so | dll} ]
```

```
system-configuration :
{ generic-operating-system | specific-operating-system } [ (processor-
architecture) ]
```

```
generic-operating-system :
{ Unix | Windows }
```

```
specific-operating-system :
{ AIX | HPUX | Linux | OSX | Solaris | WindowsNT }
```

```
processor-architecture :
{ 32 | 64 | ARM | IA64 | PPC | SPARC | X86 | X86_64 }
```

```
c-call :
[ operating-system:]function-name@library; ...
```

```
operating-system :
Unix
```

```
clr-call :
dll-name::function-name( param-type-1[, ... ] )
```

```
perl-call :
<file=perl-file> $sa_perl_return = perl-subroutine( $sa_perl_arg0[, ... ] )
```

```
php-call :
<file=php-file> print php-func( $argv[1][, ... ] )
```

```
java-call :
[package-name.]class-name.method-name java-method-signature
```

```
java-method-signature :
( [ java-field-descriptor, ... ] ) java-return-descriptor
```

```
java-field-descriptor および java-return-descriptor :
{ Z
  | B
  | S
```

```
  | I
  | J
  | F
  | D
  | C
  | V
  | [descriptor
  | Lclass-name;
}
```

```
js-call :
        <js-return-descriptor><file=js-object> js-func( js-field-descriptor[ ...])
```

```
js-field-descriptor および js-return-descriptor :
{ S
  | B
  | I
  | U
  | D
  | [descriptor
}
```

パラメータ

外部プロシージャまたはネイティブプロシージャを呼び出す永続的なストアドプロシージャの作成には、さまざまなプログラミング言語を使用できます。PROC は PROCEDURE の同義語として使用できます。

OR REPLACE 句

CREATE OR REPLACE PROCEDURE を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存の権限は保持されます。使用中のプロシージャを置き換えようとすると、エラーが返されます。

parameter

パラメータ名は、カラム名など他のデータベース識別子に対するルールに従って付けてください。これらは有効な SQL データ型にする必要があります。

パラメータには、IN、OUT、INOUT のいずれかのキーワードをプレフィクスとして付けることができます。これらの値のいずれも指定しない場合、パラメータはデフォルトで INOUT になります。キーワードには次の意味があります。

IN

このパラメータは、プロシージャに値を与える式です。

OUT

このパラメータは、プロシージャから値を受け取ることがある変数です。

INOUT

このパラメータはプロシージャに値を与え、プロシージャから新しい値を受け取ることがある変数です。

データ型を明示的に設定するか、%TYPE または %ROWTYPE 属性を指定し、データ型をデータベース内の別のオブジェクトのデータ型に設定できます。%TYPE を使用して、テーブルまたはビュー内のカラムのデータ型に設定します。%ROWTYPE を使用して、テーブルまたはビュー内のローから派生する複合データ型に設定します。

CALL 文を使ってプロシージャを実行する場合、必ずしもすべてのパラメータを指定する必要はありません。CREATE PROCEDURE 文の中にデフォルト値がある場合、不明のパラメータにデフォルト値を割り当てます。CALL に引数が指定されておらず、デフォルトも設定されていない場合には、エラーが発生します。

SQLSTATE と SQLCODE は、プロシージャが終了するときに、SQLSTATE または SQLCODE 値を出力する、特別な OUT パラメータです。SQLSTATE と SQLCODE の特別値は、プロシージャのリターンステータスのテストを目的として、プロシージャ呼び出しの直後にチェックできます。

SQLSTATE と SQLCODE 特別値は、その次の SQL 文によって修正されます。SQLSTATE と SQLCODE をプロシージャ引数として与えると、リターンコードは変数の中に格納されます。

OR REPLACE (CREATE OR REPLACE PROCEDURE) を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存の権限は保持されます。使用中のプロシージャを置き換えようとすると、エラーが返されます。

TEMPORARY 外部呼び出しプロシージャは作成できません。

RESULT 句

RESULT 句は結果セットのカラムの数と型を宣言します。RESULT キーワードに続く括弧で囲まれたリストは、結果カラムの名前と型を定義します。CALL 文が記述されていると、この情報を Embedded SQL DESCRIBE または ODBC SQLDescribeCol が返します。

RESULT 句を指定する場合は、文の最初の句として指定する必要があります。

Embedded SQL (LANGUAGE C_ESQL32、LANGUAGE C_ESQL64) 外部プロシージャまたは ODBC (LANGUAGE C_ODBC32、LANGUAGE C_ODBC64) 外部プロシージャは、結果セットを返さないか、1 つの結果セットを返すことができます。

Perl、PHP (LANGUAGE PERL、LANGUAGE PHP)、または JavaScript の外部プロシージャは、結果セットを返すことができません。データベースサーバによってロードされるネイティブ関数を呼び出すプロシージャも、結果セットを返すことができません。

CLR または Java (LANGUAGE CLR、LANGUAGE JAVA) 外部プロシージャは、結果セットを返さないか、1 つ以上の結果セットを返すことができます。

プロシージャは、その実行方法に応じて、それぞれカラム数が異なる複数の結果セットを生成します。たとえば次のプロシージャは、2 カラムを返す場合も、1 カラムを返す場合もあります。

```
CREATE PROCEDURE names( IN formal char(1))
BEGIN
   IF formal = 'n' THEN
      SELECT GivenName
      FROM GROUPO.Employees
   ELSE
      SELECT Surname, GivenName
      FROM Employees
   END IF
END;
```

これらの結果セットプロシージャは RESULT 句を指定しないで記述するか、Transact-SQL で記述します。これらの使用には、次の制約があります。

Embedded SQL

正しい形式の結果セットを取得するには、結果セットのカーソルが開かれてからローが返されるまでの間に、プロシージャコールを記述 (DESCRIBE) します。DESCRIBE 文の CURSOR `cursor-name` 句は必須です。

ODBC、OLE DB、ADO.NET

変数結果セットプロシージャは、これらのインタフェースを使用するアプリケーションで使用できます。結果セットの記述は、ドライバまたはプロバイダによって実行されます。

Open Client アプリケーション

変数結果セットプロシージャは Open Client アプリケーションで使用できます。

プロシージャが返す結果セットが 1 つだけの場合は、RESULT 句を使用します。この句を使用すると、カーソルがオープンした後で ODBC と Open Client のアプリケーションが結果セットを記述し直すのを防ぐことができます。

複数の結果セットを処理するために ODBC は、プロシージャが定義した結果セットではなく、現在実行中のカーソルを記述します。したがって、ODBC はいつもプロシージャ定義の RESULT 句内で定義されているカラム名を記述するわけではありません。この問題を回避するには、結果セットを生成する SELECT 文でカラムエイリアスを使用します。

NO RESULT SET 句

NO RESULT SET 句を指定する場合は、文の最初の句として指定する必要があります。

このプロシージャによって結果セットが返されないことを宣言します。この宣言によって、パフォーマンスが向上することがあります。

DYNAMIC RESULT SETS 句

DYNAMIC RESULT SETS 句を指定する場合は、文の最初の句として指定する必要があります。

この句は、LANGUAGE CLR 呼び出しまたは LANGUAGE JAVA 呼び出しで使用します。DYNAMIC RESULT SETS 句を使用して、プロシージャによって返される動的結果セットの数を指定します。RESULT 句が指定され、DYNAMIC RESULT SETS 句が指定されていない場合、動的結果セットの数は 1 とみなされます。RESULT 句も DYNAMIC RESULT SETS 句も指定されていない場合、結果セットは予測されず、結果セットが生成されるとエラーになります。

C_ESQL32、C_ESQL64、C_ODBC32、C_ODBC64 外部環境も結果セット (CLR や JAVA など) を返しますが、これらは 1 つの動的結果セットに制限されます。

Perl、PHP (LANGUAGE PERL、LANGUAGE PHP) または JavaScript の外部関数を呼び出すプロシージャは、結果セットを返すことができません。データベースサーバによってロードされるネイティブ関数を呼び出すプロシージャも、結果セットを返すことができません。

SQL SECURITY 句

SQL SECURITY 句は、INVOKER (プロシージャを呼び出すユーザ) または DEFINER (プロシージャを所有するユーザ) としてプロシージャが実行されるかどうかを定義します。デフォルトは DEFINER です。外部呼び出しの場合、この句は、外部環境での修飾されていないオブジェクト参照に対して所有者のコンテキストを確立します。

SQL SECURITY INVOKER が指定されている場合は、プロシージャを呼び出すユーザごとに注釈を行う必要があるためメモリ使用量が増えます。また、SQL SECURITY INVOKER が指定されている場合は、呼び出し側としても名前の決定が行われます。そのため、適切な所有者名で、すべてのオブジェクト名 (テーブル、プロシージャなど) を修飾します。たとえば、user1 が次のプロシージャを作成するとします。

```
CREATE PROCEDURE user1.myProcedure()
   RESULT( columnA INT )
   SQL SECURITY INVOKER
   BEGIN
     SELECT columnA FROM table1;
   END;
```

user2 がこのプロシージャを実行しようとし、テーブル user2.table1 が存在しない場合、テーブルルックアップエラーが生じます。さらに、user2.table1 が存在する場合は、意図する user1.table1 の代わりにこのテーブルが使用されます。このような状況を防ぐには、文においてテーブル参照を修飾します (単なる table1 ではなく、user1.table1 とします)。

EXTERNAL NAME native-call 句

native-call にはオペレーティングシステム、プロセッサ、ライブラリ、および関数の複数のセットを指定できるため、より精密に指定された設定が、それよりも精密さに欠けて定義された設定よりも優先されます。たとえば、Solaris(X86_64):myfunc64@mylib.so は Solaris:myfunc64@mylib.so よりも優先されます。

system-configuration がサポートされている構文で system-configuration を指定しないと、プロシージャがすべてのプラットフォームで稼働すると見なされます。UNIX は、AIX、HPUX、Linux、OS X、および Solaris の UNIX ベースオペレーティングシステムを意味します。一般的な用語である Windows は、Windows オペレーティングシステムのすべてのバージョンを意味します。

いずれかの呼び出しで UNIX を指定すると、他の呼び出しは Windows 用であると見なされます。

specific-operating-system および processor-architecture の値は、SQL Anywhere サーバでサポートされるオペレーティングシステムおよびプロセッサです。

ライブラリ名 (library-file-prefix) にはファイル拡張子が付きます。この拡張子は通常、Windows では .dll、UNIX では .so です。拡張子がない場合、ライブラリに対するプラットフォーム固有のデフォルトのファイル拡張子が追加されます。例:

```
CREATE PROCEDURE mystring( IN instr LONG VARCHAR )
EXTERNAL NAME 'mystring@mylib.dll;Unix:mystring@mylib.so';
```

EXTERNAL NAME 句を、プラットフォーム固有のデフォルト値を使用して簡単に記述すると、次のようになります。

```
CREATE PROCEDURE mystring( IN instr LONG VARCHAR )
EXTERNAL NAME 'mystring@mylib';
```

呼び出されると、関数を含むライブラリがデータベースサーバのアドレス領域にロードされます。ネイティブ関数は、データベースサーバの一部として実行されます。この場合、関数によって障害が引き起こされると、データベースサーバは停止します。このような動作のため、外部環境での関数のロードと実行には、LANGUAGE 属性を使用することをおすすめします。外部環境で関数によって障害が引き起こされた場合でも、データベースサーバは動作し続けます。

EXTERNAL NAME c-call 句

コンパイルされたネイティブ C 関数をデータベースサーバ内ではなく外部環境で呼び出すには、EXTERNAL NAME 句の後に LANGUAGE 属性を続けて、ストアドプロシージャまたは関数を定義します。

LANGUAGE 属性が指定されると、その関数を含むライブラリが外部プロセスによってロードされ、外部関数がその外部プロセスの一部として実行されます。この場合、関数が原因で障害が発生しても、データベースサーバは実行し続けます。

次に、プロシージャ定義の例を示します。

```
CREATE PROCEDURE ODBCinsert(
  IN ProductName CHAR(30),
  IN ProductDescription CHAR(50)
)
NO RESULT SET
EXTERNAL NAME 'ODBCexternalInsert@extodbc.dll'
LANGUAGE C_ODBC32;
```

EXTERNAL NAME clr-call 句

Microsoft .NET 関数を外部環境で呼び出すには、プロシージャインタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE CLR 属性を指定します。

CLR ストアドプロシージャまたはファンクションの動作は、SQL ストアドプロシージャまたはファンクションと同じです。ただし、プロシージャまたはファンクションのコードは Microsoft C# または Visual Basic などの Microsoft .NET 言語で記述され、その実行はデータベースサーバの外側 (つまり別の Microsoft .NET 実行ファイル内) で行われます。

```
CREATE PROCEDURE clr_interface(
    IN p1 INT,
    IN p2 UNSIGNED SMALLINT,
    OUT p3 LONG VARCHAR)
NO RESULT SET
EXTERNAL NAME 'CLRlib.dll::CLRproc.Run( int, ushort, out string )'
LANGUAGE CLR;
```

EXTERNAL NAME perl-call 句

Perl 関数を外部環境で呼び出すには、プロシージャインタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE PERL 属性を指定します。

Perl ストアドプロシージャまたは関数の動作は、SQL ストアドプロシージャまたは関数と同じです。ただし、プロシージャまたは関数のコードは Perl で記述され、その実行はデータベースサーバの外側 (つまり Perl 実行インスタンス内) で行われます。

次に、プロシージャ定義の例を示します。

```
CREATE PROCEDURE PerlWriteToConsole( IN str LONG VARCHAR)
NO RESULT SET
EXTERNAL NAME '<file=PerlConsoleExample>
    WriteToServerConsole( $sa_perl_arg0 )'
LANGUAGE PERL;
```

EXTERNAL NAME php-call 句

PHP 関数を外部環境で呼び出すには、プロシージャインタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE PHP 属性を指定します。

PHP ストアドプロシージャまたは関数の動作は、SQL ストアドプロシージャまたは関数と同じです。ただし、プロシージャまたは関数のコードは PHP で記述され、その実行はデータベースサーバの外側 (つまり PHP 実行インスタンス内) で行われます。

次に、プロシージャ定義の例を示します。

```
CREATE PROCEDURE PHPPopulateTable()
NO RESULT SET
EXTERNAL NAME '<file=ServerSidePHPExample> ServerSidePHPSub()'
LANGUAGE PHP;
```

EXTERNAL NAME java-call 句

Java メソッドを外部環境で呼び出すには、プロシージャインタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE JAVA 属性を指定します。

Java とのインタフェースとなるストアドプロシージャまたはファンクションの動作は、SQL ストアドプロシージャまたはファンクションと同じです。ただし、プロシージャまたはファンクションのコードは Java で記述され、その実行はデータベースサーバの外側 (つまり Java VM 内) で行われます。

次に、プロシージャ定義の例を示します。

```
CREATE PROCEDURE HelloDemo( IN name LONG VARCHAR )
NO RESULT SET
EXTERNAL NAME 'Hello.main([Ljava/lang/String;)V'
```

```
LANGUAGE JAVA;
```

Java メソッドの引数と戻り値の記述子には次の意味があります。

フィールドタイプ	Java データ型
B	byte
C	char
D	double
F	float
I	int
J	long
L `class-name`;	クラス `class-name` のインスタンス。クラス名は、完全に修飾された名前です。また、名前内のドットは / に置き換える必要があります。例: **`java/lang/String`**
S	short
V	void
Z	Boolean
[	配列の各次元ごとに 1 つ使用

EXTERNAL NAME js-call 句

JavaScript 関数を外部環境で呼び出すには、プロシージャインタフェースを EXTERNAL NAME 句で定義し、それに続いて LANGUAGE JS 属性を指定します。

JavaScript ストアドプロシージャまたはファンクションの動作は、SQL ストアドプロシージャまたはファンクションと同じです。ただし、プロシージャまたはファンクションのコードは JavaScript で記述され、その実行はデータベースサーバの外側 (つまり JavaScript 実行インスタンス内) で行われます。

山括弧内の EXTERNAL NAME 文字列の先頭に、JavaScript 関数の戻り値のタイプを指定します。JavaScript では関数内の単純変数の参照による受け渡しが許可されていないため、左角括弧文字 ([) は、1 つの要素の配列が JavaScript ストアドプロシージャに渡されていることを示すために S、B、I、U、または D の文字を処理できます。この構文は、ストアドプロシージャ内の INOUT および OUT パラメータをサポートするために提供されます。

次に、プロシージャ定義の例を示します。

```
CREATE PROCEDURE JSInOutDemo( INOUT num1 INT, OUT num2 INT )
  EXTERNAL NAME '<file=JSInOutParam> JSFunctionPlusOne([I[I)'
LANGUAGE JS;
```

JavaScript メソッドの引数と戻り値の記述子には次の意味があります。

フィールドタイプ	JavaScript データ型
S	文字列
B	ブール値
I	整数
U	符号なし整数

フィールドタイプ	JavaScript データ型
D	Double

備考

次の句は、句の順序が重要で、ここにリストされている順序で表示されるよう指定する必要があります。

- 結果に関する句 (RESULT、NO RESULT SET、DYNAMIC RESULT SETS)
- SQL SECURITY
- EXTERNAL NAME

CREATE PROCEDURE 文はデータベースにプロシージャを作成します。所有者を指定することによって他のユーザのプロシージャを作成できます。プロシージャは CALL 文で呼び出します。

ストアドプロシージャが結果セットを返す場合、出力パラメータを設定したり戻り値を返したりすることはできません。

複数のプロシージャからテンポラリテーブルを参照する場合、テンポラリテーブル定義が矛盾していたり、テーブルを参照する文がキャッシュされていたりすると、問題が発生する可能性があります。

Mac OS X 10.11 上での実行時に EXTERNAL NAME 句を指定する場合、ロードする必要がある `.dylib` へのフルパスを指定するか、SQL Anywhere インストールの `lib64` ディレクトリに `.dylib` ファイルを配置する必要があります。

権限

ユーザ本人が所有する外部プロシージャを作成するには、CREATE ANY PROCEDURE および CREATE EXTERNAL REFERENCE システム権限が必要です。

他のユーザが所有する外部プロシージャを作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT システム権限および CREATE EXTERNAL REFERENCE システム権限が必要です。

既存のプロシージャを置き換えるには、そのプロシージャを所有するか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

外部言語環境の CREATE PROCEDURE は ANSI/ISO SQL 標準のコア機能ですが、ソフトウェアでサポートされている一部のコンポーネントはオプションの SQL 言語機能です。これらの機能のサブセットを次に示します。

- SQL SECURITY 句は、オプションの言語機能 T324 です。
- 外部プロシージャに LONG VARCHAR、LONG NVARCHAR、または LONG BINARY 値を渡す機能は、SQL 言語機能 T041 です。
- CREATE TABLE または DROP TRIGGER などの文を使用して外部プロシージャ内でスキーマオブジェクトを作成または変更する機能は、SQL 言語機能 T653 です。
- CONNECT、EXECUTE IMMEDIATE、PREPARE、および DESCRIBE などの文を使用して外部プロシージャ内で動的 SQL 文を使用する機能は、SQL 言語機能 T654 です。
- JAVA 外部プロシージャには、SQL 言語機能 J621 が埋め込まれています。

CREATE PROCEDURE 文のいくつかの句は、標準にありません。これらを以下に示します。

- LANGUAGES 句での C_ESQL32、C_ESQL64、C_ODBC32、C_ODBC64、CLR、PERL、PHP のサポートは、標準にありません。ANSI/ISO SQL 標準では、オプションの言語機能 B122 として "C" が `environment-name` としてサポートされています。
- `external-call` のフォーマットは実装依存です。
- RESULT 句と NO RESULT SET 句は、標準にありません。ANSI/ISO SQL 標準では、RETURNS 句が使用されます。
- 特定のルーチンパラメータのオプションの DEFAULT 句は、標準にありません。
- オプションの OR REPLACE 句は標準にありません。

Transact-SQL

外部ルーチンに対する CREATE PROCEDURE は、Adaptive Server Enterprise でサポートされています。Adaptive Server Enterprise では、C 言語および Java 言語の外部ルーチンがサポートされています。

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
ALTER PROCEDURE 文 [669 ページ]
CALL 文 [756 ページ]
CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
CREATE PROCEDURE 文 [T-SQL] [888 ページ]
DROP PROCEDURE 文 [1051 ページ]
GRANT 文 [1131 ページ]
SQL データ型 [124 ページ]

1.4.4.78 CREATE PROCEDURE 文 [Web サービス]

HTTP サーバへの HTTP 要求または SOAP 要求を作成するユーザ定義の Web クライアントプロシージャを作成します。

構文

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure-name ( [ parameter, ... ] )
[ RESULT ( attribute-column-name datatype, value-column-name datatype ) ]
URL url-string
[ TYPE { http-type-spec-string | soap-type-spec-string } ]
[ HEADER header-string ]
[ CERTIFICATE certificate-string ]
[ CLIENTPORT clientport-string ]
[ PROXY proxy-string ]
[ SET protocol-option-string ]
[ SOAPHEADER soap-header-string ]
[ NAMESPACE namespace-string ]
```

```
http-type-spec-string :
HTTP[: { GET
   | POST[:MIME-type ]
   | PUT[:MIME-type ]
   | DELETE
   | HEAD
   | OPTIONS } ]
```

```
soap-type-spec-string :
SOAP[:{ RPC | DOC }
```

```
parameter :
  parameter-mode parameter-name datatype [ DEFAULT expression ]
```

```
parameter-mode :
  IN
| OUT
| INOUT
```

```
url-string :
{ HTTP | HTTPS | HTTPS_FIPS }://[user:password@]hostname[:port][/path]
```

```
protocol-option-string : option-list [, option-list ...]
```

```
option-list :
  HTTP( http-option [ ;http-option ...] )
| SOAP( soap-option [ ;soap-option ...] )
| REDIR( redir-option [ ;redir-option ...] )
```

```
http-option :
  CHUNK={ ON | OFF | AUTO }
| EXCEPTIONS={ ON | OFF | AUTO }
| VERSION={ 1.0 | 1.1 }
| KTIMEOUT=number-of-seconds
```

```
soap-option :
  OPERATION=soap-operation-name
```

```
redir-option :
  COUNT=count
| STATUS=status-list
```

パラメータ

OR REPLACE clause

CREATE OR REPLACE PROCEDURE を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存の権限は保持されます。使用中のプロシージャを置き換えようとすると、エラーが返されます。

procedure-name

プロシージャ名。

parameter-name

パラメータ名は、カラム名など他のデータベース識別子に対するルールに従って付けてください。これらは有効な SQL データ型を持つ必要があります。

パラメータがデフォルト値を持つ場合、これを指定する必要はありません。デフォルト値を持たないパラメータを指定する必要があります。

パラメータには、IN、OUT、INOUT のいずれかのキーワードをプレフィクスとして付けることができます。これらの値のいずれも指定しない場合、パラメータはデフォルトで INOUT になります。キーワードには次の意味があります。

IN

このパラメータは、プロシージャに値を与える式です。

OUT

このパラメータは、プロシージャから値を受け取ることがある変数です。

INOUT

このパラメータはプロシージャに値を与え、プロシージャから新しい値を受け取ることがある変数です。

datatype

パラメータのデータ型。データ型を明示的に設定するか、%TYPE または %ROWTYPE 属性を指定し、データ型をデータベース内の別のオブジェクトのデータ型に設定します。%TYPE を使用して、テーブルまたはビュー内のカラムのデータ型に設定します。%ROWTYPE を使用して、テーブルまたはビュー内のローから派生する複合データ型に設定します。ただし、テーブル参照変数に設定されている %ROWTYPE (`TABLE REF (table-reference-variable) %ROWTYPE` を使用したデータ型の定義は許可されていません。

FLOAT、INT などの型指定のあるデータの転送をサポートするのは SOAP 要求のみです。HTTP 要求は文字列の転送のみサポートしているため、CHAR 型に制限されます。

RESULT clause

RESULT 句は SELECT 文でプロシージャを使用するために必要です。RESULT 句は 2 つのカラムを返します。1 番目のカラムには HTTP 応答ヘッダ、ステータス、および応答本文属性が含まれ、2 番目のカラムにはこれらの属性の値が含まれます。RESULT 句では 2 文字のデータ型を指定します。たとえば、VARCHAR や LONG VARCHAR などです。RESULT 句が指定されていない場合、デフォルトのカラム名は Attribute および Value で、それらのデータ型は LONG VARCHAR です。

URL clause

Web サービスの URI を指定します。オプションのユーザ名とパスワードのパラメータは、HTTP 基本認証に必要なクレデンシャルとして機能します。HTTP 基本認証は、ユーザとパスワードの情報を base-64 でエンコードし、HTTP 要求の Authentication ヘッダに渡します。この方法で指定すると、ユーザ名とパスワードは URL の一部として暗号化されずに渡されます。

HTTP:GET タイプのプロシージャの場合、URL 句内および (自動的に生成されて) プロシージャに渡されたパラメータからの両方でクエリパラメータを指定できるようになりました。

```
URL 'http://localhost/service?parm=1
```

HTTPS_FIPS を指定すると、強制的に FIPS 認定ライブラリが使用されます。HTTPS_FIPS を指定したときに、FIPS 認定ライブラリがない場合は、代わりに FIPS 認定ではないライブラリが使用されます。

オペレーティングシステムの証明書ストアの証明書を使用するには、**`https://`** で始まる URL を指定します。

TYPE clause

Web サービス要求を行う場合に使用するフォーマットを指定します。SOAP が指定されている場合、または TYPE 句が含まれていない場合は、SOAP:RPC が使用されます。HTTP が指定されている場合は、HTTP:POST が使用されます。

TYPE 句では、HTTP:POST および HTTP:PUT の各タイプに MIME タイプを指定できます。HTTP:PUT が使用されている場合は、MIME タイプを指定する必要があります。`MIME-type` 指定は、Content-Type 要求ヘッダや操作モードの設定に使用することにより、1 つの呼び出しパラメータのみで要求の本文を設定できます。パラメータの置換後に Web サービスのストアドプロシージャ呼び出しを行う場合は、パラメータがまったくなくなるか、1 つだけ残される場合があります。Web サービスプロシージャを (置換後に) NULL 値またはパラメータなしで呼び出すと、本文がなくコンテンツ長が 0 の要求になります。MIME タイプが指定されている場合は、要求内で単一の本文パラメータが送信されます。このため、アプリケーションは、コンテンツのフォーマットを MIME タイプに一致させる必要があります。

一般的な MIME タイプの例を次に示します。

- text/plain
- text/html
- text/xml

MIME タイプが指定されていない場合は、パラメータの名前と値 (複数のパラメータが可能) は、HTTP 要求の本文内で URL にエンコードされます。

TYPE 句のキーワードには次の意味があります。

HTTP:GET

デフォルトでは、URL で指定されたパラメータのエンコード用に、このタイプで MIME タイプ application/x-www-form-urlencoded が使用されます。

たとえば、クライアントが URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` から要求を送信すると、次の要求が生成されます。

```
GET /WebServiceName?arg1=param1&arg2=param2 HTTP/1.1
// <End of Request - NO BODY>
```

HTTP:POST

デフォルトでは、POST 要求の本文で指定されたパラメータのエンコード用に、このタイプで MIME タイプ application/x-www-form-urlencoded が使用されます。URL パラメータは、要求の本文に格納されます。

たとえば、クライアントが URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` から要求を送信すると、次の要求が生成されます。

```
POST /WebServiceName HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 19
arg1=param1&arg2=param2
// <End of Request>
```

HTTP:PUT

HTTP:PUT は HTTP:POST に似ていますが、HTTP:PUT タイプにはデフォルトのメディアタイプはありません。

次の例は、*%SQLANYSAMP17%*`¥SQLAnywhere¥HTTP¥put_data.sql` サンプルを実行しているデータベースサーバにデータをアップロードする、汎用のクライアントプロシージャを設定する方法を示します。

```
CREATE OR REPLACE PROCEDURE CPUT([data] LONG VARCHAR, resnm LONG VARCHAR,
mediatype LONG VARCHAR)
    URL 'http://localhost/resource/!resnm'
    TYPE 'HTTP:PUT:!mediatype';
CALL CPUT('hello world', 'hello', 'text/plain' );
```

HTTP:DELETE

サーバにあるリソースを削除する Web サービスクライアントプロシージャを設定できます。メディアタイプの指定はオプションです。

次の例は、put_data.sql サンプルを実行しているデータベースサーバからリソースを削除する、汎用のクライアントプロシージャを設定する方法を示します。

```
CREATE OR REPLACE PROCEDURE CDEL(resnm LONG VARCHAR)
    URL 'http://localhost/resource/!resnm'
    TYPE 'HTTP:DELETE';
CALL CDEL('hello', 'text/plain' );
```

HTTP:HEAD

HEAD メソッドは GET メソッドと同じですが、サーバは本文を返しません。メディアタイプを指定できます。

```
CREATE OR REPLACE PROCEDURE CHEAD(resnm LONG VARCHAR)
    URL 'http://localhost/resource/!resnm'
    TYPE 'HTTP:HEAD';
CALL CHEAD( 'hello' );
```

HTTP:OPTIONS

OPTIONS メソッドは GET メソッドと同じですが、サーバは本文を返しません。メディアタイプを指定できます。このメソッドは CORS (クロスオリジンリソース共有) が可能です。

SOAP:RPC

このタイプは、Content-Type ヘッダを 'text/xml' に設定します。SOAP 操作と SOAP パラメータは、SOAP エンベロープの XML ドキュメントにカプセル化されます。

SOAP:DOC

このタイプは、Content-Type ヘッダを 'text/xml' に設定します。SOAP:RPC タイプに似ていますが、より多くのデータ型を送信できます。SOAP 操作と SOAP パラメータは、SOAP エンベロープの XML ドキュメントにカプセル化されます。

TYPE 句に MIME タイプを指定すると、その MIME タイプに Content-Type ヘッダが自動的に設定されます。

HEADER clause

HTTP Web サービスクライアントプロシージャを作成する場合は、この句を使用して、HTTP 要求ヘッダのエントリを追加、変更、または削除します。ヘッダの仕様は、RFC2616 Hypertext Transfer Protocol の HTTP/1.1 および RFC822 Standard for ARPA Internet Text Messages に非常に近いものになっています。たとえば、HTTP ヘッダに指定できるのは印字可能な ASCII 文字のみで、大文字と小文字は区別されません。

ヘッダは、`header-name:value-name` ペアとして定義できます。各ヘッダとその値は、コロン (:) を使用して区切ります。このため、ヘッダにコロンは使用できません。各ペアを ¥n、¥x0d¥n、<LF> (改行)、または <CR><LF> (キャリッジリターンに続く改行) で区切ることで、複数のヘッダを定義できます。

ヘッダ内の連続する複数のスペースは、単一の空白文字に変換されます。

CERTIFICATE clause

安全な (HTTPS) 要求を行うには、HTTPS サーバの証明書の正味に使用される証明書 (または署名チェイン内の上位にある証明書) にクライアントがアクセスできる必要があります。必要な情報は、セミコロンで区切られたキーワード/値のペアの文字列で指定されます。次のキーワードを使用できます。

キーワード	省略形	説明
file		証明書のファイル名。オペレーティングシステムの証明書ストアにある証明書を使用する場合は * を指定します。certificate または certificate_name キーワードが指定されているときは指定できません。
certificate	cert	証明書自体 file または certificate_name キーワードが指定されているときは指定できません。
certificate_name	cert_name	データベースに保存される証明書の名前 file または certificate キーワードが指定されているときは指定できません。
company	co	証明書で指定された会社
unit		証明書で指定された会社の部署
name		証明書で指定された通称
skip_certificate_name_check		クライアントライブラリがサーバホスト名とデータベースサーバの証明書ホスト名との照合を省略するかどうかを制御します。
allow_expired_certs		クライアントライブラリが期限切れまたはまだ有効化されていないルート証明書やデータベースサーバ証明書を受け入れるかどうかを制御します。

証明書は、HTTPS サーバに対する要求、または安全でないサーバから安全なサーバにリダイレクトされる可能性がある要求に対してのみ必要です。PEM でフォーマットされた証明書のみがサポートされています。

CLIENTPORT clause

HTTP クライアントプロシージャが TCP/IP を使用して通信するポート番号を識別します。これは "送信" TCP/IP 接続をフィルタするファイアウォール経由で接続するためのもので、この用途にかぎり推奨されています。単一のポート番号、ポート番号の範囲、または両方の組み合わせを指定できます。たとえば、`CLIENTPORT '85,90-97'` を指定できます。

PROXY clause

プロキシサーバの URI を指定します。クライアントがプロキシを介してネットワークにアクセスする場合に使用します。`proxy-string` は、通常 HTTP または HTTPS の url-string です。これは、通常ネットワーク管理者から取得する必要があるサイト固有の情報です。この句は、プロシージャがプロキシサーバに接続し、そのプロキシサーバを介して Web サービスに要求を送信することを示します。たとえば、次の PROXY 句はプロキシサーバを `proxy.example.com` に設定します。

```
PROXY http://proxy.example.com
```

SET clause

HTTP、SOAP、および REDIR (リダイレクト) のプロトコル固有動作オプションを指定します。SET 句は 1 つしか使用できません。次の表に、サポートされている SET オプションを示します。CHUNK、EXCEPTIONS、VERSION、KTIMEOUT は HTTP プロトコルに適用され、OPERATION は SOAP プロトコルに適用され、COUNT と STATUS は REDIR オプションに適用されます。REDIR オプションは、HTTP または SOAP プロトコルオプションとともに含めることができます。

CHUNK={ ON | OFF | AUTO }

(省略形 CH) この HTTP オプションでは、チャンクを使用するかどうかを指定します。チャンクを使用すると、HTTP メッセージを複数の部分に分割できます。可能な値は、ON (常にチャンクを使用)、OFF (チャンクは使用しない)、AUTO (内容が 8196 バイトを超えた場合にのみチャンクを使用、ただし自動生成されたマークアップを除く) です。たとえば、次の SET 句を使用すると、チャンクは有効になります。

```
SET 'HTTP(CHUNK=ON)'
```

CHUNK オプションを指定しないときの、デフォルトの動作は AUTO です。チャンクされた要求が、AUTO モードのときに、ステータスが 505 `HTTP Version Not Supported`、501 `Not Implemented`、または 411 `Length Required` で失敗した場合、クライアントはチャンクされた転送コーディングなしで要求をリトライします。

チャンクされた転送コーディングによる要求を HTTP サーバがサポートしない場合は、CHUNK オプションを OFF (チャンクを使用しない) に設定してください。

CHUNK モードは HTTP バージョン 1.1 からサポートされる転送エンコードです。CHUNK を ON に設定する場合、バージョン (VER) が 1.1 に設定される必要がありますが、デフォルトのバージョンとして 1.1 を使用する場合は、VER に何も設定しないでください。

EXCEPTIONS={ ON | OFF | AUTO }

(省略形 EX) この HTTP オプションは、ステータスコードの処理を制御することを可能にします。デフォルトは ON です。

ON または AUTO に設定すると、HTTP クライアントプロシージャは HTTP ステータスコードが成功の場合 (1XX や 2XX) に結果セットを返し、これ以外のすべてのコードの場合に SQLE_HTTP_REQUEST_FAILED 例外を発生させます。

```
SET 'HTTP(EXCEPTIONS=AUTO)'
```

OFF に設定すると、HTTP クライアントプロシージャは、HTTP ステータスコードとは関係なく常に結果セットを返します。属性カラムに *Status* という語が入っている結果ローの値カラムに HTTP ステータスコードが含まれます。

HTTP ステータスコードと関連のない例外 (たとえば SQLE_UNABLE_TO_CONNECT_TO_HOST) は、EXCEPTIONS 設定とは関係なく、適宜発生します。

VERSION={ 1.0 | 1.1 }

(省略形 VER) この HTTP オプションでは、HTTP メッセージのフォーマットに使用する HTTP プロトコルのバージョンを指定できます。たとえば、次の SET 句は HTTP のバージョンを 1.1 に設定します。

```
SET 'HTTP(VERSION=1.1)'
```

可能な値は 1.0 と 1.1 です。VERSION が指定されていない場合は、次のようになります。

- CHUNK が ON に設定された場合、1.1 が HTTP バージョンとして使用される
- CHUNK が OFF に設定された場合、1.0 が HTTP バージョンとして使用される
- CHUNK が AUTO に設定された場合、クライアントが CHUNK モードで送信しているかどうかによって 1.0 か 1.1 が使用される

KTIMEOUT=number-of-seconds

(省略形 KTO) この HTTP オプションは、キープアライブタイムアウト条件を指定して、Web クライアントプロシージャがキープアライブ HTTP/HTTPS 接続をインスタンス化し、一定期間キャッシュできるようにします。HTTP キープアライブ接続をキャッシュするには、HTTP バージョンは 1.1 に設定し、KTIMEOUT は非ゼロ値に設定する必要があります。HTTP と HTTPS との間でパフォーマンスに大きな違いを感じている場合、HTTPS 接続にとって KTIMEOUT は特に役立ちます。データベース接続でキャッシュできるのは 1 つのキープアライブ HTTP 接続だけです。同じ URI を使用する Web クライアントプロシージャへのその後の呼び出しでは、キープアライブ接続が再利用されます。このため、実行する Web クライアント呼び出しに含まれる URI のスキーマ、宛先ホスト、ポートが、キャッシュされた URI と一致することが必要です。また、HEADER 句で Connection: close を指定しない必要があります。KTIMEOUT が指定されないか、ゼロに設定されている場合、HTTP/HTTPS 接続はキャッシュされません。

OPERATION=soap-operation-name

(省略形 OP) この SOAP オプションでは、SOAP 操作の名前を指定します (作成しているプロシージャの名前と異なる場合)。OPERATION の値は、リモートプロシージャコールの名前に似ています。たとえば、login という SOAP 操作を呼び出す accounts_login というプロシージャを作成する場合は、次のように指定します。

```
CREATE PROCEDURE accounts_login( name LONG VARCHAR, pwd LONG VARCHAR )
   SET 'SOAP(OPERATION=login)'
```

OPERATION オプションを指定しない場合、SOAP 操作の名前は、作成するプロシージャの名前と一致する必要があります。

COUNT=count

(省略形 CNT) この REDIR オプションはリダイレクトの制御を可能にします。以下の "ステータス" を参照してください。

STATUS=status-list

(省略形 STAT) この REDIR オプションはリダイレクトの制御を可能にします。*302 Found* および *303 See Other* などの HTTP 応答ステータスコードを使用して、特に HTTP POST を実行した後に Web アプリケーションを新しい URI にリダイレクトします。たとえば、クライアント要求は次のようにすることができます。

```
GET /people/alice HTTP/1.1
Host: www.example.com
Accept: text/html, application/xhtml+xml
Accept-Language: en, de
```

web サーバ応答は次のようにすることができます。

```
HTTP/1.1 302 Found
Location: http://www.example.com/people/alice.en.html
```

応答では、クライアントは別の HTTP 要求を新しい URI に送信します。REDIR オプションを使用して、許可されるリダイレクションの最大数および自動的にリダイレクトする HTTP 応答ステータスコードを制御できます。

たとえば、`SET 'REDIR(COUNT=3; STATUS=301,307)'` では、最大制限の 3 リダイレクションおよび 301 と 307 ステータスのリダイレクションが可能です。302 や 303 などの他のリダイレクションステータスコードが受信されると、エラーが発行されます (SQLE_HTTP_REQUEST_FAILED)。

デフォルトのリダイレクション制限 `count` は 5 です。デフォルトでは、HTTP クライアントプロシージャがすべての HTTP リダイレクションステータスコード (301、302、303、307) に応答して自動的にリダイレクトします。すべてのリダイレクションステータスコードを許可しないようにするには、`SET 'REDIR(COUNT=0)'` を使用します。このモードでは、リダイレクション応答はエラーになりません (SQLE_HTTP_REQUEST_FAILED)。その代わりに、HTTP ステータスと応答ヘッダが結果セットとともに返されます。これにより、呼び出し側は *Location* ヘッダに含まれる URI に基づいて、条件付きで要求を再発行できます。

303 See Other ステータスを受信する POST HTTP メソッドを指定する Web サービスプロシージャは、GET HTTP メソッドを使用してリダイレクト要求を発行します。

Location ヘッダには、絶対パスまたは相対パスのいずれかを含むことができます。HTTP クライアントプロシージャがいずれかを処理します。このヘッダには、クエリパラメータを含めることもでき、これらのパラメータはリダイレクトされたロケーションに転送されます。たとえば、ヘッダに次のようなパラメータが含まれる場合、後続の GET または POST にこれらのパラメータが含まれます。

```
Location: alternate_service?a=1&b=2
```

上の例では、クエリパラメータは `a=1&b=2` です。

複数のオプション設定を 1 つの SET 句で組み合わせると、次の例のようになります。

```
CREATE PROCEDURE accounts_login( name LONG VARCHAR, pwd LONG VARCHAR )
   SET 'HTTP( CHUNK=ON; VERSION=1.1 ), REDIR(COUNT=5;STATUS=302,303)'
   ...
```

次の例は、大文字と小文字から成る省略形の使用を示します。

```
CREATE PROCEDURE accounts_login( name LONG VARCHAR, pwd LONG VARCHAR )
   SET 'HTTP( CH=ON; Ver=1.1 ), REDIR(CNT=5;Stat=302,303)'
   ...
```

SOAPHEADER clause

(SOAP フォーマットのみ) SOAP Web サービスをプロシージャとして宣言する場合は、この句を使用して 1 つ以上の SOAP 要求ヘッダエントリを指定します。SOAP ヘッダは、静的定数として宣言したり、代入パラメータメカニズムを使用して動的に設定したりできます (hd1、hd2 などに IN、OUT、または INOUT パラメータを宣言)。Web サービスプロシージャでは、1 つ以上の IN モード代入パラメータと、1 つの INOUT または OUT 代入パラメータを定義できます。

次の例は、クライアントが、いくつかのヘッダエントリを代入パラメータを使用して送信し、応答 SOAP ヘッダデータを受信するよう指定する方法を示しています。

```
CREATE PROCEDURE soap_client(INOUT hd1 LONG VARCHAR, IN hd2 LONG VARCHAR, IN hd3
LONG VARCHAR)
   URL 'localhost/some_endpoint'
   SOAPHEADER '!hd1!hd2!hd3';
```

NAMESPACE clause

(SOAP フォーマットのみ) この句は、SOAP:RPC 要求と SOAP:DOC 要求の両方に通常必要なメソッドネームスペースを示します。要求を処理する SOAP サーバは、このネームスペースを使用して、SOAP 要求メッセージ本文内のエンティティの名前を解釈します。ネームスペースは、Web サービスサーバから使用できる SOAP サービスの WSDL (Web Services Description Language) から取得できます。デフォルト値は、プロシージャの URL のオプションのパスコンポーネントの直前までです。

`namespace-string` の変数名を指定できます。変数が NULL の場合、ネームスペースプロパティは無視されます。

備考

パラメータ値は、要求の一部として渡されます。使用される構文は、要求のタイプによって決まります。HTTP:GET の場合、パラメータは URL の一部として渡されます。HTTP:POST 要求の場合、値は要求の本文に置かれます。SOAP 要求へのパラメータは、常に要求本文にバンドルされます。

Web サービスクライアントプロシージャを作成または置換できます。PROC は PROCEDURE の同義語として使用できます。

SOAP 要求の場合、プロシージャ名はデフォルトで SOAP 操作名として使用されます。詳細については、SET 句を参照してください。

TEMPORARY Web サービスプロシージャを作成することはできません。

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

ユーザ本人が所有するプロシージャを作成するには、CREATE PROCEDURE システム権限が必要です。

他のユーザが所有するプロシージャを作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT システム権限が必要です。

既存のプロシージャを置き換えるには、そのプロシージャを所有するか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

Adaptive Server Enterprise によってサポートされません。

例

1. 次の例は、FtoC という Web サービスクライアントプロシージャを作成します。

```
CREATE PROCEDURE FtoC( IN temperature FLOAT,
    INOUT inoutheader LONG VARCHAR,
    IN inheader LONG VARCHAR )
  URL 'http://localhost:8082/FtoCService'
  TYPE 'SOAP:DOC'
  SOAPHEADER '!inoutheader!inheader';
```

2. 次の例は、データベースに格納されている証明書を使用する SecureSendWithMimeType というセキュアな Web サービスクライアントプロシージャを作成します。

```
CREATE CERTIFICATE client_cert
FROM FILE 'C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥Certificates
¥¥rsaroot.crt';
CREATE PROCEDURE SecureSendWithMimeType(
    value LONG VARCHAR,
    mimeType LONG VARCHAR,
    urlSpec LONG VARCHAR
)
URL '!urlSpec'
CERTIFICATE 'certificate_name=client_cert'
TYPE 'HTTPS:POST:!mimeType';
CALL SecureSendWithMimeType('<hello>this is xml</hello>',
    'text/xml',
    'https://localhost:4043/EchoService'
);
```

3. 次の例は、オペレーティングシステムの証明書ストアにある証明書を使用する SecureSendWithMimeType というプロシージャを作成します。

```
CREATE PROCEDURE SecureSendWithMimeType(
    value LONG VARCHAR,
    mimeType LONG VARCHAR,
    urlSpec LONG VARCHAR
)
URL '!urlSpec'
CERTIFICATE 'file=*'
TYPE 'HTTPS:POST:!mimeType';
```

4. 次の例は、証明書 myrootcert.crt がデータベースサーバの証明書署名チェーンのルートであることを検証し、その他の検証は行わない SecureSendWithMimeType というプロシージャを作成します。

```
CREATE PROCEDURE SecureSendWithMimeType(
    value LONG VARCHAR,
    mimeType LONG VARCHAR,
    urlSpec LONG VARCHAR
)
URL '!urlSpec'
CERTIFICATE 'file=myrootcert.crt;skip_certificate_name_check=ON'
TYPE 'HTTPS:POST:!mimeType';
```

5. 次の例は、NAMESPACE 句の変数を使用してプロシージャを作成します。
   1. 次の文では、NAMESPACE 句の変数を作成します。

```
CREATE VARIABLE @ns LONG VARCHAR
SET @ns = 'http://wsdl.domain.com/';
```

   2. 次の文は、NAMESPACE 句の変数を使用する FtoC という名前のプロシージャを作成します。

```
CREATE PROCEDURE FtoC( IN temperature FLOAT,
    INOUT inoutheader LONG VARCHAR,
    IN inheader LONG VARCHAR )
  URL 'http://localhost:8082/FtoCService'
  TYPE 'SOAP:DOC'
  SOAPHEADER '!inoutheader!inheader'
  NAMESPACE @ns;
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
ALTER PROCEDURE 文 [669 ページ]
CALL 文 [756 ページ]
CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [T-SQL] [888 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
DROP PROCEDURE 文 [1051 ページ]
GRANT 文 [1131 ページ]

1.4.4.79 CREATE PROCEDURE 文 [T-SQL]

Adaptive Server Enterprise 互換の方法で、データベース内にプロシージャを作成します。

構文

次の Transact-SQL CREATE PROCEDURE 文のサブセットが SQL Anywhere でサポートされています。

```
CREATE [ OR REPLACE ]PROCEDURE [owner.]procedure-name
[ NO RESULT SET ]
[ [ ( ] @parameter-name data-type [ = default ] [ OUTPUT ], ... [ ) ] ]
[ WITH RECOMPILE ] AS statement-list
```

パラメータ

OR REPLACE clause

CREATE OR REPLACE PROCEDURE を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存の権限は保持されます。使用中のプロシージャを置き換えようとすると、エラーが返されます。

NO RESULT SET clause

このプロシージャによって結果セットが返されないことを宣言します。この句は、プロシージャが結果セットを返さないことを外部環境から知る必要がある場合に役立ちます。

WITH RECOMPILE clause

この句は、Transact-SQL との互換性を保つために実装されていますが、無視されます。SQL Anywhere は、常にデータベース起動後に初めて実行されたプロシージャを再コンパイルし、コンパイルされたプロシージャをデータベースが停止するまで保管します。

備考

Transact-SQL と SQL Anywhere 文 (Watcom SQL) の間の次の相違点のリストは、両方のダイアレクトで記述する場合に役立ちます。

@ をプレフィクスとする変数名

@ 符号は Transact-SQL 変数名を示します。Watcom-SQL では、変数には有効な識別子であればどれでも使用でき、@ プレフィクスはオプションです。

入出力パラメータ

Watcom SQL プロシージャパラメータは、デフォルトでは INOUT であり、IN、OUT、または INOUT としても指定できます。Transact-SQL プロシージャパラメータは、デフォルトでは INPUT パラメータです。OUTPUT キーワードを追加すると、入出力として指定できます。Transact-SQL ダイアレクトには、出力専用のパラメータはありません。

Watcom SQL ダイアレクトを使用してパラメータ OUT を宣言すると、出力専用になります。ダイアレクトの混在はおすすめできません。これは、プロシージャ宣言をアップロードしたり、プロシージャ宣言を使用してデータベースを再構築したりする場合に問題が生じるからです。プロシージャ宣言をアップロードしたり、プロシージャ宣言を使用してデータベースを再構築したりすると、再構築プロシージャ宣言は Transact-SQL ダイアレクトとなり、OUTPUT キーワードが使用され、パラメータは入出力になります。

パラメータのデフォルト値

Watcom SQL のプロシージャパラメータには、キーワード DEFAULT を使用してデフォルト値を設定します。Transact-SQL では等号 (=) を使用してデフォルト値を設定します。

結果セットを返す

Watcom SQL は RESULT 句を使用して、返される結果セットを指定します。Transact-SQL プロシージャでは、最初のクエリのカラム名またはエイリアス名が呼び出し環境に返されます。

次の Transact-SQL プロシージャは、結果セットが Transact-SQL ストアドプロシージャから返される方法を示します。

```
CREATE PROCEDURE showdept @deptname varchar(30)
AS
   SELECT Employees.Surname, Employees.GivenName
```

```
   FROM Departments, Employees
   WHERE Departments.DepartmentName = @deptname
   AND Departments.DepartmentID = Employees.DepartmentID;
```

次に、これに対応する Watcom SQL のプロシージャを示します。

```
CREATE PROCEDURE showdept2(in deptname
         varchar(30) )
RESULT ( lastname char(20), firstname char(20))
ON EXCEPTION RESUME
BEGIN
   SELECT Employees.Surname, Employees.GivenName
   FROM Departments, Employees
   WHERE Departments.DepartmentName = deptname
   AND Departments.DepartmentID = Employees.DepartmentID
END;
```

プロシージャ本体

Transact-SQL プロシージャの本体は、AS キーワードをプレフィクスとして付けた Transact-SQL 文のリストです。Watcom SQL プロシージャの本文は、BEGIN と END キーワードで囲まれた複合文です。

権限

ユーザ本人が所有するプロシージャを作成するには、CREATE PROCEDURE 権限が必要です。

他のユーザが所有するプロシージャを作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT の権限が必要です。

既存のプロシージャを置き換えるには、そのプロシージャを所有するか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

SQL Anywhere は Adaptive Server Enterprise の CREATE PROCEDURE 文の構文のサブセットをサポートしています。

SQL Anywhere の Transact-SQL ダイアレクトでは、Transact-SQL の SQL プロシージャのみがサポートされています。外部プロシージャを作成するには、Watcom SQL の構文を使用してください。Adaptive Server Enterprise では、NO RESULT SET 句はサポートされていません。Transact-SQL WITH RECOMPILE オプション句があっても無視されます。SQL Anywhere は、常にデータベース起動後に初めて実行されたプロシージャを再コンパイルし、コンパイルされたプロシージャをデータベースが停止するまで保管します。

SQL Anywhere では、Transact-SQL プロシージャのグループはサポートされていません。

関連情報

CREATE FUNCTION 文 [837 ページ]
CREATE PROCEDURE 文 [891 ページ]

1.4.4.80 CREATE PROCEDURE 文

データベースにユーザ定義 SQL プロシージャを作成します。

構文

```
CREATE [ OR REPLACE | TEMPORARY ] PROCEDURE [ owner.]procedure-name
( [ parameter, ... ] )
[ RESULT ( result-column, ... ) | NO RESULT SET ]
[ SQL SECURITY { INVOKER | DEFINER } ]
[ ON EXCEPTION RESUME ]
compound-statement | AT location-string
```

```
parameter :
parameter-mode parameter-name data-type [ DEFAULT expression ]
| SQLCODE
| SQLSTATE
```

```
parameter-mode :
IN
| OUT
| INOUT
```

```
result-column : column-name data-type
```

パラメータ

OR REPLACE 句

CREATE OR REPLACE PROCEDURE を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存の権限は保持されます。使用中のプロシージャを置き換えようとすると、エラーが返されます。

TEMPORARY 句

CREATE TEMPORARY PROCEDURE を指定すると、作成した接続でのみ参照できるストアドプロシージャになり、接続を削除するとプロシージャも自動的に削除されます。テンポラリストアドプロシージャを明示的に削除することもできます。テンポラリストアドプロシージャに対して ALTER、GRANT、または REVOKE は実行できません。また他の関数とは異なり、テンポラリストアドプロシージャはカタログやトランザクションログに記録されていません。

テンポラリプロシージャは、作成者 (現在のユーザ) または指定された所有者の権限で実行されます。テンポラリプロシージャに所有者を指定できるのは次の場合です。

- テンポラリプロシージャが永続的なストアドプロシージャ内に作成された場合
- テンポラリプロシージャと永続的なプロシージャとで所有者が同じ場合

テンポラリプロシージャの所有者を削除するには、テンポラリプロシージャを先に削除する必要があります。

読み込み専用データベースに接続し、外部プロシージャにすることができない場合、テンポラリストアドプロシージャを作成または削除できます。

たとえば、次のテンポラリプロシージャは CustRank という架空のテーブルを削除します。この例では、テーブル名が一意であり、プロシージャの作成者がテーブルの所有者を指定しなくても参照できると想定しています。

```
CREATE TEMPORARY PROCEDURE drop_table( IN @TableName char(128) )
BEGIN
    IF EXISTS  ( SELECT * FROM SYS.SYSTAB WHERE table_name = @TableName ) THEN
       EXECUTE IMMEDIATE 'DROP TABLE "' || @TableName || '"';
       MESSAGE 'Table "' || @TableName || '" dropped' to client;
    END IF;
END;
CALL drop_table( 'CustRank' );
```

parameter

パラメータ名は、カラム名など他のデータベース識別子に対するルールに従って付けてください。これらは有効な SQL データ型にする必要があります。

パラメータには、IN、OUT、INOUT のいずれかのキーワードをプレフィクスとして付けることができます。これらの値のいずれも指定しない場合、パラメータはデフォルトで INOUT になります。キーワードには次の意味があります。

IN

このパラメータは、プロシージャに値を与える式です。

OUT

このパラメータは、プロシージャから値を受け取ることがある変数です。

INOUT

このパラメータはプロシージャに値を与え、プロシージャから新しい値を受け取ることがある変数です。

データ型を明示的に設定するか、%TYPE または %ROWTYPE 属性を指定し、データ型をデータベース内の別のオブジェクトのデータ型に設定します。%TYPE を使用して、テーブルまたはビュー内のカラムのデータ型に設定します。%ROWTYPE を使用して、テーブルまたはビュー内のローから派生する複合データ型に設定します。ただし、テーブル参照変数に設定されている %ROWTYPE (`TABLE REF (table-reference-variable) %ROWTYPE` を使用したデータ型の定義は許可されていません。

CALL 文を使ってプロシージャを実行する場合、必ずしもすべてのパラメータを指定する必要はありません。CREATE PROCEDURE 文の中にデフォルト値がある場合、不明のパラメータにデフォルト値を割り当てます。CALL に引数が指定されておらず、デフォルトも設定されていない場合には、エラーが発生します。

SQLSTATE と SQLCODE は、プロシージャが終了するときに、SQLSTATE または SQLCODE 値を出力する、特別な OUT パラメータです。SQLSTATE と SQLCODE の特別値は、プロシージャのリターンステータスのテストを目的として、プロシージャ呼び出しの直後にチェックできます。

SQLSTATE と SQLCODE 特別値は、その次の SQL 文によって修正されます。SQLSTATE と SQLCODE をプロシージャ引数として与えると、リターンコードは変数の中に格納されます。

CREATE OR REPLACE PROCEDURE を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存の権限は保持されます。OR REPLACE をテンポラリプロシージャで使用することはできません。置き換え対象のプロシージャが使用中の場合は、エラーが返されます。CREATE OR REPLACE PROCEDURE 文が実行されると、接続のためのオープンカーソルが閉じられます。

RESULT 句

RESULT 句は結果セットのカラムの数と型を宣言します。RESULT キーワードに続く括弧で囲まれたリストは、結果カラムの名前と型を定義します。CALL 文が記述されていると、この情報を Embedded SQL DESCRIBE または ODBC SQLDescribeCol が返します。

%TYPE または %ROWTYPE 属性を使用して、結果セット内のカラムのデータ型を定義できます。テーブルまたはビュー内のカラムのデータ型に設定する場合は、%TYPE を使用します。テーブルまたはビュー内のローから派生する複合データ型に設定する場合は、%ROWTYPE を使用します。

RESULT 句を指定する場合は、文の最初の句として指定する必要があります。

プロシージャは、その実行方法に応じて、それぞれカラム数が異なる複数の結果セットを生成する場合があります。たとえば次のプロシージャは、2 カラムを返す場合も、1 カラムを返す場合もあります。

```
CREATE PROCEDURE names( IN formal char(1))
BEGIN
    IF formal = 'n' THEN
        SELECT GivenName
        FROM GROUPO.Employees
    ELSE
        SELECT Surname, GivenName
        FROM GROUPO.Employees
    END IF
END;
```

これらの結果セットプロシージャは RESULT 句を指定しないで記述するか、Transact-SQL で記述します。これらの使用には、次の制約があります。

Embedded SQL

正しい形式の結果セットを取得するには、結果セットのカーソルが開かれてからローが返されるまでの間に、プロシージャコールを記述 (DESCRIBE) します。DESCRIBE 文の CURSOR `cursor-name` 句は必須です。

ODBC、OLE DB、ADO.NET

変数結果セットプロシージャは、これらのインタフェースを使用するアプリケーションで使用できます。結果セットの記述は、ドライバまたはプロバイダによって実行されます。

Open Client アプリケーション

変数結果セットプロシージャは Open Client アプリケーションで使用できます。

Web サービス

Web サービスはストアドプロシージャの RESULTS 句を使用して、結果セットのカラムの数と型を指定します。Web サービスは、複数の結果セットを返すプロシージャや EXECUTE IMMEDIATE を使用する変数結果セットには対応していません。

> **注記**
>
> WITH RESULT SET ON 句を含む EXECUTE IMMEDIATE 文をプロシージャで使用しており、文から返される結果セットがプロシージャから返される結果セットと同じである場合には、EXECUTE IMMEDIATE 文の結果セットの最初のカラムのみが返されます。

プロシージャが結果セットを 1 つしか返さない場合、RESULT 句を使用してください。この句を使用すると、カーソルがオープンした後で ODBC と Open Client のアプリケーションが結果セットを記述し直すのを防ぐことができます。

複数の結果セットを処理するために ODBC は、プロシージャが定義した結果セットではなく、現在実行中のカーソルを記述します。したがって、ODBC はいつもプロシージャ定義の RESULT 句内で定義されているカラム名を記述するわけではありません。この問題を回避するには、結果セットを生成する SELECT 文でカラムエイリアスを使用します。

NO RESULT SET 句

NO RESULT SET 句を指定する場合は、文の最初の句として指定する必要があります。

このプロシージャによって結果セットが返されないことを宣言します。この句は、プロシージャが結果セットを返さないことを外部環境から知る必要がある場合に役立ちます。

SQL SECURITY 句

SQL SECURITY 句は、INVOKER (プロシージャを呼び出すユーザ) または DEFINER (プロシージャを所有するユーザ) としてプロシージャが実行されるかどうかを定義します。デフォルトは DEFINER です。

SQL SECURITY INVOKER が指定されている場合は、プロシージャを呼び出すユーザごとに注釈を行う必要があるためメモリ使用量が増えます。SQL SECURITY INVOKER が指定されている場合は、呼び出し側としても名前の決定が行われます。このため、必ずすべてのオブジェクト名 (テーブル、プロシージャなど) を該当する所有者で修飾してください。たとえば、user1 が次のプロシージャを作成するとします。

```
CREATE PROCEDURE user1.myProcedure()
    RESULT( columnA INT )
    SQL SECURITY INVOKER
BEGIN
    SELECT columnA FROM table1;
END;
```

user2 がこのプロシージャを実行しようとし、テーブル user2.table1 が存在しない場合、テーブルルックアップエラーが生じます。さらに、user2.table1 が存在する場合は、意図する user1.table1 の代わりにこのテーブルが使用されます。このような状況を防ぐには、文においてテーブル参照を修飾します (単なる table1 ではなく、user1.table1 とします)。

ON EXCEPTION RESUME 句

この句は、Transact-SQL のようなエラー処理を Watcom SQL 構文のプロシージャで使用可能にします。

ON EXCEPTION RESUME を使用すると、プロシージャは on_tsql_error オプションの設定に応じたアクションを実行します。on_tsql_error を Conditional (デフォルト) に設定すると、次の文がエラーを処理する場合は実行が継続され、そうでない場合は終了します。

エラー処理文には、次のようなものがあります。

- IF
- SELECT @variable =

- CASE
- LOOP
- LEAVE
- CONTINUE
- CALL
- EXECUTE
- SIGNAL
- RESIGNAL
- DECLARE
- SET VARIABLE

ON EXCEPTION RESUME では、明示的なエラー処理コードを使用しないでください。

この句は BEGIN...END 文の TRY ブロック内では無視されます。

compound-statement

BEGIN と END で囲まれ、セミコロンで区切られた SQL 文のセット。

AT 句

`location-string` に指定されたリモートプロシージャのプロキシストアドプロシージャを現在のデータベース上に作成します。AT 句では、`location-string` 内のフィールドデリミタとしてセミコロン (;) を使用できます。セミコロンがない場合は、ピリオドがフィールドデリミタになります。セミコロンを使用すると、データベースと所有者の各フィールドにファイル名と拡張子を使用できます。

AT 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます ({`variable-name`})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、`'bostonase.master.dbo.{@myprocedure}'` を含む AT 句は、`@myprocedure` が SQL 変数で、`@myprocedure` 変数の現在の内容がリモートプロシージャの使用時に置き換えられることを示します。

リモートプロシージャが結果セットを返すことができる場合は、たとえすべてのケースで結果セットを返せるわけではなくても、ローカルプロシージャ定義には RESULT 句を含めてください。

備考

CREATE PROCEDURE 文はデータベースにプロシージャを作成します。プロシージャは CALL 文で呼び出します。

外部プロシージャまたはネイティブプロシージャを呼び出す永続的なストアドプロシージャの作成には、さまざまなプログラミング言語を使用できます。

PROC は PROCEDURE の同義語として使用できます。

複数のプロシージャからテンポラリテーブルを参照する場合、テンポラリテーブル定義が矛盾していたり、テーブルを参照する文がキャッシュされていたりすると、問題が発生する可能性があります。

プロシージャの本体は複合文です。複合文は BEGIN で始まり、END で終わります。NewDepartment では、複合文は BEGIN 文と END 文に挟まれた 1 つの INSERT 文です。

プロシージャのパラメータは IN、OUT、または INOUT のいずれかです。デフォルトでは、パラメータは INOUT パラメータです。NewDepartment プロシージャのパラメータは、プロシージャによって変更されないため、すべてが IN パラメータです。パラメータを使用して呼び出し元に値を返さない場合は、パラメータを IN に設定してください。

権限

ユーザ本人が所有するプロシージャを作成するには、CREATE PROCEDURE システム権限が必要です。

他のユーザが所有するプロシージャを作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT の権限が必要です。

テンポラリプロシージャの作成には権限は必要ありません。

既存のプロシージャを置き換えるには、そのプロシージャを所有するか、または次のいずれかの権限が必要です。

- CREATE ANY PROCEDURE および DROP ANY PROCEDURE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY PROCEDURE システム権限。

関連する動作

オートコミット、テンポラリプロシージャにも適用。

標準

ANSI/ISO SQL 標準

CREATE PROCEDURE は ANSI/ISO SQL 標準のコア機能ですが、SQL Anywhere でサポートされている一部のコンポーネントはオプションの SQL 言語機能です。これらの機能のサブセットを次に示します。

- SQL SECURITY 句は、オプションの ANSI/ISO SQL 言語機能 T324 です。
- SQL プロシージャに LONG VARCHAR、LONG NVARCHAR、または LONG BINARY の値を渡す機能は、ANSI/ISO SQL 言語機能 T041 です。
- CREATE TABLE または DROP TRIGGER などの文を使用して SQL プロシージャ内でスキーマオブジェクトを作成または変更する機能は、ANSI/ISO SQL 言語機能 T651 です。
- EXECUTE IMMEDIATE、PREPARE、および DESCRIBE などの文を使用して SQL プロシージャ内で動的 SQL 文を使用する機能は、ANSI/ISO SQL 言語機能 T652 です。

CREATE PROCEDURE 文のいくつかの句は、標準にありません。これらを以下に示します。

- TEMPORARY 句。
- ON EXCEPTION RESUME 句。
- AT 句。
- 特定のルーチンパラメータ用のオプションの DEFAULT 句。
- RESULT 句と NO RESULT SET 句。ANSI/ISO SQL 標準では、RETURNS キーワードが使用されます。
- オプションの OR REPLACE 句。

Transact-SQL

CREATE PROCEDURE は、Adaptive Server Enterprise でサポートされています。

例

次のプロシージャは、Employees テーブルに問い合わせ、指定の給与 (sal) の指定の割合 (percentage) 内にある給与を返します。

```
CREATE OR REPLACE PROCEDURE AverageEmployees( IN percentage NUMERIC( 5,3), IN sal
NUMERIC( 20, 3 ) )
RESULT( Department CHAR(40), GivenName person_name_t, Surname person_name_t,
Salary NUMERIC( 20, 3) )
BEGIN
    DECLARE maxS NUMERIC( 20, 3 );
    DECLARE minS NUMERIC( 20, 3 );
    IF percentage >= 1 THEN
        SET percentage = percentage / 100;
    ELSEIF percentage < 0 THEN
        SELECT 'Percentage error', 'Err','Err', -1;
        RETURN;
    END IF;
    SELECT MIN( E.Salary ), MAX( E.Salary ) INTO minS, maxS
    FROM GROUPO.Employees E;
    IF sal < minS OR sal > maxS THEN
        SELECT 'Salary out of bounds', 'Err', 'Err', -2;
        RETURN;
    END IF;
    SELECT D.DepartmentName, E.GivenName, E.Surname, E.Salary
    FROM GROUPO.Employees E JOIN Departments D ON E.DepartmentID = D.DepartmentID
    WHERE E.Salary BETWEEN sal *( 1 - percentage ) AND sal * ( 1 + percentage );
END;
```

次のプロシージャは、CASE 文を使用してクエリの結果を分類します。

```
CREATE PROCEDURE ProductType (IN product_ID INT, OUT type CHAR(10))
BEGIN
    DECLARE prod_name CHAR(20);
    SELECT name INTO prod_name FROM GROUPO.Products
    WHERE ID = product_ID;
    CASE prod_name
    WHEN 'Tee Shirt' THEN
        SET type = 'Shirt'
    WHEN 'Sweatshirt' THEN
        SET type = 'Shirt'
    WHEN 'Baseball Cap' THEN
        SET type = 'Hat'
    WHEN 'Visor' THEN
        SET type = 'Hat'
    WHEN 'Shorts' THEN
        SET type = 'Shorts'
    ELSE
        SET type = 'UNKNOWN'
    END CASE;
END;
```

次の例は、前の例で作成された ProductType プロシージャを置き換えます。プロシージャが置き換えられた後、Tee Shirt と Sweatshirt のパラメータが更新されます。

```
CREATE OR REPLACE PROCEDURE ProductType (IN product_ID INT, OUT type CHAR(10))
BEGIN
    DECLARE prod_name CHAR(20);
    SELECT name INTO prod_name FROM GROUPO.Products
    WHERE ID = product_ID;
    CASE prod_name
    WHEN 'Tee Shirt' THEN
        SET type = 'T Shirt'
    WHEN 'Sweatshirt' THEN
```

```
        SET type = 'Long Sleeve Shirt'
    WHEN 'Baseball Cap' THEN
        SET type = 'Hat'
    WHEN 'Visor' THEN
        SET type = 'Hat'
    WHEN 'Shorts' THEN
        SET type = 'Shorts'
    ELSE
        SET type = 'UNKNOWN'
    END CASE;
END;
```

次のプロシージャはカーソルのロー上でカーソルとループを使用して、単一の値を返します。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION
    FOR SQLSTATE '02000';
    DECLARE curThisCust CURSOR FOR
        SELECT CompanyName,
             CAST(SUM(SalesOrderItems.Quantity *
             Products.UnitPrice) AS INTEGER) VALUE
        FROM GROUPO.Customers
        LEFT OUTER JOIN SalesOrders
        LEFT OUTER JOIN SalesOrderItems
        LEFT OUTER JOIN Products
        GROUP BY CompanyName;
    DECLARE ThisValue INT;
    DECLARE ThisCompany CHAR(35);
    SET TopValue = 0;
    OPEN curThisCust;
    CustomerLoop:
    LOOP
        FETCH NEXT curThisCust
        INTO ThisCompany, ThisValue;
        IF SQLSTATE = err_notfound THEN
            LEAVE CustomerLoop;
        END IF;
        IF ThisValue > TopValue THEN
            SET TopValue = ThisValue;
            SET TopCompany = ThisCompany;
            END IF;
    END LOOP CustomerLoop;
    CLOSE curThisCust;
END;
```

次の例は、SQL Anywhere のサンプルデータベースの Departments テーブルに対して INSERT を実行し、新しい部署を作成するプロシージャ NewDepartment を作成します。

```
CREATE PROCEDURE NewDepartment(
   IN id INT,
   IN name CHAR(35),
   IN head_id INT )
BEGIN
   INSERT
   INTO GROUPO.Departments ( DepartmentID,
       DepartmentName, DepartmentHeadID )
   VALUES ( id, name, head_id );
END;
```

次の文は、プロシージャ DepartmentsCloseToCustomerLocation を作成し、%TYPE 属性を使用して、その IN パラメータを Customers テーブル内の ID カラムのデータ型に設定します。

```
CREATE OR REPLACE PROCEDURE DepartmentsCloseToCustomerLocation( IN customer_ID
Customers.ID%TYPE )
  BEGIN
    DECLARE cust_rec Customers%ROWTYPE;
    SELECT City, State, Country
    INTO cust_rec.City, cust_rec.State, cust_rec.Country
    FROM Customers
    WHERE ID = customer_ID;
    SELECT Employees.Surname, Employees.GivenName, Departments.DepartmentName
    FROM Employees JOIN Departments
    ON Departments.DepartmentHeadID = Employees.EmployeeID
    WHERE Employees.City = cust_rec.City
    AND Employees.State = cust_rec.State
    AND Employees.Country = cust_rec.Country;
  END;
CALL DepartmentsCloseToCustomerLocation(158);
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
ALTER PROCEDURE 文 [669 ページ]
BEGIN 文 [745 ページ]
CALL 文 [756 ページ]
CREATE FUNCTION 文 [837 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
CREATE PROCEDURE 文 [T-SQL] [888 ページ]
CREATE SERVER 文 [914 ページ]
DROP PROCEDURE 文 [1051 ページ]
EXECUTE IMMEDIATE 文 [SP] [1090 ページ]
GRANT 文 [1131 ページ]
SQL データ型 [124 ページ]

1.4.4.81 CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]

パブリケーションを作成します。Mobile Link では、パブリケーションが SQL Anywhere リモートデータベース内の同期データを識別します。SQL Remote では、統合データベース内とリモートデータベース内の両方のレプリケートされたデータがパブリケーションによって識別されます。

構文

Mobile Link の一般的な使用方法

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner. ]  publication-name
( article-definition, ... )
```

```
article-definition :
  TABLE table-name [ ( column-name, ... ) ]
[ WHERE search-condition ]
```

Mobile Link のスクリプト化されたアップロード

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner. ] publication-name
WITH SCRIPTED UPLOAD
( article-definition, ... )
```

```
article-definition :
  TABLE table-name [ ( column-name, ... ) ]
[ USING ( [ PROCEDURE ] [ owner.] procedure-name
   FOR UPLOAD { INSERT | DELETE | UPDATE }, ... ) ]
```

Mobile Link のダウンロード専用パブリケーション

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner.] publication-name
FOR DOWNLOAD ONLY
( article-definition, ... )
```

```
article-definition :  TABLE table-name [ ( column-name, ... ) ]
```

SQL Remote

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner.] publication-name
( article-definition, ... )
```

```
article-definition :
  TABLE table-name [ ( column-name, ... ) ]
[ WHERE search-condition ]
[ SUBSCRIBE BY expression ]
```

パラメータ

IF NOT EXISTS 句

IF NOT EXISTS 句を指定し、指定したパブリケーションがすでに存在する場合、変更は行われず、エラーも返されません。

article-definition

パブリケーションは複数のアーティクルで構成されています。各アーティクルで、パブリケーションに含まれる単一のテーブルのローとカラムが識別されます。CONTAINS クエリ内のすべてのカラム参照は、同じテーブルを参照しなければなりません。

アーティクルにカラム名のリストが含まれている場合、それらのカラムがパブリケーションに含まれます。カラム名がリストに含まれない場合、テーブル内のすべてのカラムがパブリケーションに含まれます。Mobile Link の同期の場合、カラム名がリストされないと、テーブルのプライマリキーのすべてのカラムをリストに含める必要があります。

Mobile Link のスクリプト化されたアップロード構文では、スクリプト化されたアップロードを実行するパブリケーションに使用されます。また、アーティクルの記述によって、アップロードの定義に使用するスクリプトも登録します。

Mobile Link のダウンロード専用パブリケーションは、ダウンロードのみのパブリケーションに使用されます。また、アーティクルはダウンロードするテーブルとカラムのみを指定します。

WHERE 句

WHERE 句を使用すれば、テーブルローのサブセットをアーティクルに含めるように定義することができます。

Mobile Link アプリケーションでは、WHERE 句はアップロードに含まれているローに影響します (ダウンロードについては download_cursor スクリプトで定義されます)。Mobile Link の SQL Anywhere リモートデータベースでは、WHERE 句は、アーティクルに含まれているカラムのみを参照できます。また、サブクエリ、変数、または非決定的関数を含めることはできません。

SUBSCRIBE BY 句

SQL Remote の場合、アーティクルに含めるテーブルローのサブセットを定義するには、SUBSCRIBE BY 句を使用する方法があります。この句を使用すると単一のパブリケーション定義を使って複数のサブスクライバが 1 つのテーブルの別々のローを受信できます。

備考

CREATE PUBLICATION 文はデータベースにパブリケーションを作成します。所有者名を指定すれば、別のユーザのパブリケーションも作成できます。

Mobile Link では、パブリケーションは SQL Anywhere リモートデータベースでは必須で、Ultra Light データベースではオプションです。このようなパブリケーションとそれに対するサブスクリプションによって、Mobile Link サーバにアップロードされるデータが決定されます。

Mobile Link パブリケーションのオプションは、CREATE SYNCHRONIZATION SUBSCRIPTION 文または ALTER SYNCHRONIZATION SUBSCRIPTION 文の ADD OPTION 句で設定します。

Mobile Link のスクリプト化されたアップロード構文は、スクリプト化されたアップロードのパブリケーションを作成します。USING 句を使用して、アップロードの定義に使用するストアドプロシージャを登録します。各テーブルで、3 つまでのストアドプロシージャ (挿入、削除、更新用のプロシージャを 1 つずつ) を使用できます。

Mobile Link のダウンロード専用パブリケーション構文は、トランザクションログファイルなしで同期できるダウンロード専用パブリケーションを作成します。ダウンロード専用パブリケーションが同期されると、ダウンロードされたローは、リモートデータベースのローに加えられた変更を上書きします。

SQL Remote では、パブリッシュは双方向オペレーションなので、データは統合データベースとリモートデータベースのどちらへでも入力できます。SQL Remote インストール環境では、統合データベースとすべてのリモートデータベースのパブリケーション定義は同じにします。統合データベースで SQL Remote 抽出ユーティリティを実行すると、リモートデータベースで自動的に正しい CREATE PUBLICATION 文が実行されます。

すべての構文では、その文を実行するために文内で参照されるすべてのテーブルに対して、排他的にアクセスできる権限が必要です。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、2 つのテーブルのすべてのカラムとローをパブリッシュします。

```
CREATE PUBLICATION pub_contact (
   TABLE GROUPO.Contacts,
   TABLE GROUPO.Customers
);
```

次の文は、1 つのテーブルの一部のカラムのみをパブリッシュします。

```
CREATE PUBLICATION pub_customer (
   TABLE GROUPO.Customers ( ID, CompanyName, City )
);
```

次の文は、Customers テーブルの Status カラムを検証する WHERE 句を組み込んで、ニューヨーク (NY) に在住する顧客のローのみをパブリッシュします。

```
CREATE PUBLICATION pub_customer (
   TABLE GROUPO.Customers ( ID, CompanyName, City, State, Status )
   WHERE State = 'NY'
);
```

次の文は、subscribe-by 値を与えて一部のローのみをパブリッシュします。この方法を使用できるのは、SQL Remote の場合のみです。

```
CREATE PUBLICATION pub_customer (
   TABLE GROUPO.Customers ( ID, CompanyName, City, State )
   SUBSCRIBE BY State
);
```

subscribe-by 値は、SQL Remote サブスクリプションの作成時に次のように使用されます。

```
CREATE SUBSCRIPTION TO pub_customer ( 'NY' )
   FOR jsmith;
```

次の例は、スクリプト化されたアップロードを使用する Mobile Link パブリケーションを作成します。

```
CREATE PUBLICATION pub WITH SCRIPTED UPLOAD (
      GROUPO.TABLE t1 (a, b, c) USING (
         PROCEDURE my.t1_ui FOR UPLOAD INSERT,
         PROCEDURE my.t1_ud FOR UPLOAD DELETE,
         PROCEDURE my.t1_uu FOR UPLOAD UPDATE
      ),
      GROUPO.TABLE t2 AS my_t2 USING (
         PROCEDURE my.t2_ui FOR UPLOAD INSERT
      )
   );
```

次の例は、ダウンロードのみのパブリケーションを作成します。

```
CREATE PUBLICATION p1 FOR DOWNLOAD ONLY (
   GROUPO.TABLE t1
);
```

関連情報

ALTER PUBLICATION 文 [Mobile Link] [SQL Remote] [672 ページ]
DROP PUBLICATION 文 [Mobile Link] [SQL Remote] [1052 ページ]
CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [948 ページ]
ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [701 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.82 CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote]

データベースからの出力メッセージのメッセージリンクとリターンアドレスを識別します。

構文

```
CREATE REMOTE [MESSAGE] TYPE message-system
[ ADDRESS address-string ]
```

```
message-system :
FILE
| FTP
| HTTP
| SMTP
```

パラメータ

message-system

SQL Remote がサポートするメッセージシステムの 1 つ。次のいずれかの値を指定します。FILE、FTP、または SMTP。

address-string

指定したメッセージシステムに対して有効なアドレスを含む文字列。

備考

Message Agent は、サポートされたメッセージリンクのうちの 1 つを使用して、データベースから出力メッセージを送信します。リモートデータベースが抽出ユーティリティで作成されていれば、指定したリンクを使用するユーザのリターンメッセージは、指定したアドレスに送られます。Message Agent は、リンク用のリモートユーザがある場合にのみ、それらのリンクを開始します。

アドレスには、指定したメッセージシステムに応じた、パブリッシャのアドレスを指定します。電子メールシステムの場合、アドレス文字列には有効な電子メールアドレスを指定します。ファイル共有システムの場合、アドレス文字列には SQLREMOTE 環境変数で設定されているディレクトリのサブフォルダを指定します。SQLREMOTE 環境変数でディレクトリが指定されていない場合は、現在のディレクトリを指定してください。この設定は、リモートデータベースで GRANT CONSOLIDATE 文を使って無効にできます。

アドレスを削除するには、CREATE REMOTE MESSAGE TYPE 文を ADDRESS 句なしで実行します。

dbinit ユーティリティは、アドレスを除くメッセージタイプを自動的に作成します。CREATE REMOTE MESSAGE TYPE 文は、他の CREATE 文と異なり指定したメッセージタイプがすでに存在してもエラーを返さずに既存のタイプを変更します。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

抽出ユーティリティを使用してリモートデータベースを抽出した場合は次の文を使用するとファイルメッセージシステムのメッセージ受信者すべてがメッセージを `company` サブディレクトリに送り返すように設定できます。

また、dbremote に対して、company サブディレクトリで入力メッセージを確認するように指定します。

```
CREATE REMOTE MESSAGE TYPE file
ADDRESS 'company';
```

関連情報

GRANT PUBLISH 文 [SQL Remote] [1147 ページ]
GRANT REMOTE 文 [SQL Remote] [1149 ページ]
GRANT CONSOLIDATE 文 [SQL Remote] [1141 ページ]
DROP REMOTE MESSAGE TYPE 文 [SQL Remote] [1053 ページ]
ALTER REMOTE MESSAGE TYPE 文 [SQL Remote] [674 ページ]

1.4.4.83 CREATE ROLE 文

ロールの作成または置換、ユーザ拡張ロールの作成、またはロール管理者の変更を行います。

構文

```
CREATE [ OR REPLACE ] ROLE { role-name | FOR USER userid }
[ WITH ADMIN [ ONLY ] administrator-userid [,...] ]
```

パラメータ

OR REPLACE clause

この句を使用して、ロールがまだ存在しない場合はロールを作成し、ロールが存在する場合はその管理者を置き換えます。

role-name

このパラメータを使用して、ロールの名前を指定します。この名前は、データベース内のすべてのユーザおよびロールでユニークでなければなりません。

FOR USER userid clause

この句を使用して、指定されたユーザを、他のユーザに割り当て可能なユーザ拡張ロールに変換します。ユーザはすでに別のロールとして拡張されていてはなりません。

WITH ADMIN and WITH ADMIN ONLY administrator-userid clause

ロールの管理者を指定します (オプション)。WITH ADMIN は、administrator-userid がロールを行使したり管理したりできることを意味します。WITH ADMIN ONLY は、administrator-userid がロールの管理のみ行えることを意味します。句が指定されていない場合は、MANAGE ROLES システム権限を持つユーザなら誰でもロールを管理できます。

min_role_admins データベースオプションは、各ロールに必要な管理者の最小数を制御します。ロールを作成するときに十分な数の管理者を指定しない場合、この文はエラーを返します。

備考

新しいロールの名前は、'SYS_' で始まったり、'_ROLE' で終わったりしてはなりません。たとえば、SYS_MyBackup_ROLE はユーザ定義ロールの名前としては使用できず、MyBackup_ROLE や SYS_MyBackup は使用できます。

ADMIN 句が指定されている場合は、指定されたユーザだけがロールを管理できます。ADMIN 句が指定されていない場合は、デフォルトで管理権限のみを有するロールが MANAGE ROLES システム権限に付与されます。これはグローバル管理者がロールを管理できることを意味します。

ユーザ拡張ロールを作成する (つまりユーザをロールに拡張する) 場合は、`CREATE ROLE FOR USER userid` 構文を使用します。

ロールにシステム権限を付与するには、GRANT 文を使用します。

権限

新しいロールを作成するには、MANAGE ROLES システム権限が必要です。

OR REPLACE 句が指定され、ロールがすでに存在する場合は、そのロールに対する管理権限も必要になります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は Sales ロールを作成します。このロールは、MANAGE ROLES システム権限を持つユーザなら誰でも管理できます。

```
CREATE ROLE Sales;
```

次の文は、ユーザ JaneSmith を拡張して、別のユーザに割り当てることができるロールにします。

```
CREATE ROLE FOR USER JaneSmith;
```

次の文は、ロールに対する管理権限 (のみ) を有するロール管理者として MaryJones と JeffTurkott を指定して、ロール Finance を作成します。

```
CREATE ROLE Finance
WITH ADMIN ONLY MaryJones, JeffTurkott;
```

次の例は、前の例で作成された既存の Finance ロールを置き換えて、ロール管理者として MaryJones と JeffTurkott を EllenChong と DaveLexx に置き換えます (今回はロールの行使権限を使用)。

```
CREATE OR REPLACE ROLE Finance
WITH ADMIN EllenChong, DaveLexx;
```

関連情報

DROP ROLE 文 [1056 ページ]
ALTER ROLE 文 [675 ページ]

1.4.4.84 CREATE SCHEMA 文

データベースユーザのテーブルとビューのコレクションを作成します。

構文

```
CREATE SCHEMA
 AUTHORIZATION userid
[  create-table-statement
   | create-view-statement
   | grant-statement
]  ... ;
```

備考

CREATE SCHEMA 文はスキーマを作成します。スキーマは、テーブルやビューと、それぞれに関連する権限のコレクションです。

userid には、現在の接続のユーザ ID を指定する必要があります。別のユーザに対するスキーマを作成することはできません。

CREATE SCHEMA 文に含まれるいずれかの文にエラーが発生すると、CREATE SCHEMA 文全体がロールバックされます。

CREATE SCHEMA 文を使用すると、個別の CREATE と GRANT 文を 1 つにまとめて一度に処理することができます。データベース内に SCHEMA データベースオブジェクトは作成されません。オブジェクトを削除するには、個別の DROP TABLE ま

たは DROP VIEW 文を使用します。権限を取り消すには、付与されているそれぞれの権限に対して REVOKE 文を使用します。

個々の CREATE または GRANT 文は、文デリミタで区切りません。文デリミタは CREATE SCHEMA 文自身の末尾を区切ります。

それぞれの CREATE または GRANT 文は、まずオブジェクトを作成してから、それに権限を付与するという順番に並べます。

ユーザに複数のスキーマを作成することはできますが、お奨めしません。

権限

必要なシステム権限は、定義する CREATE SCHEMA 文で指定される処理によって異なります。必要なシステム権限の詳細については、該当する文 (CREATE TABLE、CREATE VIEW、GRANT) のシステム権限の項を参照してください。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

コア機能。単一ユーザに対して複数スキーマを作成する機能は、オプションの SQL 言語機能 F171 です。ソフトウェアでは、CREATE SCHEMA 文内での REVOKE 文の使用をサポートされていません。また、Transact-SQL バッチまたはプロシージャ内での使用も許可されていません。

Transact-SQL

CREATE SCHEMA 文で GRANT 文と REVOKE 文をサポートする Adaptive Server Enterprise でサポートされています。

例

次の CREATE SCHEMA 文は、2 つのテーブルがあるスキーマを作成します。この文は、CREATE TABLE システム権限を持つユーザ ID sample_user で実行します。テーブル t2 を作成する文が失敗すると、どちらのテーブルも作成されません。

```
CREATE SCHEMA AUTHORIZATION sample_user
CREATE TABLE t1 ( id1 INT PRIMARY KEY )
CREATE TABLE t2 ( id2 INT PRIMARY KEY );
```

次の CREATE SCHEMA 文の文デリミタは、最初の CREATE TABLE 文の後に配置されます。文デリミタは CREATE SCHEMA 文の終わりをマークするので、例は、データベースサーバによって 2 つの文バッチとして解釈されます。テーブル t2 を作成する文が失敗しても、テーブル t1 は作成されます。

```
CREATE SCHEMA AUTHORIZATION sample_user
```

```
CREATE TABLE t1 ( id1 INT PRIMARY KEY );
CREATE TABLE t2 ( id2 INT PRIMARY KEY );
```

関連情報

CREATE TABLE 文 [952 ページ]
CREATE VIEW 文 [995 ページ]
GRANT 文 [1131 ページ]

1.4.4.85 CREATE SEMAPHORE 文

セマフォを作成または置換し、カウンタの初期値を設定します。セマフォとはロックメカニズムであり、カウンタを使用して、外部ライブラリやプロシージャなどのリソースの可用性を伝達および制御します。

構文

```
CREATE [ OR REPLACE | TEMPORARY ] SEMAPHORE [ IF NOT EXISTS ] [ owner.]semaphore-name
[ START WITH initial-count ]
```

パラメータ

owner

セマフォの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

semaphore-name

セマフォの名前。CHAR データベース照合に有効な識別子を指定します。semaphore-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

OR REPLACE clause

この句は、同じ名前の永続セマフォの定義 (存在する場合) を上書き (更新) するときに使用します。

OR REPLACE 句を指定し、その時点でこの名前のセマフォが使用中の場合、文からエラーが返されます。

この句は、TEMPORARY 句または IF NOT EXISTS 句と一緒に使用しないでください。

TEMPORARY clause

この句は、一時セマフォを作成するときに使用します。

この句は、OR REPLACE 句と一緒に使用しないでください。

IF NOT EXISTS clause

この句は、存在しないセマフォを作成するときに使用します。同じ名前と存続期間 (永久または一時) を持つセマフォが存在する場合は、何も実行されずエラーも返されません。

この句は、OR REPLACE 句と一緒に使用しないでください。

START WITH clause

この句は、セマフォカウンタの初期値を指定するときに使用します。この句を指定しない場合、`initial-count` は 1 に設定されます。

`initial-count` は、変数 (たとえば、**`START WITH @initial-count`**) を使用して指定できます。

`initial-count` に NULL を設定するか、変数を設定し、その変数が NULL の場合、句を指定しないのと同じ動作になります。

備考

CREATE SEMAPHORE 文は、セマフォを作成し、カウンタを設定します。NOTIFY SEMAPHORE 文が実行されるたびに、関連するセマフォのカウンタが増分されます。現在のカウントが正の整数であると仮定して、WAITFOR SEMAPHORE 文が実行されるたびに、関連するセマフォのカウンタが減分されます。

永続および一時のミューテックスおよびセマフォは、同じネームスペースを共有するため、これらのオブジェクトは同じ名前で作成できません。OR REPLACE および IF NOT EXISTS 句を使用すると、命名に関する不適切なエラーが発生することがあります。たとえば、永続ミューテックスが存在していて、それと同じ名前の一時セマフォを作成しようとすると、IF NOT EXISTS を指定してもエラーが返されます。同様に、一時セマフォが存在していて、OR REPLACE を指定して同じ名前の永続セマフォで置き換えようとすると、エラーが返されます。これは、同じ名前で 2 つめのオブジェクトを作成しようとするのと同じためです。

永続セマフォの定義は、データベースの再起動後も保持されます。ただし、再起動後はカウントが `initial-count` に戻ります。

一時セマフォは、セマフォを作成した接続が終了するまで、または DROP 操作が明示的に実行されるまで存続します。別の接続が一時セマフォを待機中で、一時セマフォを作成した接続が終了した場合は、待機中の接続にエラーが返されます。

永続セマフォを置き換えると (OR REPLACE 句)、古いセマフォが削除され、このセマフォを待機しているすべての接続に通知されます。

OR REPLACE 句を指定した場合、その名前の永続セマフォが存在し、そのセマフォを待機している接続がブロックされていても、セマフォは置き換えられます。この場合、待機中の接続はブロック解除され、セマフォが削除されたことを示すエラーが返されます。ただし、これには例外があります。置き換えるセマフォの定義に同じ設定が使用されている場合は、待機中の接続には何も影響がありません。

権限

CREATE ANY MUTEX SEMAPHORE または CREATE ANY OBJECT のシステム権限が必要です。

関連する動作

オートコミット (永続セマフォの場合のみ)。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、license_counter というセマフォを作成し、カウンタに 3 を設定します。

```
CREATE SEMAPHORE license_counter START WITH 3;
```

関連情報

DROP SEMAPHORE 文 [1059 ページ]
NOTIFY SEMAPHORE 文 [1215 ページ]
WAITFOR SEMAPHORE 文 [1405 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.86 CREATE SEQUENCE 文

複数のテーブル間でユニークな値となるプライマリキー値、およびテーブルのデフォルト値を生成するために使用できるシーケンスを作成します。

構文

```
CREATE [ OR REPLACE ] SEQUENCE [ owner.] sequence-name
[ INCREMENT BY signed-integer ]
[ START WITH signed-integer ]
[ MINVALUE signed-integer | NO MINVALUE ]
[ MAXVALUE signed-integer | NO MAXVALUE ]
[ CACHE integer | NO CACHE ]
[ CYCLE | NO CYCLE ]
```

パラメータ

OR REPLACE 句

OR REPLACE を指定すると、新しいシーケンスが作成されるか、同じ名前の既存のシーケンスが置き換えられます。OR REPLACE 句を使用しなかった場合、現在のユーザに対してすでに存在するシーケンスの名前を指定すると、エラーが返されます。

INCREMENT BY 句

最後に割り当てられた値から次のシーケンス値までの増分量を定義します。デフォルトは 1 です。負の値を指定すると、降順のシーケンスが生成されます。INCREMENT BY 値が 0 の場合は、エラーが返されます。

START WITH 句

シーケンスの開始値を定義します。START WITH 句に値を指定しない場合、昇順シーケンスでは MINVALUE が使用され、降順シーケンスでは MAXVALUE が使用されます。START WITH の値が MINVALUE または MAXVALUE で指定された範囲外の場合は、エラーが返されます。

MINVALUE 句

シーケンスで生成される最小値を定義します。デフォルトは 1 です。INVALUE が $(2^{\wedge 63}-1)$ より大きいか $-(2^{\wedge 63}-1)$ より小さい場合は、エラーが返されます。また、MINVALUE が MAXVALUE より大きい場合も、エラーが返されます。

MAXVALUE 句

シーケンスで生成される最大値を定義します。デフォルトは $(2^{\wedge 63}-1)$ です。MAXVALUE が $(2^{\wedge 63}-1)$ より大きいか $-(2^{\wedge 63}-1)$ より小さい場合は、エラーが返されます。

CACHE 句

より速くアクセスできるようにメモリに保持される、事前に割り付けられたシーケンス値の数を指定します。キャッシュが不足すると、シーケンスキャッシュが再移植され、対応するエントリがトランザクションログに書き込まれます。キャッシュの現在の値は、チェックポイントの時点で ISYSSEQUENCE システムテーブルに転送されます。デフォルトは 100 です。

CYCLE 句

最大値または最小値に達した後に、値の生成を継続するかどうかを指定します。

デフォルトは NO CYCLE です。この場合、最大値または最小値に達すると、エラーが返されます。

備考

シーケンスは、数値の自動生成が可能なデータベースオブジェクトです。シーケンスは特定またはユニークなテーブルカラムにバインドされません。

シーケンスでは、次のいずれかの方法で値を生成できます。

- 単調な増加または減少を無制限に行う
- ユーザ定義の制限まで単調な増加または減少を行い、停止する
- ユーザ定義の制限まで単調な増加または減少を行い、循環して始めに返り、再び開始する

CYCLE 句を使用して、シーケンスで値を使い果たした場合の動作を制御します。

シーケンスが増加して MAXVALUE を上回り、CYCLE が指定されている場合には、次のシーケンス値として MINVALUE が使用されます。シーケンスが減少して MINVALUE を下回り、CYCLE が指定されている場合には、次のシーケンス値として MAXVALUE が使用されます。CYCLE が指定されていない場合は、エラーが返されます。

ビュー定義またはマテリアライズドビュー定義では、シーケンス値を使用できません。

権限

シーケンスを作成するには、CREATE ANY SEQUENCE または CREATE ANY OBJECT システム権限が必要です。

既存のシーケンスを置き換えるには、次のいずれかのシステム権限が必要です。

- CREATE ANY SEQUENCE および DROP ANY SEQUENCE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY SEQUENCE システム権限。

関連する動作

なし

標準

ANSI/ISO SQL 標準

シーケンスは、SQL 言語機能 T176 を構成します。ソフトウェアでは、シーケンスのデータ型にオプションの指定はできません。これを行うには、シーケンスで CAST を使用します。

さらに、次のものは標準にありません。

- CACHE 句
- OR REPLACE 構文
- CURRVAL 式
- DEFAULT 式でのシーケンスの使用

例

次の例は、4 で開始して 2 ずつ増加し、循環せず、15 個の値を一度にキャッシュする、Test という名前のシーケンスを作成します。

```
CREATE SEQUENCE Test
START WITH 4
INCREMENT BY 2
NO MAXVALUE
NO CYCLE
CACHE 15;
```

関連情報

ALTER SEQUENCE 文 [677 ページ]
DROP SEQUENCE 文 [1060 ページ]

1.4.4.87 CREATE SERVER 文

リモートサーバまたはディレクトリアクセスサーバの作成

構文

リモートサーバの作成

```
CREATE [ REMOTE ] SERVER server-name
CLASS server-class-string | variable
USING connection-info-string | variable
[ READ ONLY [ ON | OFF | VALUE variable ] ]
[ DEFAULT LOGIN string | variable [ IDENTIFIED BY string | variable ] ]
```

```
server-class-string :
{ 'ADSODBC' | 'ADS_ODBC'
| 'ASEODBC' | 'ASE_ODBC'
| 'DB2ODBC' | 'DB2_ODBC'
| 'HANAODBC' | 'HANA_ODBC'
| 'IQODBC' | 'IQ_ODBC'
| 'MIRROR'
| 'MSACCESSODBC' | 'MSACCESS_ODBC'
| 'MSSODBC' | 'MSS_ODBC'
| 'MYSQLODBC' | 'MYSQL_ODBC'
| 'ODBC'
| 'ORAODBC' | 'ORA_ODBC'
| 'SAODBC' | 'SA_ODBC'
| 'ULODBC' | 'UL_ODBC' }
```

```
connection-info-string :
  { 'data-source-name' | 'sqlanywhere-connection-string' }
```

ディレクトリアクセスサーバの作成

```
CREATE SERVER server-name
CLASS 'DIRECTORY'
USING using-string | variable
[ READ ONLY [ ON | OFF | VALUE variable ] ]
[ ALLOW { 'ALL' | 'SPECIFIC' | variable } USERS ]
```

```
using-string :
  'ROOT = path [ ;SUBDIRS = n ] [ ;CREATEDIRS = { YES | NO } ] [ ;DELIMITER = { / 
| ¥ } ]'
```

リモートサーバの作成 (SAP HANA 構文)

```
CREATE REMOTE SOURCE remote-source-name
   ADAPTER adapter-name | variable
   CONFIGURATION connection-info-string | variable
   [ READ ONLY [ ON | OFF | VALUE variable ] ]
```

```
    [ WITH CREDENTIAL TYPE { 'PASSWORD' | variable } USING { 'USER=remote-
user;password=remote-password' | variable ]
```

パラメータ

CREATE [REMOTE] SERVER

リモートサーバ用の REMOTE キーワードはオプションで、他のデータベースとの互換性を保つために用意されています。

CLASS 句

リモート接続に使用するサーバクラスを指定します。サーバクラスには、詳細なサーバ機能情報が入っています。

DIRECTORY クラスは、ローカルコンピュータのディレクトリにアクセスするディレクトリアクセスサーバを作成するために使用されます。

サーバのクラスは次のとおりです。

SAODBC

SQL Anywhere の場合。

ULODBC

Ultra Light

> **注記**
>
> Mac OS X 上で実行している Ultra Light データベースには、リモートサーバを作成できません。

ADSODBC

SAP Advantage Database Server の場合。

ASEODBC

SAP Adaptive Server Enterprise (バージョン 10 以降)

DB2ODBC

IBM DB2

HANAODBC

SAP HANA

IQODBC

SAP IQ

MSACCESSODBC

Microsoft Access

MSSODBC

Microsoft SQL Server

MYSQLODBC

Oracle MySQL

ODBC

その他の ODBC データソース
ORAODBC

Oracle Database サーバ (バージョン 8.0 以降)

> **注記**
>
> リモートデータアクセスを使用する際に、Unicode をサポートしていない ODBC ドライバを使用すると、その ODBC ドライバから受け取るデータに対して、文字セット変換が実行されません。

READ ONLY 句 (リモートサーバ)

リモートサーバが読み込み専用モードでアクセスされることを指定します。この句を指定しない場合、または READ ONLY OFF を指定する場合は、リモートサーバは読み込み専用モードでアクセスされません。READ ONLY または READ ONLY ON を指定すると、リモートサーバが読み込み専用モードでアクセスされます。

READ ONLY 句 (ディレクトリアクセスサーバ)

ディレクトリからアクセスするファイルが読み込み専用で修正できないかどうかを指定します。デフォルトでは、READ ONLY は NO に設定されています。

ALLOW USERS 句 (ディレクトリアクセスサーバ)

外部ログインのユーザに対してディレクトリアクセスサーバへのアクセスを制限するには、ALLOW 'SPECIFIC' USERS を指定します。アクセスが必要なユーザごとに、ディレクトリアクセスサーバへの外部ログインを明示的に作成する必要があります。この句はデフォルトです。

ディレクトリアクセスサーバにアクセスするユーザにこだわらないか、データベース内の全員にアクセス権を付与する場合は、ALLOW 'ALL' USERS 句を指定します。この句は、すべてのユーザが利用可能なディレクトリアクセスサーバへのデフォルトの外部ログインを作成します。

USING 句 (リモートサーバ)

リモートサーバの作成時に、USING 句によってデータベースサーバに接続文字列が提示されます。適切な接続文字列は使用されるドライバによって決まり、ドライバは指定した値によって決まります。

USING 句は、ODBC データソース名を示す `'DSN=data-source-name'` か UNIX のドライババイナリまたは Windows のドライバ名を示す `'DRIVER=driver-name'` またはその両方を含むことができる ODBC 接続文字列です。

SQL Anywhere リモートサーバ (SAODBC サーバクラス) の場合、`connection-info-string` パラメータには任意の有効な接続文字列を指定できます。サポートされている任意の接続パラメータを使用します。たとえば、接続に問題がある場合は、LOG 接続パラメータを含めて接続試行をトラブルシューティングします。

USING 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます ({`variable-name`})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、`'DSN={@mydsn}'` を含む USING 句は、`@mydsn` が SQL 変数であり、リモートデータアクセスサーバへの接続時に `@mydsn` 変数の現在の内容が置き換えられることを示します。

USING 句 (ディレクトリアクセスサーバ)

サーバ接続情報を設定します。

ディレクトリアクセスサーバの作成時に、USING 句によってローカルディレクトリに次の値が指定いされます。

ROOT 句

ディレクトリアクセスクラスのルートとなる、データベースサーバへの相対パスを指定します。ディレクトリアクセスサーバ名を使用してプロキシテーブルを作成すると、プロキシテーブルのパスはこのルートパスに対する相対パスになります。

SUBDIRS 句

データベースサーバがアクセス可能な、ルート内のディレクトリレベルの数を表す 0 ～ 10 の数を指定します。SUBDIRS を省略したときまたは 0 に設定したときは、ディレクトリアクセスサーバ経由でルートディレクトリのファイルのみにアクセスできます。ディレクトリアクセスサーバ経由で使用できるディレクトリまたはサブフォルダのいずれかにプロキシテーブルを作成できます。

CREATEDIRS 句

ディレクトリアクセスサーバを使用してディレクトリを作成できるかどうかを指定します。デフォルトは NO です。

DELIMITER 句

パスが、スラッシュ (/) 文字で区切られるのか、バックスラッシュ (¥) 文字で区切られるのかを指定します。デフォルトでは、ネイティブのパスのデリミタが使用されます。

USING 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます ({`variable-name`})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、`'ROOT={@mypath}'` を含む USING 句は、`@mypath` が SQL 変数であり、ディレクトリアクセスサーバへの接続の確立時に `@mypath` 変数の現在の内容が置き換えられることを示します。

DEFAULT LOGIN 句 (リモートサーバ)

デフォルトログインで使用されるリモートサーバのアカウントの、デフォルトのユーザ ID とパスワード (オプション) を指定します。DEFAULT LOGIN 句の値は 128 バイトまでに制限されます。

IDENTIFIED BY 句

IDENTIFIED BY 句は、リモートパスワードがリモートユーザのパスワードになるように指定します。この値には文字列または変数のいずれかを使用できます。リモートユーザとリモートパスワードの組み合わせはリモートサーバで有効でなければなりません。

IDENTIFIED BY 句を省略すると、NULL のパスワードがリモートサーバに送信されます。ただし、IDENTIFIED BY "" (空の文字列) を指定すると、空の文字列がパスワードとして送信されます。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

CREATE REMOTE SOURCE (SAP HANA 構文)

パラメータとその値を含むこの構文は、リモートサーバを作成する構文 (CREATE [REMOTE]) とセマンティック上同じで、SAP HANA サーバとの互換性を保つために用意されています。2 つの構文は、次のように 1 対 1 で句が一致します。

ADAPTER adapter-name

CREATE [REMOTE] SERVER 構文については、CLASS 句の説明を参照してください。

CONFIGURATION connection-info-string

CREATE [REMOTE] SERVER 構文については、USING 句の説明を参照してください。

READ ONLY [ON | OFF] 句

CREATE [REMOTE] SERVER 構文 (リモートサーバ) については、READ ONLY 句の説明を参照してください。

WITH CREDENTIAL TYPE 句

CREATE [REMOTE] SERVER 構文については、DEFAULT LOGIN 句の説明を参照してください。

備考 (リモートサーバ)

CREATE SERVER 文を使用して、別のデータソースのデータにアクセスするリモートサーバを作成します。リモートサーバを作成したら、そのリモートテーブルにマップするローカルプロキシテーブルを作成する必要があります。データベースユーザは、プロキシテーブルを使用して、リモートテーブルの内容にアクセスします。リモートサーバと通信する必要があるデータベースユーザごとに、外部ログインを作成します。

まず、現在の既存ユーザの外部ログインを使用して、リモートサーバに接続しようとします。このユーザに外部ログインがない場合、DEFAULT LOGIN 認証を使用して接続しようとします。リモートサーバの作成時に DEFAULT LOGIN が設定されておらず、ユーザに外部ログインが定義されていない場合、現在の既存ユーザの ID とパスワードを使用して接続が試みられます。

リモートサーバへの接続に関与するプロシージャを実行する際にに、リモートデータアクセス接続の実行に、ログインしたユーザの外部ログインクレデンシャルを使用するか、有効なユーザの外部ログインクレデンシャルを使用するかどうかを extern_login_credentials オプションで指定できます。

UNIX では、データベースサーバは特定のユーザ権限で実行されているため、ファイルパーミッションはデータベースサーバユーザに付与されている権限に基づきます。

リモートサーバからデータにアクセスする際に、Unicode をサポートしていない ODBC ドライバを使用すると、その ODBC ドライバから受け取るデータに対して、文字セット変換が実行されません。

リモートサーバを定義すると、ISYSSERVER システムテーブルにエントリが追加されます。SYSSERVER システムビューを問い合わせることによって、リモートサーバのリストを表示します。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合は、データベースサーバからエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

備考 (ディレクトリアクセスサーバ)

CREATE SERVER 文を使用して、データベースサーバが稼働しているコンピュータのローカルディレクトリ構造にアクセスするディレクトリアクセスサーバを作成します。ディレクトリアクセスサーバを使用する必要があるデータベースユーザごとに、外部ログインを作成します。ディレクトリアクセスサーバを作成したら、そのプロキシテーブルを作成する必要があります。データベースユーザは、プロキシテーブルを使用して、データベースサーバのローカルファイルシステムにあるディレクトリの内容にアクセスします。

SYSSERVER システムビューを問い合わせることによって、ディレクトリアクセスサーバのリストを表示します。

権限

SERVER OPERATOR システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、SQL Anywhere ODBC ドライバを使用して、RemoteSA という名前の SQL Anywhere リモートサーバを作成します。

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;DSN=RemoteDS';
```

次の例は、ODBC ドライバマネージャを使用しないで、SQL Anywhere ODBC ドライバを直接ロードします。

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere Native;DSN=RemoteDS';
```

次の例は、変数のリファレンスを使用して、動的なリモートデータアクセスサーバを作成します。この例を実行するには、MANAGE ANY USER および CREATE TABLE システム権限が必要です。

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;DSN={dsn_string};Server=saremote;UID=DBA;PWD=sql';
CREATE VARIABLE dsn_string LONG VARCHAR;
SET dsn_string = 'Test17';
CREATE EXTERNLOGIN DBA TO RemoteSA;
CREATE EXISTING TABLE test_employees
AT 'RemoteSA...Employees';
SELECT * FROM test_employees;
DROP REMOTE CONNECTION TO RemoteSA CLOSE ALL;
```

次の例は、ASE ODBC ドライバを使用して、ase_prod という名前の Adaptive Server Enterprise (ASE) リモートサーバを作成します。

```
CREATE SERVER ase_prod
CLASS 'ASEODBC'
USING 'DSN=remoteASE';
```

次の例は、Oracle サーバ oracle723 のリモートサーバを作成します。ODBC データソース名は oracle723 です。

```
CREATE SERVER oracle723
CLASS 'ORAODBC'
USING 'oracle723';
```

次の例は、ディレクトリ `c:¥temp` 内のファイルのみを表示するディレクトリアクセスサーバを作成します。

```
CREATE SERVER diskserver0
CLASS 'DIRECTORY'
USING 'ROOT=c:¥¥temp';
CREATE EXTERNLOGIN DBA TO diskserver0;
CREATE EXISTING TABLE diskdir0 AT 'diskserver0;;;.';
-- Get a list of those files.
SELECT privileges, file_name, size FROM diskdir0;
```

次の例は、2 つの異なるディレクトリを探索するために使用される、動的なディレクトリアクセスサーバを作成します。

```
CREATE SERVER diskserver9
CLASS 'DIRECTORY'
USING '{dir_options}';
CREATE EXTERNLOGIN DBA TO diskserver9;
CREATE EXISTING TABLE diskdir9 AT 'diskserver9;;;.';
CREATE VARIABLE dir_options VARCHAR(256);
SET dir_options = 'ROOT=c:¥¥temp;SUBDIRS=9;DELIMITER=/';
SELECT * FROM diskdir9;
DROP REMOTE CONNECTION TO diskserver9 CLOSE ALL;
SET dir_options = 'ROOT=c:¥¥ProgramData;SUBDIRS=9;DELIMITER=/';
SELECT * FROM diskdir9;
```

リモートサーバの作成時に ODBC ドライバマネージャをバイパスするには、次の構文を使用します。この後には、`connection-info-string` の残りの部分を続けます。

```
CREATE SERVER remote-server
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere Native;DSN=my-dsn;UID=my-username;PWD=my-pwd';
```

この構文を使用すると、リモートデータアクセスで SQL Anywhere ODBC ドライバを直接ロードできます。この構文は Windows と UNIX でサポートされます。SQL Anywhere ODBC ドライバを直接ロードすると、現在のサーバのバージョンの ODBC ドライバが使用されるようになります。また、SQL Anywhere ODBC ドライバをリモートデータアクセスにのみ使用する場合は、登録する必要はありません。

UNIX プラットフォームでは、SQL Anywhere ODBC ドライバも参照できます。構文は次のとおりです。

```
USING 'DRIVER=SQL Anywhere 17;DSN=my-dsn'
```

注記

アプリケーションが SQL Anywhere 以外のリモートサーバも使用している場合、または `'DRIVER=SQL Anywhere Native'` を使用しないで定義されている SQL Anywhere リモートサーバがある場合、リモートデータアクセスは他のリモートサーバのドライバマネージャを引き続き使用します。

関連情報

ALTER SERVER 文 [679 ページ]
CREATE EXTERNLOGIN 文 [816 ページ]
CREATE EXISTING TABLE 文 [812 ページ]
DROP SERVER 文 [1061 ページ]
DROP REMOTE CONNECTION 文 [1055 ページ]
SYSSERVER システムビュー [1835 ページ]

1.4.4.88 CREATE SERVICE 文 [HTTP Web サービス]

新しい HTTP Web サービスを作成します。

構文

```
CREATE [ OR REPLACE ] SERVICE service-name
TYPE { 'RAW' | 'HTML' | 'JSON' | 'XML' }
[ URL [PATH] { ON | OFF | ELEMENTS } ]
[ common-attributes ]
[ AS { statement | NULL } ]
```

```
common-attributes :
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| GET
| HEAD
| OPTIONS
| PUT
| DELETE
| NONE
| *
```

パラメータ

service-name

Web サービス名に使用できるのは、任意の順序の英数字文字、またはスラッシュ (/)、ハイフン (-)、アンダースコア (_)、ピリオド (.)、感嘆符 (!)、チルダ (~)、アスタリスク (*) 、アポストロフィ (')、左カッコ ((、右カッコ ()) です。ただし、スラッシュ (/) は、サービス名の先頭または最後の文字には使用できず、2 つ以上連続 (// など) して使用することもできません。

サービスルートに名前を付けることができますが、この名前には特別な機能があります。

TYPE 句

各サービスが特定の応答フォーマットを定義する、サービスタイプを識別します。リストされたサービスタイプのいずれかを指定します。デフォルト値はありません。

'RAW'

結果セットはフォーマットされずにクライアントに送られます。このサービスを利用するには、すべてのコンテンツのマークアップが明示的に指定されている必要があります。マークアップが付けられた現在のコンテンツを含む複雑な動的コンテンツである JavaScript およびイメージは、オンデマンドで生成できます。sa_set_http_header プロシージャを使用して Content-Type 応答ヘッダを設定することにより、メディアタイプを指定できます。すべてのブラウザでマークアップを text/plaina ではなく HTML として表示するには、HTML マークアップを生成するときに Content-Type ヘッダを 'text/html' に設定することをお奨めします。

'HTML'

結果セットはテーブルまたはビューの HTML 表現で返されます。

'JSON'

結果セットは JavaScript Object Notation (JSON) で返されます。JSON サービスは、自動的には JSON 入力を処理しません。(応答内の) データを JSON 形式で表すことのみ行います。JSON は、application/x-www-form-urlencoded がサポートされている POST/PUT メソッドを受け付けます。POST/PUT メソッドで **`Content-Type: application/json`** を指定すると、アプリケーションは http_variable('body') を使用して JSON (要求) コンテンツを取得します。データベースサーバは自動的には JSON 入力を解析しません。解析するかどうかはアプリケーションによって決まります。

'XML'

結果セットは XML として返されます。結果セットがすでに XML の場合、それ以上フォーマットは適用されません。XML でない場合は、XML として自動的にフォーマットされます。代替手段としては、sa_set_http_header プロシージャを使用して text/xml などの有効な Content-Type を設定した FOR XML RAW 句を使用して、RAW サービスから select を返す方法もあります。

URL 句

URL パスを受け入れるかどうか、受け入れる場合にはどのように処理するかを決定します。URL PATH を指定すると、URL と同じ機能があります。

OFF

URL 要求でサービス名に続いてパスを指定できないことを示します。OFF はデフォルト設定です。たとえば、パス要素 `/aaa/bbb/ccc` のため、次の形式は許可されません。

```
http://host-name/service-name/aaa/bbb/ccc
```

Web サービスの作成時に `CREATE SERVICE echo URL PATH OFF` が指定されたとします。`http://localhost/echo?id=1` に似た URL で次の値が生成されます。

Function call	Result
HTTP_VARIABLE('id')	1
HTTP_HEADER('@HttpQueryString')	id=1

ON

URL 要求でサービス名に続いてパスを指定できることを示します。URL という名前の専用の HTTP 変数を問い合わせることにより、パスの値が返されます。URL パラメータを明示的に指定するようにサービスを定義したり、HTTP_VARIABLE 関数を使用して取得したりできます。たとえば、次の形式は許可されます。

```
http://host-name/service-name/aaa/bbb/ccc
```

Web サービスの作成時に `CREATE SERVICE echo URL PATH ON` が指定されたとします。`http://localhost/echo/one/two?id=1` に似た URL で次の値が生成されます。

Function call	Result
HTTP_VARIABLE('id')	1
HTTP_VARIABLE('URL')	one/two
HTTP_HEADER('@HttpQueryString')	id=1

ELEMENTS

URL 要求でサービス名に続いてパスを指定できることを示します。**URL1**、**URL2** などの単一のパラメータキーワードを指定することにより、セグメントでパスが取得されます。各パラメータは、HTTP_VARIABLE 関数または NEXT_HTTP_VARIABLE 関数を使用して検索できます。これらの繰り返し関数は、可変の数のパス要素を指定できるアプリケーションで使用できます。たとえば、次の形式は許可されます。

```
http://host-name/service-name/aaa/bbb/ccc
```

Web サービスの作成時に `CREATE SERVICE echo URL PATH ELEMENTS` が指定されたとします。`http://localhost/echo/one/two?id=1` に似た URL で次の値が生成されます。

Function call	Result
HTTP_VARIABLE('id')	1
HTTP_VARIABLE('URL1')	one
HTTP_VARIABLE('URL2')	two
HTTP_VARIABLE('URL3')	NULL
HTTP_HEADER('@HttpQueryString')	id=1

10 個までの要素を取得できます。対応する要素が指定されていない場合は、NULL 値が返されます。上記の例では、対応する要素が指定されていないため、`HTTP_VARIABLE('URL3')` は NULL を返します。

AUTHORIZATION 句

サービスに接続するときに、ユーザが基本的な HTTP 認証を通じてユーザ名とパスワードを指定する必要があるかどうかを決定します。デフォルト値は ON です。AUTHORIZATION が OFF の場合、すべてのサービスに AS 句が必要となり、USER 句でユーザを指定する必要があります。すべての要求は、そのユーザのアカウントと権限を使用して実行されます。AUTHORIZATION が ON の場合、すべてのユーザはユーザ名とパスワードを指定する必要があります。オプションとして、USER 句でユーザ名またはグループ名を指定することにより、サービスを使用できるユーザを制限することもできます。ユーザ名が NULL の場合、既知のユーザはすべてサービスにアクセスできます。AUTHORIZATION 句を使用すると、Web サービスで、データベース認証と権限を通じてデータベース内のデータへのアクセスを制御できます。

authorization の値が ON の場合、Web サービスに接続しようとしている HTTP クライアントは、基本認証 (RFC 2617) を使用します。基本認証では、ユーザとパスワードの情報が base-64 エンコーディングを使用して難読化されます。セキュリティを強化するため、HTTPS プロトコルを使用してください。

ENABLE 句と DISABLE 句

サービスを使用できるかどうかを指定します。デフォルトでは、サービスの作成時に有効にされます。サービスを作成または変更するときに、ENABLE 句または DISABLE 句を含めることができます。サービスを無効にすると、事実上、サービスがオフラインになります。無効にしたサービスは、後で ENABLE 句を含む ALTER SERVICE 文を使用して有効にできます。無効にされたサービスに対して HTTP 要求を行うと、通常、`404 Not Found` HTTP ステータスが返されます。

METHODS 句

サービスでサポートされている HTTP メソッドを指定します。有効な値は、DEFAULT、POST、GET、HEAD、OPTIONS、PUT、DELETE、NONE です。アスタリスク (*) は、POST、GET、HEAD メソッドを表す省略形として使用できます。これらのメソッドは、RAW、HTML、XML サービスタイプのデフォルトの要求タイプです。すべての HTTP メソッドがすべてのサービスタイプに有効なわけではありません。次の表は、各サービスタイプに適用できる有効な HTTP メソッドの概要を示します。

メソッド値	適用対象のサービス	説明
DEFAULT	すべて	DEFAULT は、指定されたサービスタイプのデフォルトの HTTP メソッドのセットをリセットする場合に使用します。他のメソッド値が存在するリストに含めることはできません。
POST	RAW、HTML、JSON、XML	デフォルトでは有効になっています。
GET	RAW、HTML、JSON、XML	デフォルトでは有効になっています。
HEAD	RAW、HTML、JSON、XML	デフォルトでは有効になっています。
OPTIONS	RAW、HTML、JSON、XML	デフォルトでは有効になっていません。
PUT	RAW、HTML、JSON、XML	デフォルトでは有効になっていません。
DELETE	RAW、HTML、JSON、XML	デフォルトでは有効になっていません。
NONE	すべて	NONE は、サービスへのアクセスを無効にするために使用します。
*	RAW、HTML、JSON、XML	`'POST,GET,HEAD'` と指定することと同じです。

たとえば、次のいずれかの句を使用して、サービスですべての HTTP メソッドタイプがサポートされていることを指定できます。

```
METHODS '*,OPTIONS,PUT,DELETE'
METHODS 'POST,GET,HEAD,OPTIONS,PUT,DELETE'
```

サービスタイプに対する要求タイプのリストをデフォルトにリセットするには、次の句を使用できます。

```
METHODS 'DEFAULT'
```

SECURE 句

セキュアリスナまたは非セキュアリスナで、サービスにアクセスできるかどうかを指定します。ON は、HTTPS 接続のみが受け入れられ、HTTP ポートが受信する接続は HTTPS ポートに自動的にリダイレクトされることを示します。OFF は、

Web サーバの起動時に必要なポートが指定されていると、HTTP 接続と HTTPS 接続の両方が受け入れられることを示します。デフォルト値は OFF です。

USER 句

Web サービス要求の実行権限を持つデータベースユーザまたはユーザグループを指定します。サービスが AUTHORIZATION OFF で設定されている場合は USER 句を必ず指定し、AUTHORIZATION ON (デフォルト) で設定されている場合は、ユーザを制限する場合に指定します。認証を必要とするサービスに対して HTTP 要求を行うと、`401 Authorization Required` HTTP 応答ステータスになります。この応答に基づいて、ユーザ ID とパスワードを要求するプロンプトが Web ブラウザで表示されます。

> ⚠ 警告
>
> 認証が有効になっている (AUTHORIZATION ON: デフォルト) 場合には、USER 句を指定することを強くお奨めします。指定しないと、すべてのユーザが認証されます。

USER 句は、サービス要求を処理するのに使用できるデータベースユーザアカウントを制御します。データベースのアクセスパーミッションは、サービスのユーザに割り当てられた権限に制限されます。

statement

サービスにアクセスしたときに起動する、ストアドプロシージャ呼び出しなどのコマンドを指定します。

`statement` なしの DISH 以外のサービスに対する HTTP 要求により、URL 内で実行する SQL 式が指定されます。認証が必要ですが、この機能はサーバを SQL の挿入に対して脆弱にするため、実稼動システムでは使用しないでください。サービス内で文が定義されると、指定された SQL 文はサービスによって実行できる文のみとなります。

通常の Web サービスアプリケーションでは、`statement` を使用して関数またはプロシージャを呼び出します。ホスト変数をパラメータとして渡して、クライアント提供の HTTP 変数にアクセスできます。

次の `statement` は、AuthenticateUser という名前のプロシージャに 2 つのホスト変数を渡すプロシージャコールを示しています。この呼び出しは Web クライアントが user_name および user_password 変数を供給することを仮定しています。

```
CALL AuthenticateUser ( :user_name, :user_password );
```

備考

サービス定義は、ISYSWEBSERVICE テーブルに格納され、SYSWEBSERVICE システムビューから検査できます。

USER 句が指定された場合、`user-name` が削除されるとサービスは削除されます。

権限

MANAGE ANY WEB SERVICE システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

CREATE SERVICE は、XML と RAW タイプに関してのみ Adaptive Server Enterprise でサポートされています。

例

次の例は、JSON サービスの作成方法を示します。

-xs (http または https) オプションを指定してデータベースサーバを起動してから、次の SQL 文を実行してサービスを設定します。

```
CREATE PROCEDURE ListEmployees()
RESULT (
 EmployeeID            integer,
 Surname               person_name_t,
 GivenName             person_name_t,
 StartDate             date,
 TerminationDate       date )
BEGIN
    SELECT EmployeeID, Surname, GivenName, StartDate, TerminationDate
    FROM GROUPO.Employees
END;
CREATE SERVICE "jsonEmployeeList"
    TYPE 'JSON'
    AUTHORIZATION OFF
    SECURE OFF
    USER DBA
    AS CALL ListEmployees();
```

JSON サービスは、AJAX コールバックで簡単に使用できるデータを提供します。

次の SQL 文を実行して、読み取り可能な形式でサービスを提供する HTML サービスを作成します。

```
CREATE SERVICE "EmployeeList"
    TYPE 'HTML'
    AUTHORIZATION OFF
    SECURE OFF
    USER DBA
    AS CALL ListEmployees();
```

Web ブラウザで、`http://localhost/EmployeeList` のような URL を使用して、サービスにアクセスします。

関連情報

ALTER SERVICE 文 [HTTP Web サービス] [683 ページ]
DROP SERVICE 文 [1063 ページ]
sa_parse_json システムプロシージャ [1701 ページ]
SYSWEBSERVICE システムビュー [1862 ページ]
sa_set_http_header システムプロシージャ [1619 ページ]
識別子 [6 ページ]
ROW コンストラクタ [複合] [510 ページ]
ARRAY コンストラクタ [複合] [234 ページ]

1.4.4.89 CREATE SERVICE 文 [SOAP Web サービス]

新しい HTTP または DISH を介した SOAP サービスを作成します。

構文

HTTP を介した SOAP サービス

```
CREATE [ OR REPLACE ] SERVICE service-name
TYPE 'SOAP'
[ DATATYPE { ON | OFF | IN | OUT } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]
AS statement
```

```
common-attributes :
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| HEAD
| NONE
```

DISH サービス

```
CREATE SERVICE service-name
TYPE 'DISH'
[ GROUP { group-name | NULL } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]
```

```
common-attributes:
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
```

```
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| GET
| HEAD
| NONE
| *
```

パラメータ

service-name

Web サービス名に使用できるのは、任意の順序の英数字文字、またはスラッシュ (/)、ハイフン (-)、アンダースコア (_)、ピリオド (.)、感嘆符 (!)、チルダ (~)、アスタリスク (*) 、アポストロフィ (')、左括弧 (()、右括弧 ()) です。ただし、スラッシュ (/) は、サービス名の先頭または最後の文字には使用できず、2 つ以上連続 (// など) して使用することもできません。

他のサービスとは異なり、スラッシュ (/) は、DISH サービス名のどこにも使用できません。

サービスルートに名前を付けることができますが、この名前には特別な機能があります。

TYPE clause

各サービスが特定の応答フォーマットを定義する、サービスタイプを識別します。リストされたサービスタイプのいずれかを指定します。デフォルト値はありません。

'SOAP'

結果セットは、SOAP エンベロープと呼ばれる XML ペイロードとして返されます。FORMAT 句を使用すると、データのフォーマットをさらに絞り込むことができます。SOAP サービスへの要求は、単なる汎用 HTTP 要求ではなく有効な SOAP 要求である必要があります。

'DISH'

DISH サービス (SOAP ハンドラを決定) は、GROUP コンテキスト内で SOAP サービスを参照する SOAP 終了ポイントです。また、SOAP クライアントツールキットで使用するために WSDL (Web Services Description Language) を生成することにより、SOAP サービスに対してインタフェースも公開します。

GROUP clause

GROUP 句が指定されていない DISH サービスは、データベース内で定義されているすべての SOAP サービスを公開します。慣例により、SOAP サービス名は、GROUP 要素と NAME 要素で構成できます。最後のスラッシュ文字によって、名前がグループから区切られます。たとえば、'aaa/bbb/ccc' と定義されている SOAP サービス名は 'ccc' であり、グループは 'aaa/bbb' です。この方法で DISH サービスを区切るのは、無効です。代わりに、SOAP 終了ポイントとなる SOAP サービスのグループを指定するために、GROUP 句が適用されます。

注記

有効な XML を生成するため、WSDL ではスラッシュがアンダースコアに変換されます。GROUP 句が指定されていない DISH サービスでは、スラッシュを含む可能性のあるすべての SOAP サービスが公開されるため、このような DISH サービスを使用する場合には注意してください。アンダースコアを含む SOAP サービス名とグループを一緒に使用する場合は、あいまいにならないように注意してください。

DATATYPE clause

SOAP サービスにだけ適用されます。DATATYPE OFF が指定されていると、SOAP の入力パラメータと応答データは XMLSchema 文字列型として定義されます。ほとんどの場合、実際のデータ型を使用すると、SOAP クライアントで計算を行う前にデータをキャストする必要がなくなるため、この方法をおすすめします。パラメータのデータ型は、DISH サービスが生成する WSDL のスキーマセクションで公開されます。出力データ型は、データのカラムごとに XML スキーマ型の属性として表されます。

DATATYPE 句に指定できる値を次に示します。

ON

入力パラメータと結果セットの応答のデータ型指定を生成します。

OFF

すべての入力パラメータと応答のデータ型は、XMLSchema 文字列 (デフォルト) になります。

IN

入力パラメータに対してのみ実際のデータ型を生成します。応答のデータ型は XMLSchema 文字列になります。

OUT

応答に対してのみ実際のデータ型を生成します。入力パラメータは XMLSchema 文字列になります。

FORMAT clause

この句は、SOAP クライアントアプリケーションに応答を送信するときの出力フォーマットを指定します。

SOAP サービスフォーマットが SOAP サービスで指定されていない場合は、関連する DISH サービスフォーマットによって指定されます。デフォルトのフォーマットは DNET です。

SOAP サービスのグループにフォーマットルールを活用するには、SOAP 要求を DISH サービス (SQL Anywhere SOAP 終了ポイント) に指示してください (SOAP 操作)。SOAP サービスの FORMAT 指定は、DISH サービスの指定よりも優先されます。DISH サービスのフォーマット指定は、SOAP サービスで FORMAT 句が定義されていない場合に使用されます。どちらのサービスにも FORMAT が指定されていない場合のデフォルトは 'DNET' です。

次のフォーマットがサポートされます。

'DNET'

.NET クライアントアプリケーションで使用するために、互換性のある System.Data.DataSet フォーマットで出力されます (デフォルト)。

'CONCRETE'

この出力フォーマットは、ローとカラムのオブジェクトの配列を表すインタフェースを生成できるが、DNET フォーマットを使用できないクライアント SOAP ツールキットをサポートするために使用されます。Java クライアントおよび .NET クライアントは、この出力フォーマットを簡単に使用できます。

特定のフォーマットは、明示的なデータセットオブジェクトまたは SimpleDataset として WSDL 内で公開されます。どちらのデータセット表現も、各ローにカラムの配列が含まれているローの配列を表すデータ構造を示します。明示的なデータセットオブジェクトは、ロー内の各カラムに対応するパラメータ名とデータ型を提供することにより、結果セットの実際の形式を表す利点があります。一方、SimpleDataset はあらゆるタイプの無数のカラムが含まれているローを公開します。

FORMAT 'CONCRETE' EXPLICIT ON では、RESULT 句を定義するストアドプロシージャを Service 文で呼び出す必要があります。この条件が満たされる場合、SOAP サービスは名前がサービス名で始まり Dataset を付加した明示的なデータセットを公開します。

この条件が満たされない場合、SimpleDataset が使用されます。

'XML'

出力は XMLSchema 文字列フォーマットで生成されます。応答としての XML ドキュメントでは、カラムデータを抽出するために SOAP クライアントでさらに処理が必要です。このフォーマットは、ローおよびカラムの配列を表す中間インタフェースオブジェクトを生成できない SOAP クライアントに適しています。

NULL

NULL タイプを指定すると、SOAP サービスまたは DISH サービスでデフォルトの動作が使用されます。ALTER SERVICE 文で NULL タイプを使用すると、既存のサービスのフォーマットタイプが上書きされます。

AUTHORIZATION clause

サービスに接続するときに、ユーザが基本的な HTTP 認証を通じてユーザ名とパスワードを指定する必要があるかどうかを決定します。デフォルト値は ON です。AUTHORIZATION が OFF の場合、SOAP サービスに AS 句が必要となり、USER 句でユーザを指定する必要があります。すべての要求は、そのユーザのアカウントと権限を使用して実行されます。AUTHORIZATION が ON の場合、すべてのユーザはユーザ名とパスワードを指定する必要があります。オプションとして、USER 句でユーザ名またはグループ名を指定することにより、サービスを使用できるユーザを制限することもできます。ユーザ名が NULL の場合、既知のユーザはすべてサービスにアクセスできます。AUTHORIZATION 句を使用すると、Web サービスで、データベース認証と権限を通じてデータベース内のデータへのアクセスを制御できます。

authorization の値が ON の場合、Web サービスに接続しようとしている HTTP クライアントは、基本認証 (RFC 2617) を使用します。基本認証では、ユーザとパスワードの情報が base-64 エンコーディングを使用して難読化されます。セキュリティを強化するため、HTTPS プロトコルを使用することをおすすめします。

ENABLE and DISABLE clauses

サービスを使用できるかどうかを指定します。デフォルトでは、サービスの作成時に有効にされます。サービスを作成または変更するときに、ENABLE 句または DISABLE 句を含めることができます。サービスを無効にすると、事実上、サービスがオフラインになります。無効にしたサービスは、後で ENABLE 句を含む ALTER SERVICE 文を使用して有効にできます。無効にされたサービスに対して HTTP 要求を行うと、通常、`404 Not Found` HTTP ステータスが返されます。

METHODS clause

サービスでサポートされている HTTP メソッドを指定します。有効な値は、DEFAULT、POST、GET、HEAD、NONE です。POST、GET、HEAD メソッドを表す省略形として、アスタリスク (*) を使用できます。SOAP サービスのデフォルトのメソッドタイプは、POST と HEAD です。DISH サービスのデフォルトのメソッドタイプは、GET、POST、および HEAD です。すべての HTTP メソッドがすべてのサービスタイプに有効なわけではありません。次の表は、各サービスタイプに適用できる有効な HTTP メソッドの概要を示します。

メソッド値	適用対象のサービス	説明
DEFAULT	両方	DEFAULT は、指定されたサービスタイプのデフォルトの HTTP メソッドのセットをリセットする場合に使用します。他のメソッド値が存在するリストに含めることはできません。
POST	両方	SOAP ではデフォルトで有効になります。
GET	DISH のみ	DISH ではデフォルトで有効になります。
HEAD	両方	SOAP と DISH でデフォルトで有効になります。

メソッド値	適用対象のサービス	説明
NONE	両方	NONE は、サービスへのアクセスを無効にするために使用します。SOAP サービスに適用すると、SOAP 要求がそのサービスに直接アクセスできなくなります。このため、DISH サービスの SOAP 終了ポイント経由で SOAP 操作にアクセスします。 各 SOAP サービスに **METHODS 'NONE'** を指定することをおすすめします。
*	DISH のみ	**'POST,GET,HEAD'** と指定することと同じです。

たとえば、次のいずれかの句を使用して、サービスですべての HTTP を介した SOAP メソッドタイプがサポートされていることを指定できます。

```
METHODS 'POST,HEAD'
```

サービスタイプに対する要求タイプのリストをデフォルトにリセットするには、次の句を使用できます。

```
METHODS 'DEFAULT'
```

SECURE clause

セキュアリスナまたは非セキュアリスナで、サービスにアクセスできるかどうかを指定します。ON は、HTTPS 接続のみが受け入れられ、HTTP ポートが受信する接続は HTTPS ポートに自動的にリダイレクトされることを示します。OFF は、Web サーバの起動時に必要なポートが指定されていると、HTTP 接続と HTTPS 接続の両方が受け入れられることを示します。デフォルト値は OFF です。

USER clause

Web サービス要求の実行権限を持つデータベースユーザまたはユーザグループを指定します。サービスが AUTHORIZATION OFF で設定されている場合は USER 句を必ず指定し、AUTHORIZATION ON (デフォルト) で設定されている場合は、ユーザを制限する場合に指定します。認証を必要とするサービスに対して HTTP 要求を行うと、`401 Authorization Required` HTTP 応答ステータスになります。この応答に基づいて、ユーザ ID とパスワードを要求するプロンプトが Web ブラウザで表示されます。

> ⚠ 警告
>
> 認証が有効になっている (AUTHORIZATION ON: デフォルト) 場合には、USER 句を指定することを強くおすすめします。指定しないと、すべてのユーザが認証されます。

USER 句は、サービス要求を処理するのに使用できるデータベースユーザアカウントを制御します。データベースのアクセスパーミッションは、サービスのユーザに割り当てられた権限に制限されます。

statement

サービスにアクセスしたときに起動する、ストアドプロシージャ呼び出しなどのコマンドを指定します。

DISH サービスは、NULL 文または文のいずれかを指定する必要がある唯一のサービスです。SOAP サービスで文を定義する必要があります。他のサービスが NULL 文を持つ場合もありますが、AUTHORIZATION ON で設定した場合だけです。

`statement` なしの DISH 以外のサービスに対する HTTP 要求により、URL 内で実行する SQL 式が指定されます。認証が必要ですが、この機能はサーバを SQL の挿入に対して脆弱にするため、本稼働システムでは使用しないでください。サービス内で文が定義されると、指定された SQL 文はサービスによって実行できる文のみとなります。

通常の Web サービスアプリケーションでは、`statement` を使用して関数またはプロシージャを呼び出します。ホスト変数をパラメータとして渡して、クライアント提供の HTTP 変数にアクセスできます。

次の `statement` は、AuthenticateUser という名前のプロシージャに 2 つのホスト変数を渡すプロシージャコールを示しています。この呼び出しは Web クライアントが user_name および user_password 変数を供給することを仮定しています。

```
CALL AuthenticateUser ( :user_name, :user_password );
```

備考

サービス定義は、ISYSWEBSERVICE テーブルに格納され、SYSWEBSERVICE ビューから検査できます。

権限

MANAGE ANY WEB SERVICE システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

CREATE SERVICE は、SOAP タイプに関してのみ Adaptive Server Enterprise でサポートされています。

関連情報

ALTER SERVICE 文 [SOAP Web サービス] [688 ページ]
DROP SERVICE 文 [1063 ページ]
SYSWEBSERVICE システムビュー [1862 ページ]

1.4.4.90 CREATE SPATIAL REFERENCE SYSTEM 文

空間参照系を作成するか、置き換えます。

構文

```
{ CREATE [ OR REPLACE ] SPATIAL REFERENCE SYSTEM
| CREATE SPATIAL REFERENCE SYSTEM IF NOT EXISTS }
srs-name
[ srs-attribute [ srs-attribute ... ]]
```

```
srs-name :string
```

```
srs-attribute :
IDENTIFIED BY srs-id
| DEFINITION { definition-string | NULL }
| ORGANIZATION { organization-name IDENTIFIED BY organization-srs-id | NULL }
| TRANSFORM DEFINITION { transform-definition-string | NULL }
| LINEAR UNIT OF MEASURE linear-unit-name
| ANGULAR UNIT OF MEASURE { angular-unit-name | NULL }
| TYPE { ROUND EARTH | PLANAR }
| COORDINATE coordinate-name { UNBOUNDED | BETWEEN low-number AND high-
number }
| ELLIPSOID SEMI MAJOR AXIS semi-major-axis-length { SEMI MINOR AXIS semi-minor-axis-
length | INVERSE FLATTENING inverse-flattening-ratio }
| SNAP TO GRID { grid-size | DEFAULT }
| TOLERANCE { tolerance-distance | DEFAULT }
| AXIS ORDER axis-order
| POLYGON FORMAT polygon-format
| STORAGE FORMAT storage-format
```

```
srs-id :integer
```

```
semi-major-axis-length :number
```

```
semi-minor-axis-length :number
```

```
inverse-flattening-ratio :number
```

```
grid-size :DOUBLE: 通常 0 ～ 1
```

```
tolerance-distance :number
```

```
axis-order : { 'x/y/z/m' | 'long/lat/z/m' | 'lat/long/z/m' }
```

```
polygon-format : { 'CounterClockWise' | 'Clockwise' | 'EvenOdd' }
```

```
storage-format : { 'Internal' | 'Original' | 'Mixed' }
```

パラメータ

OR REPLACE clause

OR REPLACE を指定すると、空間参照系がデータベースにまだ存在しない場合は空間参照系が作成され、存在する場合は置き換えられます。使用中の空間参照系を置き換えようとすると、エラーが返されます。OR REPLACE 句を指定しないでデータベースにすでに存在する空間参照系を置き換えようとしても、エラーが返されます。

CREATE SPATIAL REFERENCE IF NOT EXISTS

CREATE SPATIAL REFERENCE IF NOT EXISTS を指定すると、指定された名前の空間参照系がデータベースにすでに存在するかどうかがチェックされます。存在しない場合は、データベースサーバによって空間参照系が作成されます。存在する場合は、追加のアクションは行われず、エラーも返されません。

IDENTIFIED BY clause

この句は、空間参照系の SRID (`srs-id`) を指定するときに使用します。組織で `organization-srs-id` を指定して空間参照系を定義した場合は、`srs-id` をその値に設定してください。

IDENTIFIED BY 句が指定されていない場合は、デフォルトにより、ORGANIZATION 句または DEFINITION 句で定義された `organization-srs-id` に SRID が設定されます。デフォルトの SRID として使用できる `organization-srs-id` がどちらの句にも指定されていない場合は、エラーが返されます。

空間参照系が周知の座標系に基づいておらず、異なる測地線解釈がされている場合は、1000000000 (10 億) に周知の値を加えた値に `srs-id` 値を設定します。たとえば、測地線の空間参照系 WGS 84 (ID 4326) の平面解釈の SRID は、1000004326 になります。

SQL Anywhere で提供される周知の座標系に基づいていない空間参照系には、SRID 0 を除き 2000000000 (20 億) 以上の SRID が付与されます。2000000000 ～ 2147483647 の範囲の SRID 値は SQL Anywhere で予約されているため、この範囲の SRID は作成しないでください。

OGC などの定義機関または他のベンダで予約されている SRID を選択する可能性を低減するため、0 ～ 32767 (EPSG で予約済み) または 2147483547 ～ 2147483647 の範囲の SRID は選択しないでください。

また、SRID は 32 ビットの符号付き整数として格納されるため、2^{31}-1 または 2147483647 を超える数値は指定できません。

DEFINITION clause

この句は、座標系のデフォルトを設定または上書きするときに使用します。DEFINITION 句以外の句で属性が設定されている場合は、DEFINITION 句の設定内容に関係なく、他の句で指定された値が優先されます。

`definition-string` は、SQL/MM と OGC で定義されるような、空間参照系の Well Known Text 構文の文字列です。たとえば、次のクエリは WGS 84 の定義を返します。

```
SELECT  ST_SpatialRefSys::ST_FormatWKT( definition )
   FROM ST_SPATIAL_REFERENCE_SYSTEMS
   WHERE srs_id=4326;
```

Interactive SQL で戻り値をダブルクリックすると、読みやすい形で値が表示されます。

DEFINITION 句が指定されている場合は、definition-string が解析され、属性のデフォルト値の選択に使用されます。たとえば、definition-string に、organization-name と organization-srs-id を定義する AUTHORITY 要素が含められる場合があります。

definition-string のパラメータ値は、SQL 文の句で明示的に設定した値で上書きされます。たとえば、ORGANIZATION 句が指定されていると、definition-string の ORGANIZATION の値が上書きされます。

ORGANIZATION clause

この句は、新しい空間参照系の基となる空間参照系を作成した組織に関する情報を指定するときに使用します。organization-name は、空間参照系を作成した組織の名前で、organization-srs-id は、空間参照系を識別するために組織で使用する数値識別子です。

TRANSFORM DEFINITION clause

この句は、空間参照系に使用する変換の説明を指定するときに使用します。現時点では、PROJ.4 変換のみがサポートされています。たとえば、WGS 84 の transform-definition-string は '+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs' です。

サポートされていない変換定義を指定すると、エラーが返されます。

変換定義は、空間参照系間でデータを変換するときに ST_Transform メソッドで使用されます。一部の変換には、transform-definition-string が定義されていなくてもかまいません。

LINEAR UNIT OF MEASURE clause

この句は、空間参照系の線形測定単位を指定するときに使用します。指定する値は、ST_UNITS_OF_MEASURE 連結ビューで定義された線形測定単位と一致している必要があります。

この句が指定されておらず、DEFINITION 句で定義されていない場合、デフォルトは METRE です。

事前に定義された測定単位をデータベースに追加するには、sa_install_feature system プロシージャを使用します。

カスタム測定単位をデータベースに追加するには、CREATE SPATIAL UNIT OF MEASURE 文を使用します。

> **注記**
>
> METRE と METER はいずれも有効なスペルですが、SQL/MM 標準に準拠している METRE を使用することをおすすめします。

ANGULAR UNIT OF MEASURE clause

この句は、空間参照系の角度測定単位を指定するときに使用します。指定する値は、ST_UNITS_OF_MEASURE 統合ビューで定義された角度測定単位と一致している必要があります。

この句が指定されておらず、DEFINITION 句で定義されていない場合、デフォルトは、地理的空間参照系の場合は DEGREE、非地理的空間参照系の場合は NULL です。

角度測定単位は、地理的空間参照系の場合は NULL 以外、非地理的空間参照系の場合は NULL にしてください。

事前に定義された測定単位をデータベースに追加するには、sa_install_feature system プロシージャを使用します。

カスタム測定単位をデータベースに追加するには、CREATE SPATIAL UNIT OF MEASURE 文を使用します。

TYPE clause

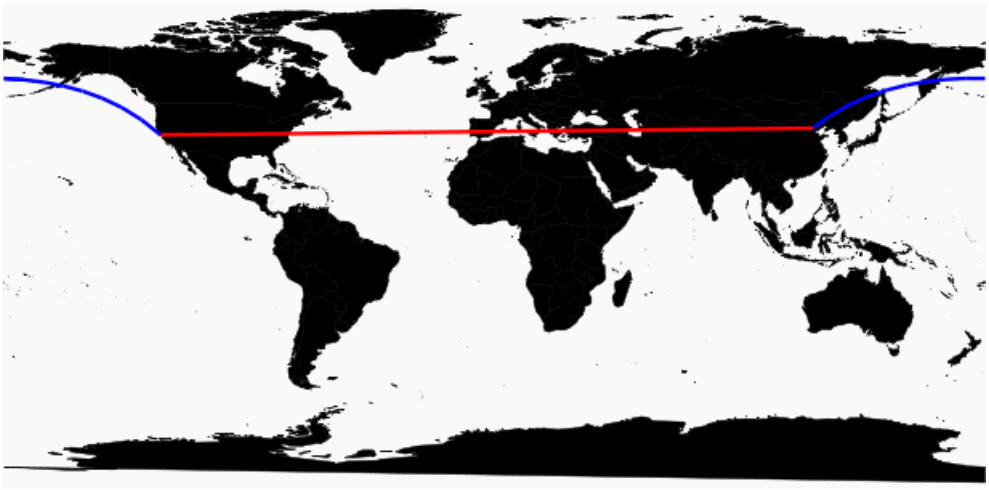
この TYPE 句は、SRS で点と点を結ぶ線を解釈する方法を制御するために使用します。地理的空間参照系では、TYPE 句で ROUND EARTH (デフォルト) または PLANAR のいずれかを指定できます。ROUND EARTH モデルでは、点と点

を結ぶ線が大楕円弧として解釈されます。地表面の 2 点を指定し、この 2 点および地球の中心と交差するように平面を選択したとします。この平面は地球と交差し、2 点を結ぶ線はこの交差に沿った最短距離になります。

2 点がまったく反対側にある場合、2 点および地球の中心と交差する単一でユニークな平面は存在しません。このような正反対の点を結ぶ線セグメントは無効であり、ROUND EARTH モデルでエラーになります。

ROUND EARTH モデルでは、地球が回転楕円体と見なされ、地球の湾曲に沿った線が選択されます。場合によっては、2 点間の線が x= 経度、y= 緯度の正距円筒図法における直線として解釈される平面モデルの使用が必要となることがあります。

次の例では、青い線は ROUND EARTH モデルで使用される線の解釈を示し、赤い線は対応する PLANAR モデルを示します。



PLANAR モデルは、他のソフトウェア製品で使用される空間解釈と一致させるために使用できます。また、ROUND EARTH モデルでサポートされていないメソッド (ST_Area、ST_ConvexHull など) には制限があり、一部のメソッドは部分的にのみサポートされている (ST_Distance はポイントジオメトリでのみサポートされる) ため、PLANAR モデルも便利です。円ストリングに基づいたジオメトリは、ROUND EARTH 空間参照系ではサポートされていません。

非地理的 SRS の場合には、タイプを PLANAR にしてください (これが、TYPE 句が指定されておらず、DEFINITION 句が指定されていないか非地理的定義を使用している場合のデフォルトです)。

COORDINATE clause

この句は、空間参照系の次元に境界を指定するときに使用します。`coordinate-name` は、空間参照系で使用される座標系の名前です。非地理的座標系の場合は、`coordinate-name` に x、y、または m を使用できます。地理的座標系の場合は、`coordinate-name` に LATITUDE、LONGITUDE、z、または m を使用できます。

次元で境界を配置しない場合は、UNBOUNDED を指定します。境界の上限と下限を設定するには、BETWEEN 句を使用します。

X 座標と Y 座標には、関連する境界が必要です。地理的座標系の場合は、これらの設定を COORDINATE 句で上書きしないかぎり、デフォルトにより経度座標は -180 ～ 180 度の間、緯度座標は -90 ～ 90 の間になります。非地理的座標系の場合は、X 座標と Y 座標の両方について CREATE 文で境界を指定してください。

LATITUDE と LONGITUDE は、地理的座標系に使用されます。LATITUDE と LONGITUDE の境界は、指定がなければデフォルトにより地球全体になります。

ELLIPSOID clause

ELLIPSOID 句は、タイプ ROUND EARTH の空間参照系の楕円として、地球を表現するときに使用する値を指定するために使用します。DEFINITION 句が指定されている場合は、この句で楕円定義を指定できます。ELLIPSOID 句が指定されていると、このデフォルトの楕円が上書きされます。

地球の自転が引き起こす扁平化により地球の中心から北極または南極までの距離が赤道までの距離より短くなるため、地球は完全な球体ではありません。このため、地球は、半長径 (中心から赤道までの距離) と半短径 (中心から極までの距離) に異なる値を使用する楕円としてモデル化されます。半長径と逆扁平率を使用して楕円を定義するのが最も一般的ですが、代わりに半短径を使用して指定することもできます (たとえば、完全な球形を使用して地球を近似させる場合には、この方法を使用します)。半長径と半短径は空間参照系の線形単位で定義され、逆扁平率 (1/f) は次の式で求められる比率です。

```
1/f = (semi-major-axis) / (semi-major-axis - semi-minor-axis)
```

SQL Anywhere では、地理的空間参照系で距離を計算するときに楕円定義が使用されます。

地理的空間参照系では (DEFINITION 句または ELLIPSOID 句で) 楕円を定義する必要がありますが、非地理的空間参照系では指定しません。

SNAP TO GRID clause

平面空間参照系では、SNAP TO GRID 句を使用して、SQL Anywhere で計算に使用されるグリッドサイズを定義します。デフォルトでは、X と Y の空間の境界におけるすべての点で 12 有効桁数を格納できるように SQL Anywhere でグリッドサイズが選択されます。たとえば、空間参照系の境界が X は -180 ～ 180 の間、Y は -90 ～ 90 の間の場合、グリッドサイズ 0.000000001 (1E-9) が選択されます。

`grid-size` には、グリッドにスナップした点を、バインド済み空間のすべての点において同じ精度で表現できるだけの、十分に大きい値を指定してください。`grid-size` が小さすぎると、サーバからエラーが通知されます。

0 に設定すると、グリッドへのスナップは行われません。

曲面空間参照系では、SNAP TO GRID を 0 に設定してください。

SNAP TO GRID DEFAULT を指定すると、グリッドサイズがデータベースサーバで使用されるデフォルトに設定されます。

TOLERANCE clause

平面空間参照系では、TOLERANCE 句を使用して、点を比較するときに使用する精度を指定します。2 点間の距離が `tolerance-distance` より小さい場合、その 2 点は同じと見なされます。`tolerance-distance` を指定することにより、入力データまたは制限された内部精度の不正確さに対する許容度を制御できます。デフォルトでは、`tolerance-distance` は `grid-size` と等しくなるように設定されます。

0 に設定すると、2 点が完全に等しい場合にのみ同じであると見なされます。

曲面空間参照系では、TOLERANCE を 0 に設定してください。

POLYGON FORMAT clause

SQL Anywhere では、多角形を構成するリングの方向によって多角形が内部的に解釈されます。定義された点の順序でリングを移動した場合、多角形の内側がリングの左側になります。PLANAR と ROUND EARTH の空間参照系では、同じルールが適用されます。

SQL Anywhere で使用される解釈は一般的ですが、汎用的な解釈ではありません。正反対の方向を使用する製品もあれば、多角形の解釈にリングの方向を使用しない製品もあります。POLYGON FORMAT 句を使用して、必要に応じて入力データと一致する多角形の解釈を選択できます。次の値がサポートされます。

'CounterClockwise'

入力は SQL Anywhere の内部解釈に従います。リングの方向に従って、多角形の内側が左側になります。

'Clockwise'

入力は SQL Anywhere の内部解釈と反対になります。リングの方向に従って、多角形の内側が右側になります。

'EvenOdd'

EvenOdd はデフォルトフォーマットです。EvenOdd を指定すると、リングの方向は無視され、代わりに多角形の内部はリングのネストによって決まります。外部のリングは最大のリングとなり、内部のリングはこのリングの内側の小さなリングとなります。斜線は、リング内の点から、すべてのリングを交差して外側に向かってトレースされます。交差するリングの数が偶数の場合は、外部リングです。奇数の場合は、内部リングです。

STORAGE FORMAT clause

空間データを外部フォーマット (WKT や WKB など) からデータベースに挿入するとき、空間操作のパフォーマンスとセマンティックを向上するため、データベースサーバでデータが正規化されます。正規化された表現は、元の表現と異なる場合があります (多角形リングの方向や個々の座標に格納される精度など)。空間の等価性は正規化の後でも保持されますが、精度やリング方向など、元の入力特性の中には再現できないものもあります。場合によっては、元の表現を、正規化された表現とともに、または明示的に格納することがあります。

格納内容を制御するには、STORAGE FORMAT 句に続けて、次のいずれかの値を指定します。

'Internal'

SQL Anywhere は正規化された表現のみを格納します。元の入力特性を再現する必要がない場合にこの値を指定します。これは、平面 (TYPE PLANAR) の空間参照系のデフォルトです。

> **注記**
>
> Mobile Link を使用して空間データを同期する場合は、代わりに **Mixed** を指定してください。Mobile Link では同期時に等価性がテストされるため、データが元のフォーマットである必要があります。

'Original'

SQL Anywhere は元の表現のみを格納します。元の入力特性を再現できますが、格納された値のすべての操作に対して正規化の手順を繰り返す必要があり、データの操作が遅くなる可能性があります。

'Mixed'

SQL Anywhere で内部バージョンが格納されます。元のバージョンと異なる場合は、元のバージョンも格納されます。両方のバージョンを格納すると、元の表現特性を再現でき、格納された値の操作に対して正規化の手順を繰り返す必要はありません。ただし、ジオメトリごとに 2 つの表現が格納される可能性があるため、記憶領域の要件が大幅に増加します。

Mixed は、曲面空間参照系 (TYPE ROUND EARTH) のデフォルトフォーマットです。

備考

地理空間参照系では、LINEAR と ANGULAR の両方の測定単位を指定できます。非地理空間参照系では、LINEAR 測定単位のみを指定できます。LINEAR 測定単位は、点と領域間の距離を計算するために使用されます。ANGULAR 測定単位は、角度の緯度と経度を解釈する方法を示し、投影座標系の場合は NULL、地理的座標系の場合は NULL 以外になります。

操作から返される派生ジオメトリはすべて正規化されます。

SQL Anywhere 以外のデータベースで同期されるデータを操作する場合は、データの元の特性を保持できるようにするため、STORAGE FORMAT を 'Original' または 'Mixed' のいずれかに設定してください。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワード (IDENTIFIED BY 句) を指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE ANY SPATIAL OBJECT または CREATE ANY OBJECT のシステム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、空間参照系 mySpatialRS を作成します。

```
CREATE SPATIAL REFERENCE SYSTEM  "mySpatialRS"
IDENTIFIED BY 1000026980
LINEAR UNIT OF MEASURE "metre"
TYPE PLANAR
COORDINATE X BETWEEN 171266.736269555 AND 831044.757769222
COORDINATE Y BETWEEN 524881.608973277 AND 691571.125115319
DEFINITION 'PROJCS["NAD83 / Kentucky South",
GEOGCS["NAD83",
DATUM["North_American_Datum_1983",
SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],
AUTHORITY["EPSG","6269"]],
PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
AUTHORITY["EPSG","4269"]],
UNIT["metre",1,AUTHORITY["EPSG","9001"]],
PROJECTION["Lambert_Conformal_Conic_2SP"],
PARAMETER["standard_parallel_1",37.93333333333333],
PARAMETER["standard_parallel_2",36.73333333333333],
PARAMETER["latitude_of_origin",36.33333333333334],
PARAMETER["central_meridian",-85.75],
PARAMETER["false_easting",500000],
PARAMETER["false_northing",500000],
AUTHORITY["EPSG","26980"],
AXIS["X",EAST],
AXIS["Y",NORTH]]'
TRANSFORM DEFINITION '+proj=lcc
+lat_1=37.93333333333333+lat_2=36.73333333333333+lat_0=36.33333333333334+lon_0=-85
.75+x_0=500000+y_0=500000+ellps=GRS80+datum=NAD83+units=m+no_defs';
```

関連情報

sa_install_feature システムプロシージャ [1539 ページ]
CREATE SPATIAL UNIT OF MEASURE 文 [940 ページ]
ST_UNITS_OF_MEASURE 統合ビュー [1869 ページ]
ST_SPATIAL_REFERENCE_SYSTEMS 統合ビュー [1866 ページ]
ALTER SPATIAL REFERENCE SYSTEM 文 [694 ページ]
sa_install_feature システムプロシージャ [1539 ページ]

1.4.4.91 CREATE SPATIAL UNIT OF MEASURE 文

空間測定単位を作成するか、置き換えます。

構文

```
CREATE [ OR REPLACE ] SPATIAL UNIT OF MEASURE identifier
TYPE {LINEAR | ANGULAR }
[ CONVERT USING number ][ IF NOT EXISTS ]
```

パラメータ

OR REPLACE clause

OR REPLACE の指定は、空間測定単位を作成するか、既存の空間測定単位を同じ名前で置き換えます。この句は、現在の権限を保持します。使用中の空間単位を置き換えようとすると、エラーが返されます。

この句を指定する場合、IF NOT EXISTS 句を指定することはできません。

TYPE clause

角度 (ANGULAR) または距離 (LINEAR) のどちらに測定単位を使用するかを定義します。

CONVERT USING

ベース単位に関係する空間単位の換算係数。線形単位の場合、ベース単位は METRE です。角度単位の場合、ベース単位は RADIAN です。

IF NOT EXISTS

空間測定単位が存在せず、メッセージを返さない場合は、空間測定単位を作成します。OR REPLACE 句を指定している場合、この句を指定することはできません。

備考

CONVERT USING 句は、定義された測定単位の値をベース測定単位 (ラジアンまたはメートル) に換算する方法を定義するために使用します。指定された換算係数を測定値に乗算して、ベース測定単位の値を取得します。たとえば、測定値 512 ミリメートルに換算係数 0.001 を乗算して、測定値 0.512 メートルを取得します。

距離 (ST_Distance または ST_Length) または領域を計算する場合、空間参照系では常に線形測定単位が使用されます。たとえば、空間参照系の線形測定単位がマイルの場合、使用する領域単位は平方マイルになります。場合によっては、線形測定単位を使用するよう指定するオプションパラメータが空間メソッドで受け入れられることがあります。たとえば、空間参照系の線形測定単位がマイルの場合、オプションパラメータ 'metre' を使用すると、2 つのジオメトリ間の距離をメートルで取得できます。

投影座標系の場合は、空間参照系の線形単位で X 座標と Y 座標が指定されます。地理的座標系の場合は、空間参照系に関連する角度測定単位で緯度と経度が指定されます。多くの場合、この角度測定単位は度ですが、有効な任意の角度測定単位を使用できます。

sa_install_feature システムプロシージャを使用して、事前に定義された測定単位をデータベースに追加できます。

権限

MANAGE ANY SPATIAL OBJECT または CREATE ANY OBJECT のシステム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Test という空間測定単位を作成します。

```
CREATE SPATIAL UNIT OF MEASURE Test
TYPE LINEAR
CONVERT USING 15;
```

関連情報

sa_install_feature システムプロシージャ [1539 ページ]
DROP SPATIAL UNIT OF MEASURE 文 [1065 ページ]

1.4.4.92 CREATE STATISTICS 文

オプティマイザが使用するカラムの統計情報を再作成し、ISYSCOLSTAT システムテーブルに格納します。

構文

```
CREATE STATISTICS object-name [ ( column-list ) ]
```

```
object-name :
table-name | materialized-view-name | temp-table-name
```

備考

CREATE STATISTICS 文は、データベースサーバがデータベースクエリの最適化に使用するカラムの統計情報を再作成します。また、ベーステーブル、マテリアライズドビュー、ローカルテンポラリテーブル、グローバルテンポラリテーブルで実行できます。プロキシテーブルに統計情報は作成できません。カラムの統計情報には、指定したカラムに関してデータベース内でのデータ分散を反映するヒストグラムが含まれます。デフォルトで、カラムの統計情報は 5 つ以上のローを持つテーブルでは自動的に作成されます。

データベースクエリの変化が激しい場合や、データが不均等に分散していたり、頻繁に変更されたりする場合は、テーブルまたはカラムに対して CREATE STATISTICS 文を実行すると、パフォーマンスを改善できることもあります。

統計情報がすでに存在している場合、テーブルが空でなければテーブルのサイズに関係なく、CREATE STATISTICS 文は既存のカラムの統計情報を更新します。テーブルが空の場合は何も実行されません。空のテーブルについてカラムの統計情報が存在する場合、CREATE STATISTICS 文を使用しても変化しません。空のテーブルについてカラムの統計情報を削除するには、DROP STATISTICS 文を実行します。

CREATE STATISTICS の実行プロセスは、テーブルの完全スキャンを実行します。このため、慎重な検討を行ってから CREATE STATISTICS 文を実行してください。

統計情報を削除する場合は、CREATE STATISTICS 文を使用して再作成することをおすすめします。統計情報がない場合、オプティマイザは非効率的なデータアクセスプランを生成し、データベースパフォーマンスの低下を招くことがあります。

CREATE STATISTICS 文と DROP STATISTICS 文は、統計情報に関するコミットを必要としませんし、コミットしても意味はありません。DROP STATISTICS だけは関連する動作としてコミットが行われます。ただし、コミットしても統計情報の変更が保存されるわけではありません (統計ガバナーが保守を行います)。したがって、DROP STATISTICS の関連する動作としてオートコミットが行われるとは言っても、コミットされるのは統計情報の消去を除いたすべての変更です。

権限

テーブル所有者であるか、MANAGE ANY STATISTICS または CREATE ANY OBJECT システム権限を持っている必要があります。

関連する動作

実行プランは変化する可能性があります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、SalesOrderItems テーブルの ProductID カラムに対するカラムの統計情報を更新します。

```
CREATE STATISTICS GROUPO.SalesOrderItems ( ProductID );
```

関連情報

DROP STATISTICS 文 [1067 ページ]
LOAD TABLE 文 [1182 ページ]
SYSCOLSTAT システムビュー [1796 ページ]
sa_get_histogram システムプロシージャ [1515 ページ]

1.4.4.93 CREATE SUBSCRIPTION 文 [SQL Remote]

パブリケーションに対してユーザのサブスクリプションを作成します。

構文

```
CREATE SUBSCRIPTION
TO publication-name [ ( subscription-value ) ]
FOR subscriber-id
```

```
publication-name : identifier
```

```
subscription-value : string
```

```
subscriber-id : string
```

パラメータ

publication-name

このパブリケーション名でユーザのサブスクリプションが作成されます。パブリケーションの所有者を含めることもできます。

subscription-value

パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは、サブスクリプション式がサブスクリプション値と一致するすべてのローを受信します。

`subscription-value` の変数名を指定できます。変数が NULL の場合、空の文字列が指定されたように動作します。

subscriber-id

パブリケーションに対するサブスクライバのユーザ ID。統合データベースで、リモートユーザにサブスクリプションを作成する場合、そのリモートユーザに REMOTE 権限が付与されている必要があります。リモートデータベースで、統合ユーザにサブスクリプションを作成する場合、そのユーザに CONSOLIDATED 権限が付与されている必要があります。

備考

SQL Remote では、パブリケーションとサブスクリプションは双方向の関係です。統合データベース上のパブリケーションに対してリモートユーザ用のサブスクリプションを作成するときにはリモートデータベース上のパブリケーションの統合ユーザにもサブスクリプションを作成してください。抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードでは、デフォルトで、適切な PUBLISH 権限と CONSOLIDATE 権限が、リモートデータベース内のユーザに付与されます。

subscription-value が指定されている場合、この値がパブリケーションの各 SUBSCRIBE BY 式と比較されます。サブスクライバは、指定した文字列が式の値と一致するすべてのローを受信します。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合はエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、パブリケーション pub_sales にユーザ p_chin のサブスクリプションを作成します。サブスクライバは、サブスクリプション式が値 Eastern と一致するすべてのローを受信します。

```
CREATE SUBSCRIPTION
TO pub_sales ( 'Eastern' )
FOR p_chin;
```

次の文では、CustomerPub パブリケーションに対してユーザ Sam_Singer のサブスクリプションを作成します。このパブリケーションは、WHERE 句を使用して作成されます。

```
CREATE SUBSCRIPTION
  TO CustomerPub
  FOR Sam_Singer;
```

次の文では、PubOrders パブリケーションに対してユーザ Sam_Singer のサブスクリプションを作成します。このパブリケーションには、サブスクリプション式 SalesRepresentative が定義されていて、Sam Singer 自身の受注に対してローを要求します。

```
CREATE SUBSCRIPTION
  TO PubOrders ( '856' )
  FOR Sam_Singer;
```

例

次の文は、値 'ny' の変数を作成します。

```
CREATE VARIABLE @state LONG VARCHAR ;
SET @state = 'ny' ;
```

次の文は pub_customer という名前のパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
```

```
TABLE DBA>Customer ( ID, company_name, City, State )
SUBSCRIBE BY State ) ;
```

次の文は、ユーザ名 Sam_Singer のサブスクリプションを作成します。

```
CREATE SUBSCRIPTION
   TO pub_customer ( @dan )
   FOR Sam_Singer ;
```

サブスクライバは、サブスクリプション式が変数 @dan の値 NY と一致するすべてのローを受信します。

関連情報

DROP SUBSCRIPTION 文 [SQL Remote] [1069 ページ]
GRANT REMOTE 文 [SQL Remote] [1149 ページ]
GRANT CONSOLIDATE 文 [SQL Remote] [1141 ページ]
SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote] [1362 ページ]
START SUBSCRIPTION 文 [SQL Remote] [1341 ページ]
GRANT PUBLISH 文 [SQL Remote] [1147 ページ]
SYSSUBSCRIPTION システムビュー [1839 ページ]

1.4.4.94 CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]

SQL Anywhere 同期プロファイルを作成します。

構文

```
CREATE [ OR REPLACE ] SYNCHRONIZATION PROFILE name string
```

パラメータ

OR REPLACE 句

CREATE OR REPLACE SYNCHRONIZATION PROFILE を指定すると、指定された同期プロファイルが存在する場合は、その定義が置き換えられます。

name

作成する同期プロファイルの名前を指定します。各プロファイルの名前はユニークにします。

string

次の説明に従って有効なオプション文字列を指定します。オプション文字列は、option-name=option-value 形式のエレメントのセミコロンで区切ったリストとして指定します。たとえば subscription=s1;verbosity=high のように指定します。

備考

同期プロファイルは、同期の制御に使用できる同期オプションの名前付きコレクションです。

ブール値を取るオプションの場合、値を TRUE に設定すると、コマンドラインで対応するオプションを指定したときと同じになります。

TRUE を指定するには、TRUE、ON、1、YES の値を使用できます。

FALSE を指定するには、FALSE、OFF、0、NO の値を使用できます。

拡張オプションの設定には、次の構文を使用します。

```
CREATE SYNCHRONIZATION PROFILE myprofile
's=mysub;e={ctp=tcpip;adr=''host=localhost;port=2439''}'
```

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link] [700 ページ]
DROP SYNCHRONIZATION PROFILE 文 [Mobile Link] [1070 ページ]
SYNCHRONIZE 文 [Mobile Link] [1358 ページ]

1.4.4.95 CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]

SQL Anywhere リモートデータベースで、Mobile Link ユーザとパブリケーションとの間のサブスクリプションを作成します。

構文

```
CREATE SYNCHRONIZATION SUBSCRIPTION [ subscription-name ]
TO publication-name
[ FOR ml-username, ... ]
[ TYPE network-protocol ]
[ ADDRESS protocol-options ]
[ OPTION option=value, ... ]
[ SCRIPT VERSION script-version ]
```

```
subscription-name : identifier
```

```
ml-username : identifier
```

```
network-protocol : http | https | tls | tcpip
```

```
protocol-options : string
```

```
value : string | integer
```

```
script-version : string
```

パラメータ

subscription-name

このサブスクリプションの識別に使用する一意の名前。代わりに名前付きパラメータを使用することをおすすめします。

TO 句

この句は、パブリケーション名を指定します。

FOR 句

この句は、1 つ以上の Mobile Link ユーザ名を指定します。ユーザ名を複数指定した場合、ユーザごとに別個のサブスクリプションが作成されます。サブスクリプション名を指定した場合、1 つの Mobile Link ユーザ名のみを指定できます。

`ml-username` は、Mobile Link サーバと同期することが許可されているユーザです。

FOR 句を省略すると、パブリケーションに対するプロトコルタイプ、プロトコルオプション、拡張オプションが設定されます。FOR 句を省略した場合、サブスクリプション名の指定または SCRIPT VERSION 句の使用はできません。

TYPE 句

同期に使用するネットワークプロトコルを指定します。デフォルトのプロトコルは tcpip です。

ADDRESS 句

Mobile Link サーバのロケーションなどのネットワークプロトコルオプションを指定します。複数のオプションは、セミコロンで区切る必要があります。

OPTION 句

サブスクリプションについて拡張オプションを設定できます。FOR 句を指定しない場合、拡張オプションはパブリケーションのデフォルト設定として機能します。

SCRIPT VERSION 句

この句は、同期中に使用するスクリプトバージョンを指定します。通常は、実装するスキーマを変更するたびに新しいスクリプトバージョンを指定してください。

FOR 句を省略した場合は、SCRIPT VERSION 句を使用できません。

備考

`subscription-name` が指定されていない場合は、ユニークな名前が生成されます。パブリケーション名がユニークである場合、生成されるサブスクリプション名は、パブリケーションと同じになります。それ以外の場合は、パブリケーション名の末尾に数値が追加されることによって、ユニークな名前 (pub001、pub002 など) が形成されます。

n`network-protocol`、`protocol-options`、および `option` の値は、複数の個所で設定できます。

この文を使用すると、オプションや他の情報が ISYSSYNC システムテーブルに格納されます。SYSSYNC システムビューを表示する正当な権限を持っているユーザなら誰でも、パスワードや暗号化証明書が含まれている可能性のある情報を表示できます。このとき考えられるセキュリティ上の問題を回避するために、dbmlsync コマンドラインに関する情報を指定できます。

この文を実行するには、パブリケーションで参照されるすべてのテーブルに排他的にアクセスできる権限が必要です。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Mobile Link ユーザ SSinger と、user が所有するパブリケーション sales_publication との間に、sales という名前のサブスクリプションを作成します。サブスクリプションが同期されると、スクリプトバージョン sales_v1 が使用され、テーブルが排他モードでロックされます。

```
CREATE SYNCHRONIZATION SUBSCRIPTION sales
TO user.sales_publication
FOR SSinger
OPTION locktables='exclusive'
SCRIPT VERSION 'sales_v1'
```

次の例は、FOR 句を省略し、パブリケーション sales_publication の設定を格納します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
TO user.sales_publication
ADDRESS 'host=test.internal;port=2439'
OPTION locktables=exclusive';
```

関連情報

ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [701 ページ]
DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [1071 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.96 CREATE SYNCHRONIZATION USER 文 [Mobile Link]

SQL Anywhere リモートデータベースで Mobile Link ユーザを作成します。

構文

```
CREATE SYNCHRONIZATION USER ml-username
[ TYPE network-protocol ]
[ ADDRESS protocol-options ]
[ OPTION option=value, ... ]
```

```
ml-username : identifier
```

```
network-protocol :
tcpip
| http
| https
| tls
```

```
protocol-options : string
```

```
value : string | integer
```

パラメータ

ml_username

Mobile Link ユーザを識別する名前。

TYPE 句

同期に使用するネットワークプロトコルを指定します。デフォルトのプロトコルは tcpip です。

ADDRESS 句

この句は、`protocol-options` を `keyword=value` の形式でセミコロンで区切って指定します。どのような設定を指定するかは、使用する通信プロトコル (TCP/IP、TLS、HTTP、HTTPS) に応じて異なります。

OPTION 句

OPTION 句では、`option=value` をカンマで区切ったリストで拡張オプションを設定できます。

各オプションの値に、等号とセミコロンは使用できません。データベースサーバは、入力されたオプションを、妥当性を検査しないですべて受け入れます。そのため、オプションのスペルを間違えたり、無効な値を入力したりしても、dbmlsync コマンドを実行して同期を行うまでエラーメッセージが表示されません。

同期ユーザ用に設定したオプションは、個々のサブスクリプション内で、または dbmlsync コマンドラインを使用して上書きできます。

`network-protocol`、`protocol-options`、`option` の設定は、他の文やコンテキストによって上書きできます。

この文を使用すると、オプションや他の情報が SQL Anywhere の ISYSSYNC システムテーブルに格納されます。SYSSYNC システムビューを表示する正当な権限を持っているユーザなら誰でも、パスワードや暗号化証明書が含まれている可能性のある情報を表示できます。このとき考えられるセキュリティ上の問題を回避するために、dbmlsync コマンドラインに関する情報を指定できます。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、TCP/IP 経由でサーバコンピュータ mlserver.mycompany.com と同期を行う、パスワード Sam を持つ Mobile Link ユーザ SSinger を作成します。ユーザ定義でのパスワードの使用には、セキュリティは適用されません。

```
CREATE SYNCHRONIZATION USER SSinger
TYPE http
ADDRESS 'host=mlserver.mycompany.com'
OPTION MobiLinkPwd='Sam';
```

関連情報

ALTER SYNCHRONIZATION USER 文 [Mobile Link] [704 ページ]
DROP SYNCHRONIZATION USER 文 [Mobile Link] [1073 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.97 CREATE TABLE 文

データベースに新しいテーブルを作成し、オプションでリモートサーバ上にテーブルを作成します。

構文

```
CREATE [ OR REPLACE ] [ GLOBAL TEMPORARY ] TABLE [ IF NOT EXISTS ] [owner.]table-
name
( { column-definition | table-constraint | pctfree | like-clause },... ) like-
clause | as-clause
[ { IN | ON } dbspace-name ]
[ ENCRYPTED ]
[ ON COMMIT { DELETE | PRESERVE } ROWS
   | NOT TRANSACTIONAL ]
[ AT location-string [ ESCAPE CHARACTER character ] ]
[ SHARE BY ALL ]
[ WITH [ NO ] DATA ]
```

```
column-definition :
column-name data-type
[ COMPRESSED ]
[ INLINE { inline-length | USE DEFAULT } ]
[ PREFIX { prefix-length | USE DEFAULT } ]
[ [ NO ] INDEX ]
[ [ NOT ] NULL ]
[ DEFAULT default-value | IDENTITY ]
[ column-constraint ... ]
[ table-element ... ]
```

```
default-value :
 special-value
| string
| global-variable
| [ - ] number
```

```
| ( constant-expression )
| ( sequence-expression )
| built-in-function( constant-expression )
| AUTOINCREMENT
| GLOBAL AUTOINCREMENT [ ( partition-size ) ]
```

```
special-value :
CURRENT DATABASE
| CURRENT DATE
| CURRENT TIME
| [ CURRENT ] TIMESTAMP
| CURRENT PUBLISHER
| CURRENT REMOTE USER
| [ CURRENT ] USER
| [ CURRENT ] UTC TIMESTAMP
| EXECUTING USER
| INVOKING USER
| LAST USER
| NULL
| PROCEDURE OWNER
| SESSION USER
```

```
column-constraint :
[ CONSTRAINT constraint-name ] {
    UNIQUE [ CLUSTERED ]
  | PRIMARY KEY [ CLUSTERED ] [ ASC | DESC ]
  | REFERENCES table-name [ ( column-name ) ]
      [ MATCH [ UNIQUE ] { SIMPLE | FULL } ]
      [ action-list ] [ CLUSTERED ]
  | CHECK ( condition )
    }
| COMPUTE ( expression )
```

```
table-element :
 { column-definition
 | table-constraint-definition
 | like-clause }
```

```
table-constraint :
[ CONSTRAINT constraint-name ] {
      UNIQUE [ CLUSTERED ] ( column-name [ ASC | DESC ],... )
  | PRIMARY KEY [ CLUSTERED ] ( column-name [ ASC | DESC ],... )
  | CHECK ( condition )
  | foreign-key-constraint
}
```

```
foreign-key-constraint :
[ NOT NULL ] FOREIGN KEY [ role-name ]
    [ ( column-name [ ASC | DESC ],... ) ]
   REFERENCES table-name
   [ ( column-name,... ) ]
   [ MATCH [ UNIQUE] { SIMPLE | FULL } ]
   [ action-list ] [ CHECK ON COMMIT ] [ CLUSTERED ] [ FOR OLAP WORKLOAD ]
```

```
action-list :
[ ON UPDATE action ]
[ ON DELETE action ]
```

```
action :
CASCADE
| SET NULL
```

```
| SET DEFAULT
| RESTRICT
```

```
like-clause :
LIKE source-table [ like-option,... ]
```

```
like-option :
{ INCLUDING | EXCLUDING } option [ ,... ]
```

```
option :
{ IDENTITY
| DEFAULTS
| CONSTRAINTS
| INDEXES
| PRIMARY KEY
| FOREIGN KEYS
| COMMENTS
| STORAGE
| ALL }
```

```
as-clause :
[ (column-name,... ) ] AS ( select-statement )
```

```
location-string :
 remote-server-name.[db-name].[owner].object-name
| remote-server-name;[db-name];[owner];object-name
```

```
pctfree : PCTFREE percent-free-space
```

```
percent-free-space : integer
```

パラメータ

OR REPLACE EXISTS clause

テーブル定義がすでに存在する場合は置き換えられ、エラーは返されません。IF NOT EXISTS が指定されている場合、この句を指定することはできません。

LIKE clause

`like-option` のデフォルトは、すべてのオプションを除外 (EXCLUDING) します。`source-table` がビューである場合、`like-option` は適用されていなと無視されます。

元のテーブルから次のカラム属性のいずれかを新しいテーブルに含めるには、INCLUDE を指定します。

IDENTITY デフォルトのオートインクリメント情報を含めます。
DEFAULTS オートインクリメント以外のデフォルトの値式を含めます。
CONSTRAINTS カラムとテーブルの検査、および一意性制約を含めます。
INDEXES プライマリキーと外部キー以外のインデックスを含めます。
PRIMARY KEY プライマリキーを含めます。
FOREIGN KEY 外部キーを含めます。

COMMENTS カラムについてのコメントを含めます。

STORAGE 圧縮設定を含めます。

AS clause SELECT 文に基づくテーブルを複製するには、この句を使用します。

IN clause

この句は、ベーステーブルが格納される DB 領域を指定するときに使用します。この句が指定されていない場合、ベーステーブルは default_dbspace オプションで指定された DB 領域に作成されます。

テンポラリテーブルはテンポラリ DB 領域だけに作成できます。GLOBAL TEMPORARY テーブルを作成するときに IN を指定すると、テーブルはテンポラリ DB 領域に作成されます。ユーザ定義の DB 領域を指定すると、エラーが返されます。

ENCRYPTED clause

ENCRYPTED 句は、テーブルの暗号化が必要なときに使用します。テーブルを暗号化する場合、データベースの作成時にテーブルの暗号化を有効にする必要があります。その後、データベースの作成時に指定された暗号化キーとアルゴリズムを使用してテーブルが暗号化されます。

ON COMMIT clause

ON COMMIT 句はテンポラリテーブル用にのみ使用します。デフォルトで、テンポラリテーブルのローは COMMIT のときに削除されます。SHARE BY ALL 句を指定する場合、ON COMMIT PRESERVE ROWS または NOT TRANSACTIONAL を指定する必要があります。

NOT TRANSACTIONAL clause

グローバルテンポラリテーブルを作成するときに NOT TRANSACTIONAL 句を使用できます。NOT TRANSACTIONAL を使用して作成されたテーブルは、COMMIT または ROLLBACK の影響を受けません。SHARE BY ALL 句を指定する場合、ON COMMIT PRESERVE ROWS または NOT TRANSACTIONAL を指定する必要があります。

AT clause

`location-string` に指定された異なるサーバにリモートテーブルを作成し、そのリモートテーブルにマッピングするプロキシテーブルを現在のデータベース上に作成します。AT 句では、`location-string` 内のフィールドデリミタとしてセミコロン (;) を使用できます。セミコロンがない場合は、ピリオドがフィールドデリミタになります。この構文を使用すると、データベースと所有者の各フィールドにファイル名と拡張子を使用できます。

たとえば次の文は、テーブル proxy_Customers を、架空の Microsoft Access データベースである MyAccessDB の新しいテーブル Customers にマッピングします。

```
CREATE TABLE proxy_Customers
(
    ID            INTEGER CONSTRAINT PRIMARY KEY,
    ClientName    TEXT,
    ClientAddress TEXT,
    Telephone     TEXT
)
AT 'MyAccessDB;;;Customers';
```

AT 句内の文字列に、波括弧で囲んだローカル変数名またはグローバル変数名を入れることができます ({`variable-name`})。SQL 変数名は、CHAR、VARCHAR、または LONG VARCHAR のデータ型にする必要があります。たとえば、`'access;{@myfile};;a1'` を含む AT 句は、`@myfile` が SQL 変数で、`@myfile` 変数の現在の内容がプロキシテーブルの作成時に置き換えられることを示します。

外部キー定義は、リモートテーブルでは無視されます。リモートテーブルを参照するローカルテーブルでの外部キー定義も無視されます。プライマリキー定義は、リモートサーバにサポートされている場合、そのデータベースサーバに送信されます。

CREATE TABLE または CREATE EXISTING 文のいずれかを使用してプロキシテーブルを作成するとき、AT 句は次の要素で構成されるロケーション文字列を含みます。

- リモートサーバの名前
- リモートカタログ
- リモート所有者またはスキーマ
- リモートテーブル名

ピリオドまたはセミコロンを使用して、ロケーション文字列を区切ります。ロケーション文字列はまた、データベースサーバがロケーション文字列を評価する際に展開される変数名を含むことができます。ロケーション文字列内の変数名は波括弧で囲みます。リモートサーバ名、カタログ名、所有者名、スキーマ名、またはテーブル名の一部として、ピリオド、セミコロン、波括弧などを用いることは非常にまれです。しかしながら、ロケーション文字列の中で、これらの区切り文字の 1 つまたはすべてを文字どおりに解釈すべき状況もあります。ESCAPE CHARACTER 句を使用すると、アプリケーションはロケーション文字列内のこれらの区切り文字をエスケープできます。

> **注記**
>
> ESCAPE 句は、ロケーション句内で区切り文字をエスケープする場合のみ必要です。一般的に、プロキシテーブルの作成時は ESCAPE 句を省略できます。エスケープ文字には、任意の 1 バイト文字を指定できます。

SHARE BY ALL clause

この句を使用できるのは、グローバルテンポラリテーブルを作成して、データベースに対するすべての接続でテーブルを共有する場合のみです。SHARE BY ALL 句を指定する場合、ON COMMIT PRESERVE ROWS または NOT TRANSACTIONAL を指定する必要があります。

WITH [NO] DATA clause この句は、AS 句を使用して元のテーブルから新しいテーブルにデータをコピーするかどうかを指定した場合にのみ使用します。

IF NOT EXISTS clause

指定された名前のテーブルがすでに存在する場合、変更は行われず、エラーも返されません。OR REPLACE が指定されている場合、この句を指定することはできません。

column-definition

テーブル内のカラムを定義します。次は、カラム定義の一部を示します。

column-name

カラム名は識別子です。同じテーブル内の 2 つのカラムが同じ名前を持つことはできません。

data-type

カラムに格納されているデータ型。データ型を明示的に定義するか、%TYPE 属性を指定して、データ型を別のテーブルまたはビュー内のカラムのデータ型に設定します。%TYPE 属性を指定する場合、%TYPE 仕様内で参照されるオブジェクト上のデフォルト値、制約、および NULL の使用の可否などの他の属性は、継承される定義の一部ではなく、個別に指定する必要があります。

COMPRESSED clause

カラムを圧縮します。

INLINE and PREFIX clauses

INLINE 句は、ローに格納する最大 BLOB サイズをバイト単位で指定します。INLINE 句で指定された値以下のサイズの BLOB がローに格納されます。INLINE 句で指定された値を超えるサイズの BLOB は、ロー外のテーブル拡張ページに格納されます。また、BLOB のサイズが INLINE 句で指定された値より大きい場合、BLOB の先頭から数バイト分を複製

してローに保持することができます。PREFIX 句は、ローに保持されるバイト数を指定します。ローの受け入れと拒否の判断に BLOB のプレフィクスバイトを必要とする要求のパフォーマンスは、PREFIX 句によって向上する可能性があります。

圧縮されたカラムのプレフィクスデータは圧縮されずに格納されるため、要求を満たすために必要なすべてのデータがプレフィクス内に格納されている場合は、解凍は必要ありません。

INLINE と PREFIX のどちらも指定しない場合、または USE DEFAULT を指定している場合、デフォルト値は次のように適用されます。

- CHAR、NCHAR、LONG VARCHAR などの文字データ型のカラムの場合、INLINE のデフォルト値は 256 で、PREFIX のデフォルト値は 8 です。
- BINARY、LONG BINARY、VARBINARY、BIT、VARBIT、LONG VARBIT、BIT VARYING、UUID などのバイナリデータ型のカラムの場合、INLINE のデフォルト値は 256 で、PREFIX のデフォルト値は 0 です。

  注記

  デフォルト以外の設定が必要な特殊な環境でない限り、デフォルト値を使用するようにしてください。デフォルト値は、パフォーマンスとディスク領域の要件のバランスを取って選択されています。たとえば、INLINE に大きな値を設定し、すべての BLOB をインラインで格納するようにした場合、ローの処理パフォーマンスは低下することがあります。また、PREFIX の値を大きくしすぎると、BLOB の一部を複製したプレフィクスデータのために、BLOB の格納に必要なディスク領域のサイズが増えることになります。

  値の 1 つのみを指定する場合、その他の値は指定した値と競合しない最大の値に自動的に設定されます。INLINE 値と PREFIX 値のどちらも、データベースページサイズを超えるサイズを指定できません。また、ローデータの格納に使用できないテーブルページには、小さいサイズですが予約済みのオーバヘッドがあります。そのため、データベースのページサイズに近い INLINE 値を指定すると、やや少ないバイト数がインラインで格納される可能性があります。

INDEX and NO INDEX clauses

BLOB の格納時に (文字型またはバイナリ型のみ)、BLOB サイズの内部しきい値 (約 8 データベースページ) を超える挿入値に対して BLOB インデックスを作成する場合は、INDEX を指定します。これはデフォルトの動作です。

BLOB インデックスは、BLOB 内のランダムアクセス検索が必要なときにパフォーマンスを改善する可能性があります。ただし、ランダムアクセスの必要がない画像ファイルやマルチメディアファイルなど、BLOB 値の種類によっては、BLOB インデックスをオフにするとパフォーマンスが改善されることがあります。カラムの BLOB インデックスをオフにするには、NO INDEX を指定します。

注記

BLOB インデックスは、テーブルインデックスとは異なります。テーブルインデックスは、1 つ以上のカラムの値のインデックスとして作成されます。

NULL and NOT NULL clauses

NULL が指定されると、カラムで NULL 値が許可されるようになります。これはデフォルトの動作です。この設定は、allow_nulls_by_default データベースオプションを使用して制御します。

デフォルトで、BIT として宣言されているカラムは NULL 入力不可ですが、明示的に NULL 入力可にすることができます。

NOT NULL が指定されると、NULL 値は許可されません。

カラムが UNIQUE 制約または PRIMARY KEY 制約の一部である場合、NULL 句が指定されていても、そのようなカラムに NULL を含めることはできません。

DEFAULT clause

DEFAULT 値を指定する場合、カラムの値を指定しない INSERT 文のカラムの値としてこのデフォルト値が使用されます。INSERT 文はカラムの値を指定しません。DEFAULT 値を指定しない場合、これは DEFAULT NULL と同じです。

DEFAULT に指定できる値を次に示します。

constant-expression

DEFAULT 句では、データベースオブジェクトを参照していない定数式を使用できます。その結果、GETDATE や DATEADD などの関数を使用できます。式が関数または単純な値でない場合、括弧で囲みます。

global-variable

グローバル変数。

sequence-expression

データベース内のシーケンスの現在の値または次の値に、DEFAULT を設定できます。

string

文字列値

AUTOINCREMENT

AUTOINCREMENT を使用する場合、カラムは整数データ型の 1 つ、または真数値型にします。

テーブルに挿入する場合、AUTOINCREMENT カラムの値を指定しないと、カラム内の任意の値より大きいユニーク値が生成されます。INSERT によって、カラムの現在の最大値よりも大きなカラム値を指定すると、その値は挿入され、以降の挿入時に開始ポイントとして使用されます。

ローを削除しても AUTOINCREMENT カウンタは減りません。ローの削除によって作成されたギャップは、挿入を行うときに明示的に割り当てることによってのみ埋めることができます。最大値未満のカラム値を明示的に挿入した後、明示的に割り当てられていない次のローを、以前の最大値より 1 大きい値を使ってオートインクリメントさせます。

カラムに直前に挿入された値は、グローバル変数 @@identity を調べることによって確認できます。

AUTOINCREMENT 値は、SYSTABCOL システムビューの max_identity カラムのデータ型に応じて、符号付き 64 ビット整数として保持されます。生成された次の値が、AUTOINCREMENT が割り当てられたカラムに格納できる最大値を超えた場合は、NULL が返されます。NULL を入力できないように宣言されたカラムであると (プライマリキーのカラムである場合など)、SQL エラーが生成されます。

カラムに使用する次の値は、sa_reset_identity プロシージャを使用してリセットできます。

GLOBAL AUTOINCREMENT

このデフォルトは、Mobile Link 同期環境または SQL Remote レプリケーションにおいて複数のデータベースを使用するときのために用意されたものです。

この句は AUTOINCREMENT と同じですが、ドメインはパーティションに分割されます。各分割には同じ数の値が含まれます。データベースの各コピーにユニークなグローバルデータベース ID 番号を割り当てます。データベースサーバでは、データベースのデフォルト値は、そのデータベース番号でユニークに識別された分割からのみ設定されます。

AUTOINCREMENT キーワードの直後に括弧で分割サイズを指定できます。この分割サイズには任意の正の整数を設定できますが、通常、分割サイズは、サイズの値がすべての分割で不足しないように選択されます。

カラムの型が BIGINT または UNSIGNED BIGINT である場合、デフォルトの分割サイズは $2^{32} = 4294967296$ です。それ以外の型のカラムの場合、デフォルトの分割サイズは $2^{16} = 65536$ です。特に、カラムの型が INT または

BIGINT ではない場合は、これらのデフォルト値が適切ではないことがあるため、分割サイズを明示的に指定するのが最も賢明です。

このデフォルトを使用する場合、各データベース内のパブリックオプション global_database_id は、ユニークな正の整数に設定します。この値は、データベースをユニークに識別し、デフォルト値の割り当て元の分割を示します。使用できる値の範囲は、np + 1 から p(n + 1) です。ここで、n はパブリックオプション global_database_id の値を表し、p は分割サイズを表します。たとえば、分割サイズを 1000、global_database_id を 3 に設定すると、範囲は 3001 ～ 4000 になります。

前の値が p(n + 1) 未満であれば、このカラム内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になります。カラムに値が含まれていない場合、最初のデフォルト値は np + 1 です。デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けません。つまり、np + 1 より小さいか、p(n + 1) より大きい数には影響されません。Mobile Link または SQL Remote 経由で別のデータベースからレプリケートした場合、このような値になることがあります。

カラムに直前に挿入された値は、グローバル変数 @@identity を調べることによって確認できます。

GLOBAL AUTOINCREMENT 値は、SYSTABCOL システムビューの max_identity カラムのデータ型に応じて、符号付き 64 ビット整数として保持されます。NULL 値は、分割で値が不足したときにも生成されます。NULL を入力できないように宣言されたカラムであると (プライマリキーのカラムである場合など)、SQL エラーが生成されます。この場合には、別の分割からデフォルト値を選択できるように、データベースに global_database_id の新しい値を割り当ててください。未使用の値が残り少ないことを検出し、このような状態を処理するには、GlobalAutoincrement タイプのイベントを作成します。

public オプション global_database_id は、負の値に設定できないため、選択された値は常に正になります。ID 番号の最大値を制限するのは、カラムデータ型と分割サイズだけです。

public オプション global_database_id がデフォルト値の 2147483647 に設定されると、NULL 値がカラムに挿入されます。NULL 値が許可されていない場合に、ローの挿入を試みるとエラーが発生します。

カラムに使用する次の値は、sa_reset_identity プロシージャを使用してリセットできます。

special-value

DEFAULT 句では、複数の特別値の 1 つを使用します (たとえば、CURRENT DATE)。これには次のもの特別値が含まれますが、これに限定されません。

[CURRENT] TIMESTAMP

テーブル内の各ローが最後に変更された日付を示すことができます。カラムの宣言に DEFAULT TIMESTAMP が指定されている場合は、ローを挿入するとタイムスタンプのデフォルト値が割り付けられます。この値は、ローが更新されるたびに現在の日付と時刻に基づいて更新されます。

挿入されたローにタイムスタンプのデフォルト値を割り付け、そのローが更新されてもタイムスタンプを更新しない場合は、DEFAULT TIMESTAMP の代わりに DEFAULT CURRENT TIMESTAMP を使用します。

DEFAULT TIMESTAMP で宣言されたカラムにはユニークな値が入ります。これにより、アプリケーションは、ほぼ同時に行われた同じローの更新を検出できます。現在の TIMESTAMP 値が直前の値と同じ場合は、default_timestamp_increment オプションの値が加えられます。

default_timestamp_increment オプションに基づいて、TIMESTAMP 値を自動的にトランケートできます。これは、記録されるタイムスタンプ値の精度が低い他のデータベースソフトウェアとの互換性を維持する場合に便利です。

グローバル変数 @@dbts は、DEFAULT TIMESTAMP を使用するカラムの最後に生成された値を表す TIMESTAMP 値を返します。

データベースでシミュレートされたタイムゾーンが使用されている場合、シミュレートされたタイムゾーンを使用してこれらの値が計算されます。

[CURRENT] UTC TIMESTAMP

テーブル内の各ローが最後に変更された日付を示すことができます。カラムの宣言に DEFAULT UTC TIMESTAMP が指定されている場合は、ローを挿入するとタイムスタンプのデフォルト値が割り付けられます。この値は、ローが更新されるたびに現在の協定世界時 (UTC: Coordinated Universal Time) に基づいて更新されます。

挿入されたローにタイムスタンプのデフォルト値を割り付け、そのローが更新されてもタイムスタンプを更新しない場合は、DEFAULT UTC TIMESTAMP の代わりに DEFAULT CURRENT UTC TIMESTAMP を使用します。

このデフォルトの動作は TIMESTAMP や CURRENT TIMESTAMP と同じですが、日付と時刻が協定世界時 (UTC: Coordinated Universal Time) になる点が異なります。

グローバル変数 @@dbts は、CURRENT UTC TIMESTAMP の使用時には更新されません。

LAST USER

LAST USER は、ローを最後に変更したユーザのユーザ ID です。

LAST USER は文字データ型のカラムでデフォルト値として使用できます。

INSERT の場合、このデフォルトは CURRENT USER と同じ効果があります。

UPDATE では、LAST USER のデフォルトを持つカラムが明示的に変更されていなければ、現在のユーザ名に変更されます。

DEFAULT TIMESTAMP または DEFAULT UTC TIMESTAMP とともに使用すると、LAST USER のデフォルトを使用して、ローを最後に変更したユーザと日時の両方を記録できます (ただし、別々のカラムに記録されます)。

IDENTITY clause

IDENTITY は、DEFAULT AUTOINCREMENT の使用に対する Transact-SQL 互換の代替手段です。IDENTITY を指定して定義されたカラムは DEFAULT AUTOINCREMENT として実装されます。

column-constraint and table-constraint clauses

カラム制約とテーブル制約によってデータベース内のデータの整合性が保証されます。整合性制約の違反を起こす文は、実行が完了しません。このような文がエラー検出の前に行った変更は取り消され、エラーがレポートされます。作成できる制約のクラスは、検査制約と **RI (**参照整合性**)** 制約の 2 つです。検査制約は、データベースに配置されているカラム値が満たす必要がある条件を指定するときに使用されます。RI 制約は、データに一意性の要件を指定するだけでなく、保守が必要な複数テーブルのデータ間に関係を確立します。

RI 制約には、プライマリキー、外部キー、一意性制約という 3 種類があります。RI 制約 (プライマリキー、外部キー、または一意性制約) を作成すると、データベースサーバは、制約のキーを構築するカラムにインデックスを暗黙的に作成することで、制約を実行します。インデックスは指定した制約のキーに作成されます。キーは、順序付きリストのカラムと各カラムのシーケンス値 (ASC/DESC) で構成されます。

制約はカラムまたはテーブルに指定できます。カラムの制約はテーブル内の 1 つのカラムを指しますが、テーブルの制約はテーブル内の 1 つ以上のカラムを指します。

PRIMARY KEY clause

プライマリキーはテーブルの各ローをユニークに定義します。プライマリキーは 1 つ以上のカラムで構成されます。1 つのテーブルに複数のプライマリキーが存在することがあります。`column-constraint` 句に PRIMARY KEY を

指定すると、そのカラムはテーブルのプライマリキーになります。table-constraint で PRIMARY KEY 句を使用して、指定した順序で組み合わせてテーブルのプライマリキーを構築する 1 つ以上のカラムを指定します。

プライマリキーのカラムの順序は、カラムの順序と一致する必要はありません。つまり、プライマリキーのカラムは、ローの物理的な順序と同じになる必要はありません。また、重複するカラム名を指定することはできません。

プライマリキーを作成すると、キーのインデックスが自動的に作成されます。各カラムに ASC (昇順) または DESC (降順) を指定することで、インデックスの値の順序を指定できます。また、CLUSTERED キーワードを指定して、インデックスをクラスタ化するかどうかを指定することもできます。

プライマリキーに含まれるカラムには NULL を使用できません。テーブル内の各ローは、ユニークなプライマリキー値を持ちます。

プライマリキーには、FLOAT や DOUBLE などの概数値データ型を使用しないことをおすすめします。概数値データ型は、算術演算後の丸め誤差がでます。

プライマリキーには空間カラムは設定できません。

FOREIGN KEY clause

外部キーは、カラムのセットの値がプライマリキーの値、または別のテーブルの一意性制約 (プライマリテーブル) と一致するよう制限します。たとえば、外部キー制約を使って、請求書テーブルの顧客番号が顧客テーブルの顧客番号と確実に一致するようにできます。

外部キーカラムの順序は、テーブルのカラム順を反映する必要はありません。

外部キー指定では、カラム名の重複は許可されません。

外部キーを作成すると、キーのインデックスが自動的に作成されます。各カラムに ASC (昇順) または DESC (降順) を指定することで、インデックスの値の順序を指定できます。また、CLUSTERED キーワードを指定して、インデックスをクラスタ化するかどうかを指定することもできます。

グローバルテンポラリテーブルは、ベーステーブルを参照する外部キーを持つことはできません。また、ベーステーブルも、グローバルテンポラリテーブルを参照する外部キーを持つことはできません。

外部キーを存在しないカラムに追加しようとすると、そのカラムが自動的に作成されます。

NOT NULL clause

外部キーカラムを NULL 入力不可にします。外部キー内の NULL は、外部テーブルのこのローに該当するローがプライマリテーブル内にないことを意味します。

role-name clause

役割名は外部キーの名前です。役割名の主な機能は、同じテーブルに対する 2 つの外部キーを区別することです。役割名を指定しない場合、役割名は以下のように設定されます。

1. テーブル名と同じ役割名を含む外部キーが存在しない場合、テーブル名が役割名として割り当てられます。
2. テーブル名がすでに使用されている場合、役割名は、0 が埋め込まれた、テーブルに固有の 3 桁の数字と結合されたテーブル名になります。

REFERENCES clause

外部キー制約は、REFERENCES カラム制約または FOREIGN KEY テーブル制約を使用して実装できます。このとき、1 つ以上のカラムを指定できます。REFERENCES カラム制約に column-name を指定する場合、一意性制約またはプライマリキーの制約を受ける、プライマリテーブルのカラム名を指定します。また、この制約は、その 1 カラムだけで構成します。column-name を指定しない場合、外部キーはプライマリテーブルに含まれる単一のプライマリキーのカラムを参照します。

MATCH clause

MATCH 句は、孤立したローの内容と参照整合性違反の内容を管理できるようにすることによって、複数カラムの外部キーを使用した場合に何が一致と見なされるかを決定します。MATCH 句では、また、キーの一意性を指定することで、一意性を別に宣言する必要がなくなります。

次に、指定できる MATCH タイプを示します。

MATCH [UNIQUE] SIMPLE

外部キーテーブルのローに一致が発生するのは、すべてのカラム値がプライマリキーテーブルのローにある対応するカラム値と一致する場合です。外部キーテーブルでローが孤立するのは、外部キーの少なくとも 1 つのカラム値が NULL である場合です。

MATCH SIMPLE はデフォルトの動作です。

UNIQUE キーワードを指定すると、NULL 以外のキー値に対して、参照テーブルで一致が 1 つのみになります。

MATCH [UNIQUE] FULL

外部キーテーブルのローに一致が発生するのは、いずれの値も NULL ではなく、値がプライマリキーテーブルのローにある対応するカラム値と一致する場合です。ローが孤立するのは、外部キーのすべてのカラム値が NULL の場合です。

UNIQUE キーワードを指定すると、NULL 以外のキー値に対して、参照テーブルで一致が 1 つのみになります。

UNIQUE clause

`column-constraint` 句では、UNIQUE 制約はカラム値をユニークにする必要があることを示します。`table-constraint` 句では、UNIQUE 制約は、テーブル内の各ローをユニークに識別する 1 つ以上のカラムを指定します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルには、複数の一意性制約を設定できます。

UNIQUE 制約はユニークインデックスとは異なります。ユニークインデックスのカラムは NULL でもかまいません。一方、UNIQUE 制約のカラムは NULL にはなりません。また、外部キーは、プライマリキーまたは UNIQUE 制約を参照できますが、ユニークインデックスは参照できません。ユニークインデックスは NULL の複数のインスタンスを含むことがあるからです。

UNIQUE 制約のカラムは、任意の順序で指定できます。また、各カラムに ASC (昇順) または DESC (降順) を指定することで、自動的に作成された対応するインデックスの値の順序を指定できます。ただし、カラム名を重複して指定することはできません。

一意性制約には、カラムに FLOAT や DOUBLE などの概数値データ型を使用しないことをおすすめします。概数値データ型は、算術演算後の丸め誤差がでます。

また、CLUSTERED キーワードを指定して、制約をクラスタ化するかどうかを指定することもできます。

action clause (CASCADE, SET NULL, SET DEFAULT, RESTRICT)

更新および削除での参照整合性アクションは、外部キーに影響を及ぼす場合があります。

CASCADE

ON UPDATE では、更新されたプライマリキーを参照しているすべての外部キーが、新しい値に更新されます。ON DELETE では、削除されたプライマリキーを参照しているすべての外部キーのローが削除されます。

SET NULL

変更されたプライマリキーを参照しているすべての外部キーを NULL に設定します。

SET DEFAULT

変更されたプライマリキーを参照しているすべての外部キーを、そのカラムのデフォルト値 (テーブル定義で指定された値) に設定します。

RESTRICT

参照されているプライマリキーの値をユーザが変更しようとした場合、エラーを生成してその変更を防止します。これが参照整合性アクションのデフォルトです。

CHECK clause

この制約で、任意の条件を検証できます。たとえば、CHECK 制約を使うと、Sex というカラムには M または F の値だけが確実に入ります。

テーブル内の 2 つ以上のカラム間の関係 (カラム A はカラム B 未満である必要があるなど) が関与する CHECK 制約の作成が必要な場合は、代わりにテーブル制約を定義します。

テーブルのどのローも CHECK 制約に違反することはできません。INSERT または UPDATE 文によってローが制約に違反する場合、操作は許可されず、この文の処理結果は取り消されます。変更は、CHECK 制約条件の評価結果が FALSE の場合にのみ拒否されます。CHECK 制約条件の評価結果が TRUE または UNKNOWN の場合、変更は許可されます。

COMPUTE clause

COMPUTE 句は、`column-constraint` 句にのみ使用します。

カラムが COMPUTE 句を使って作成される場合、すべてのローの値は式で提供されます。この制約を付けて作成されたカラムは、読み込み専用カラムです。この値は、ローが修正されたときにデータベースサーバによって変更されます。COMPUTE 式は、非決定的な値を返しません。たとえば、CURRENT TIMESTAMP などの特別値や非決定的関数は指定できません。COMPUTE 式が非決定的な値を返したとしても、それをクエリに含まれる式との一致に使用することはできません。

リモートテーブルでは COMPUTE 句は無視されます。

計算カラムの値を変更しようとする UPDATE 文は、そのカラムに対応するトリガを起動します。

CHECK ON COMMIT clause

CHECKONCOMMIT 句は wait_for_commit データベースオプションを上書きします。この句を指定すると、データベースサーバは COMMIT を待ってから外部キーに対する RESTRICT アクションをチェックします。CHECK ON COMMIT 句を指定することによって、外部キーチェックの遅延が発生しますが、CASCADE、SET NULL、SET DEFAULT、検査制約などの他のアクションの遅延は発生しません。

FOR OLAP WORKLOAD clause

外部キー定義の REFERENCES 句に FOR OLAP WORKLOAD を指定すると、データベースサーバは特定の最適化を実行し、キーに関する統計情報を収集して、OLAP の負荷に関するパフォーマンスを向上させることができます。特に、optimization_workload を OLAP に設定すると有効です。

PCTFREE clause

各テーブルページに確保する空き領域の割合を指定します。空き領域は、データが更新されたときにローのサイズが増えた場合に使用されます。テーブルページに空き領域がない場合は、ページのローのサイズが増えるたびに、ローを複数のテーブルページに分割することが必要になり、ローの断片化が発生します。また、パフォーマンス低下の可能性があります。

値 `percent-free-space` は 0 ～ 100 の整数です。値を 0 に指定すると、各ページに空き領域が残らず、各ページが完全にパックされます。大きい値を指定すると、各ローが単独でページに挿入されます。PCTFREE が設定されない場

合、または削除された場合、データベースのページサイズに応じたデフォルトの PCTFREE 値が適用されます (ページサイズが 4 KB 以上の場合は 200 バイト)。PCTFREE の値は、ISYSTAB システムテーブルに格納されます。

備考

CREATE TABLE 文は新しいテーブルを作成します。所有者を指定することにより、別のユーザに対するテーブルを作成できます。GLOBAL TEMPORARY を指定すると、テーブルはテンポラリテーブルと見なされます。指定しないと、テーブルはベーステーブルとなります。

CREATE TABLE...LIKE 構文は、別のテーブル定義に直接基づく新規のテーブルを作成します。カラム、制約、および LIKE 句を追加してテーブルを複製したり、SELECT 文を使用してテーブルを作成したりすることもできます。

CREATE TABLE 文のテーブル名の前にシャープ記号 (#) を付けてテーブルを作成すると、テンポラリテーブルが宣言されます。テンポラリテーブルは現在の接続でのみ使用できます。シャープ記号 (#) を使用して作成されたテンポラリテーブルは、ON COMMIT PRESERVE ROWS 句を使用して作成されたテーブルと同じです。

同じスコープ内にある 2 つのローカルテンポラリテーブルは、同じ所有者や名前にはできません。ベーステーブルと同じ所有者と名前のローカルテンポラリテーブルを作成すると、ローカルテンポラリテーブルのスコープが終了した時点で、そのベーステーブルは、その接続内でのみ参照できるようになります。接続では、既存のテンポラリテーブルと同じ所有者と名前のベーステーブルを作成できません。

カラムはデフォルトで NULL を許容しています。この設定は、allow_nulls_by_default データベースオプションを使用して制御できます。

権限

ユーザ本人が所有するテーブルを作成するには、CREATE TABLE システム権限が必要です。他のユーザが所有するテーブルを作成するには、CREATE ANY TABLE または CREATE ANY OBJECT のシステム権限が必要です。

ユーザ本人が所有するプロキシテーブルを作成するには、CREATE PROXY TABLE システム権限が必要です。他のユーザが所有するプロキシテーブルを作成するには、CREATE ANY TABLE または CREATE ANY OBJECT のシステム権限が必要です。

既存のテーブルを置き換えるには、そのテーブルの所有者であるか、または次のいずれかの権限が必要です。

- CREATE ANY TABLE および DROP ANY TABLE システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY TABLE システム権限。

関連する動作

オートコミット (グローバルテンポラリテーブルを作成しているときにも適用)。

標準

ANSI/ISO SQL 標準

CREATE TABLE はコア機能ですが、ソフトウェアでサポートされている一部のコンポーネントはオプションの SQL 言語機能です。これらの機能のサブセットを次に示します。

- テンポラリテーブルのサポートは、SQL 言語機能 F531 です。
- IDENTITY カラムのサポートは SQL 機能 T174 ですが、ソフトウェアでは標準とは少し異なる構文が使用されます。
- 外部キー制約のサポートには、SQL 言語機能 T191 "Referential action: RESTRICT"、F741 "Referential MATCH types"、F191 "Referential delete actions"、F701 "Referential update actions" が含まれています。このソフトウェアでは、MATCH PARTIAL はサポートされていません。
  このソフトウェアでは、SQL 言語機能 T591 (UNIQUE constraints of possibly null columns) はサポートされていません。このソフトウェアでは、PRIMARY KEY 制約または UNIQUE 制約の一部であるすべてのカラムを NOT NULL と宣言してください。

次に示す CREATE TABLE のコンポーネントは、標準にありません。

- { IN | ON } `dbspace-name` 句
- ENCRYPTED 句、NOT TRANSACTIONAL 句、SHARE BY ALL 句
- カラム定義の COMPRESSED, INLINE 句、PREFIX 句、NO INDEX 句
- AUTOINCREMENT、GLOBAL AUTOINCREMENT、CURRENT DATABASE、CURRENT REMOTE USER、CURRENT UTC TIMESTAMP、ほとんどの特別値などの、実装依存のさまざまな DEFAULT 値。また、SEQUENCE ジェネレータを参照する DEFAULT 句も、標準にありません。
- MATCH UNIQUE の指定
- PRIMARY KEY 句または FOREIGN KEY 句での指定値のソート方法 (ASC または DESC)
- 参照先テーブルの PRIMARY KEY 句での指定と異なる順序で FOREIGN KEY カラムを指定する機能

例

次の文は、file_name と file_contents という 2 つのカラムを持つテーブル file_table を作成します。contents カラムは LONG BINARY であり、圧縮されています。

```
CREATE TABLE file_table (
  file_name VARCHAR(255),
  file_contents LONG BINARY COMPRESSED
);
```

次の例は、図書データベース用にテーブルを作成し、図書情報を保持します。

```
CREATE TABLE library_books (
   -- NOT NULL is assumed for primary key columns
   isbn CHAR(20) PRIMARY KEY,
   copyright_date DATE,
   title CHAR(100),
   author CHAR(50),
   -- column(s) corresponding to primary key of room
   -- are created automatically
);
```

次の例では、図書データベース用にテーブルを作成して、貸し出し図書の情報を保持します。date_borrowed のデフォルト値は、エントリが作成される日に本が貸し出されることを示します。date_returned カラムは、本が返却されるまでは NULL です。

```
CREATE TABLE borrowed_book (
   date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
   date_returned DATE,
   book CHAR(20)
   REFERENCES library_books (isbn),
   -- The check condition is UNKNOWN until
   -- the book is returned, which is allowed
CHECK( date_returned >= date_borrowed )
);
```

次の例は、販売データベース用にテーブルを作成し、注文と注文項目情報を保持します。

```
CREATE TABLE Orders (
   order_num INTEGER NOT NULL PRIMARY KEY,
   date_ordered DATE,
   name CHAR(80)
);
CREATE TABLE Order_item (
   order_num INTEGER NOT NULL,
   item_num SMALLINT NOT NULL,
   PRIMARY KEY ( order_num, item_num ),
   -- When an order is deleted, delete all of its
   -- items.
   FOREIGN KEY ( order_num )
   REFERENCES Orders ( order_num )
   ON DELETE CASCADE
);
```

次の例は、架空のリモートサーバ SERVER_A にテーブル t1 を作成し、このリモートテーブルにマッピングするプロキシテーブル t1 を作成します。

```
CREATE TABLE t1
( a INT,
  b CHAR(10) )
AT 'SERVER_A.db1.joe.t1';
```

次の例は、2 つのテーブル Table1 と Table2 を作成し、外部キーを Table2 に追加し、値を Table1 に挿入します。最後の文は、値を Table2 に挿入しようとします。挿入しようとした値は Table1 と単純一致しないため、エラーが返されます。

```
CREATE TABLE Table1 ( P1 INT, P2 INT, P3 INT, P4 INT, P5 INT, P6 INT, PRIMARY KEY
( P1, P2 ) );
CREATE TABLE Table2 ( F1 INT, F2 INT, F3 INT, PRIMARY KEY ( F1, F2 ) );
ALTER TABLE Table2
   ADD FOREIGN KEY fk2( F1,F2 )
   REFERENCES Table1( P1, P2 )
   MATCH SIMPLE;
INSERT INTO Table1 (P1, P2, P3, P4, P5, P6) VALUES ( 1,2,3,4,5,6 );
INSERT INTO Table2 (F1,F2) VALUES ( 3,4 );
```

次の文は、キーの一部 (すべてではない) のカラムが NULL の場合に、MATCH SIMPLE と MATCH SIMPLE UNIQUE で、複数カラムの外部キーの処理方法がどのように異なるかを示します。2 番目のカラムの値がユニークではないため、この文は失敗します。

```
CREATE TABLE pt( pk INT PRIMARY KEY, str VARCHAR(10));
INSERT INTO pt VALUES(1,'one'), (2,'two');
COMMIT;
```

```
CREATE TABLE ft1( fpk INT PRIMARY KEY, FOREIGN KEY (ref) REFERENCES pt MATCH
SIMPLE);
INSERT INTO ft1 VALUES(100,1), (200,1); //This statement will insert 2 rows.
CREATE TABLE ft2( fpk INT PRIMARY KEY, FOREIGN KEY (ref) REFERENCES pt MATCH
UNIQUE SIMPLE);
INSERT INTO ft2 VALUES(100,1), (200,1);
```

次の文は、MATCH SIMPLE と MATCH FULL がどのように異なるかを示します。

```
CREATE TABLE pt2(
    pk1 INT NOT NULL,
    pk2 INT NOT NULL,
    str VARCHAR(10),
    PRIMARY KEY (pk1,pk2));
INSERT INTO pt2 VALUES(1,10,'one-ten'), (2,20,'two-twenty');
COMMIT;
CREATE TABLE ft3(
    fpk INT PRIMARY KEY,
    ref1 INT,
    ref2 INT );
ALTER TABLE ft3 ADD FOREIGN KEY (ref1,ref2)
    REFERENCES pt2 (pk1,pk2) MATCH SIMPLE;

CREATE TABLE ft4(
    fpk INT PRIMARY KEY,
    ref1 INT,
    ref2 INT );
ALTER TABLE ft4 add FOREIGN KEY (ref1,ref2)
    REFERENCES pt2 (pk1,pk2) MATCH FULL;
INSERT INTO ft3 VALUES(100,1,10);
// MATCH SIMPLE test succeeds; all column values match the corresponding values
in pt2.
INSERT INTO ft3 VALUES(200,null,null);
// MATCH SIMPLE test succeeds; at least one column in the key is null.
INSERT INTO ft3 VALUES(300,2,null);
// MATCH SIMPLE test succeeds; at least one column in the key is null.
INSERT INTO ft4 VALUES(100,1,10);
// MATCH FULL test succeeds; all column values match the corresponding values in
pt2.
INSERT INTO ft4 VALUES(200,null,null);
// MATCH FULL test succeeds; all column values in the key are null.
INSERT INTO ft4 VALUES(300,2,null);
// MATCH FULL test fails; both columns in the key must be null or match the
corresponding values in pt2.
```

例

次の例の 2 番目の文は、myT2 テーブルを作成し、%TYPE 属性を使用して、その myColumn カラムのデータ型を myT1 の last_name カラムのデータ型に設定します。null 入力可能性などの追加属性は適用されないため、myT2.myColumn には、myT1.last_name が持つ NOT NULL 制限と同じ制限は適用されません。

```
CREATE TABLE myT1
( first_name   CHAR(20),
   last_name   VARCHAR NOT NULL );
CREATE TABLE myT2
( myColumn myT1.last_name%TYPE );
```

次の例では、1 つのテーブルを作成してから、最初のテーブルの定義に基づき 2 番目のテーブルを作成します。

```
CREATE TABLE table1 ( ID INT NOT NULL DEFAULT AUTOINCREMENT, NAME LONG VARCHAR ) ;
CREATE TABLE table2 ( ADDRESS LONG VARCHAR ) ;
```

次の文は、データのない table1 と同様に、テーブルを作成します。

```
CREATE TABLE table3 LIKE table1 INCLUDING IDENTITY ;
```

次の文は、table1 と同様にテーブルを作成しますが、追加カラムがあります。

```
CREATE TABLE table4 ( LIKE table1 INCLUDING IDENTITY, LIKE table2, phone LONG
VARCHAR );
```

次の文は、table1 にデータがあり、さらに各ローの phone カラムが '555-5555' であるテーブルを作成します。

```
CREATE TABLE table5 AS ( SELECT * , '555-5555' AS phone FROM table1 ) WITH DATA ;
```

ESCAPE CHARACTER 句を活用するには、

1. test1.db と test2.db という名前の 2 つの SQL Anywhere データベースを作成します。
2. 両方のデータベースを同じサーバで起動します。

   ```
   dbsrv17 -n escape_test test1.db test2.db
   ```
3. test2 に接続し、次のテーブルを作成します。

   ```
   CREATE TABLE "table.with;fun{characters}"(c int);
   INSERT INTO "table.with;fun{characters}" VALUES(100);
   COMMIT;
   ```
4. 接続を切断し、test1 に接続します。
5. 次のとおり、test2 に対してリモートサーバを作成します。

   ```
   CREATE SERVER test2_server CLASS 'saodbc' USING 'driver=SQL Anywhere
   Native;eng=escape_test;dbn=test2';
   CREATE EXTERNLOGIN localuser TO test2_server REMOTE LOGIN remoteuser
   IDENTIFIED BY remotepwd;
   ```

   注記

   localuser は test1 へのログインに使用されたユーザ ID、一方で remotepwd は test2 へのログインに必要なリモートユーザ ID とパスワードにします。

ESCAPE CHARACTER 句を使用して、リモートの "table.with;fun{characters}" に対してプロキシテーブルを作成します。

```
CREATE EXISTING TABLE remtab AT 'test2_server;;;table.with!;fun!
{characters!}'ESCAPE CHARACTER'!';
```

または

```
CREATE EXISTING TABLE remtab AT 'test2_server...table!.with!;fun!
{characters!}'ESCAPE CHARACTER'!';
```

オプションで、プロキシテーブルに関するクエリを実行し、必要な結果セットを取得します。

```
SELECT c FROM remtab;
```

関連情報

SQL 変数 [118 ページ]
文字列 [10 ページ]
特別値 [83 ページ]
%TYPE および %ROWTYPE 属性 [110 ページ]
CREATE LOCAL TEMPORARY TABLE 文 [850 ページ]
@@identity グローバル変数 [121 ページ]
ALTER TABLE 文 [705 ページ]
CREATE EXISTING TABLE 文 [812 ページ]
DECLARE LOCAL TEMPORARY TABLE 文 [1007 ページ]
SQL データ型 [124 ページ]
CREATE DBSPACE 文 [789 ページ]

1.4.4.98 CREATE TEMPORARY TRACE EVENT 文

データベースが停止するまで持続するユーザトレースイベントを作成します。

構文

```
CREATE [ OR REPLACE ] TEMPORARY TRACE EVENT trace-event-name
[ WITH SEVERITY {
 0-255
 | ALWAYS
 | CRITICAL
 | ERROR
 | WARNING
 | INFORMATION
 | DEBUG } ]
[ ( field-name field-type [ , ... ] ] )
```

パラメータ

trace-event-name

ユーザトレースイベントの名前を指定します。ユーザトレースイベントの名前の先頭をプレフィクス SYS_ にすることはできません。システムトレースイベントの名前は指定できません。

OR REPLACE 句

新しいトレースイベントを作成するか、同じ名前の既存のトレースイベントを置き換えます。

WITH SEVERITY 句

重要度レベルを指定しない場合は、デフォルトの重要度レベル (DEBUG) が使用されます。ユーザトレースイベントは、そのトレースイベントが作成されたときにユーザが接続していたデータベースによって所有されます。サポートされる重要度の値は次のとおりです。

レベル	重要度の値の範囲
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

field-name

ユーザトレースイベントから特定のタイプの情報を収集するフィールド。このフィールドには有効な識別子を指定します。

field-type

カラムでサポートされているすべてのデータ型 (配列型以外) を使用できます。

備考

トレースイベントはメモリに格納され、それが明示的に削除されていない場合は、データベースサーバが停止すると削除されます。

システム権限

MANAGE ANY TRACE SESSION システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザトレースイベントを作成します。

```
CREATE TEMPORARY TRACE EVENT my_event( id INTEGER, information LONG VARCHAR );
```

関連情報

CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.4.4.99 CREATE TEMPORARY TRACE EVENT SESSION 文

ユーザトレースイベントセッションを作成します。

構文

```
CREATE [ OR REPLACE ] TEMPORARY TRACE EVENT SESSION session-name
[ ON SERVER ]
event-definition [ ,... ]
[ target-definition ]
```

```
event-definition :
ADD TRACE EVENT event-name [ ( WHERE search-condition ) ]
```

```
target-definition :
ADD TARGET FILE
  ( SET target-parameter-name=target-parameter-value [ ,... ] )
```

```
target-parameter-name :
{ filename_prefix
  | max_size
  | num_files
  | flush_on_write
  | ["compressed"] }
```

パラメータ

OR REPLACE 句

CREATE OR REPLACE を指定すると、トレースイベントセッションを作成するか、同じ名前を持つ既存のトレースイベントセッションを置き換えます。

ON SERVER 句

トレースイベントセッションは、データベースサーバ上のすべてのデータベースのトレースイベントを記録します。この句を指定しない場合、トレースイベントセッションは、そのセッションが作成されたデータベースのトレースイベントのみを記録します。

WHERE 句

WHERE 句を使用すると、イベントのプロパティに基づく条件に応じてイベントをトレースできます。WHERE 句には、定数やイベントフィールドを参照する式を含めることができます。`search-condition` の中で組み込み関数が使用されている場合、その関数はイベントが生成される接続に関して評価されます。たとえば、`connection_property('number') = 101` が使用されている場合、101 番の接続に関するイベントだけが記録されます。

`search-condition` は、イベントが `target-definition` に送信される前に適用されます。`search-condition` から FALSE または UNKNOWN が返された場合、イベントはターゲットに送信されません。

`search-condition` に以下のいずれかが含まれている場合、エラーが返されます。

- サブクエリ
- ユーザ定義関数
- シーケンス
- ホスト変数
- カラム参照 (イベントのフィールドは参照できますが、イベントの WHERE 句は参照できません)
- 接続またはデータベースの変数

session-name

トレースイベントセッションの名前。

event-name

セッションに追加するトレースイベントの名前。システム定義およびユーザ定義のトレースイベントがサポートされます。sp_trace_events システムプロシージャを呼び出し、システム定義のトレースイベントのリストを取得します。

target-name

サポートされる値は FILE だけです。

target-parameter-name

次のターゲットパラメータがサポートされます。

target-parameter-name	target-parameter-value
filename_prefix	パス付きまたはパスなしの ETD ファイル名のプレフィックス。ETD ファイルには、`.etd` という拡張子が付きます。このパラメータは必須です。

target-parameter-name	target-parameter-value
max_size	ファイルの最大サイズ (バイト単位)。デフォルトは 0 で、これはファイルサイズに上限がないことを意味し、ディスク領域が許す限り大きくなっていきます。指定したサイズに達すると、新しいファイルが開始されます。
num_files	このオプションは、max_size がゼロ以外の値に設定されている場合にファイルの数を制限するために使用します。イベントトレース情報が多数のファイルに分割されている場合は、保持する追加ファイルの数になります。デフォルトは 0 で、ファイル数の制限がないことを示します。ファイルが指定した最大サイズに達すると、データベースサーバは新しいファイルへの書き込みを開始します。各ファイルの名前は suffix _N となり、N は 0 から始まります。num_files が 0 以外の値の場合、古いファイルは自動的に削除されます。たとえば、num_files が 2 で、現在のファイル番号サフィックスが _100 の場合、サフィックス _100、_99、および _98 を持つファイルが保持されます (現在のファイルとそれ以外に 2 つ)。サフィックス _97 を持つファイルは削除されます。それぞれの新しいファイルに書き込みが行われるたびに、この番号サフィックスが 1 ずつ増加します。
flush_on_write	記録されるイベントが発生するたびにディスクバッファをフラッシュするかどうかを制御するブール値 (true または false)。デフォルトは false です。フラッシュを有効にすると、多くのトレースイベントが記録される場合、データベースサーバのパフォーマンスが低下することがあります。
[compressed]	ディスク領域を節約するためのログファイルの圧縮を制御するブール値 (true または false)。デフォルトは false です。このパラメータは他のコンテンツのキーワードであるため、パラメータ名を角括弧で囲む必要があります。

備考

トレースイベントセッションは、それらが ALTER TRACE EVENT SESSION 文によって明示的に開始されない限り実行されません。トレースイベントセッションを使用すると、システムの動作に関連するトレースイベントや、特定のユーザに関するトレースイベントが取得されます。トレースイベントセッションはメモリに格納され、それらが明示的に削除されていない場合は、データベースサーバが停止すると削除されます。

システム権限

MANAGE ANY TRACE SESSION システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザ定義イベント my_event とシステム定義イベント SYS_ConsoleLog_Information に関する情報を、`my_trace_file` という名前のファイルに記録するイベントトレースセッションを作成します。

```
CREATE TEMPORARY TRACE EVENT SESSION my_session
    ADD TRACE EVENT my_event, -- user event
    ADD TRACE EVENT SYS_ConsoleLog_Information -- system event
    ADD TARGET FILE (SET filename_prefix='my_trace_file', [compressed]=1); -- add
a target
```

次の文は、ユーザ DBA に関する接続イベントを記録する、MySession という名前のイベントトレースセッションを作成します。

```
CREATE TEMPORARY TRACE EVENT SESSION MySession
ADD TRACE EVENT SYS_RLL_Connect
( WHERE user='DBA' );
```

関連情報

探索条件 [53 ページ]
CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.4.4.100 CREATE TEXT CONFIGURATION 文

テキストインデックスの構築と更新に使用するテキスト設定オブジェクトを作成します。

構文

```
CREATE TEXT CONFIGURATION [ owner.]new-config-name
    FROM [ owner.]existing-config-name
```

パラメータ

FROM 句

新しいテキスト設定オブジェクトを作成するときのテンプレートとして使用するテキスト設定オブジェクトの名前を指定します。デフォルトのテキスト設定オブジェクトの名前は、default_char と default_nchar です。

テキスト設定オブジェクトを作成すると、日付値と時刻値の文字列への変換方法に影響を及ぼすデータベースオプションが、default_char と default_nchar のテキスト設定オブジェクトテンプレートからコピーされます。

備考

別のテキスト設定オブジェクトをテンプレートとして使用してテキスト設定オブジェクトを作成し、その後、必要に応じて ALTER TEXT CONFIGURATION 文を使用してオプションを変更します。

データベース内のすべてのテキスト設定オブジェクトのリストとその設定を表示するには、SYSTEXTCONFIG システムビューを問い合わせます。

権限

ユーザ本人が所有するテキスト設定オブジェクトを作成するには、CREATE TEXT CONFIGURATION システム権限が必要です。他のユーザが所有するテキスト設定オブジェクトを作成するには、CREATE ANY TEXT CONFIGURATION または CREATE ANY OBJECT システム権限が必要です。

すべてのテキスト設定オブジェクトには PUBLIC アクセスがあります。テキストインデックスを作成できる権限を持つユーザであれば、任意のテキスト設定オブジェクトを使用できます。

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

次の CREATE TEXT CONFIGURATION 文は、default_char テキスト設定オブジェクトを使用して、テキスト設定オブジェクト max_term_sixteen を作成します。後続の ALTER TEXT CONFIGURATION 文は、max_term_sixteen の最大単語長を 16 に変更します。

```
CREATE TEXT CONFIGURATION max_term_sixteen FROM default_char;
ALTER TEXT CONFIGURATION max_term_sixteen
   MAXIMUM TERM LENGTH 16;
```

関連情報

SYSTEXTCONFIG システムビュー [1849 ページ]
ALTER TEXT CONFIGURATION 文 [720 ページ]
DROP TEXT CONFIGURATION 文 [1076 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]

1.4.4.101 CREATE TEXT INDEX 文

テキストインデックスを作成します。

構文

```
CREATE TEXT INDEX [ IF NOT EXISTS ] text-index-name
ON [ owner. ] { table-name | mv-name } ( column-name, ... )
  [ IN dbspace-name ]
  [ CONFIGURATION [ owner. ]text-configuration-name ]
  [ { IMMEDIATE REFRESH
    | MANUAL REFRESH
    | AUTO REFRESH [ EVERY integer { MINUTES | HOURS } ] } ]
```

パラメータ

IF NOT EXISTS 句

IF NOT EXISTS 句が指定され、指定されたテキストインデックスが存在する場合は、変更は行われず、エラーも返されません。

ON 句

テキストインデックスを構築するテーブルとカラムを指定します。

IN 句

テキストインデックスが格納される DB 領域を指定します。この句が指定されていない場合は、テキストインデックスが参照するテーブルと同じ DB 領域でテキストインデックスが作成されます。

CONFIGURATION 句

テキストインデックスの作成時に使用するテキスト設定オブジェクトを指定します。この句が指定されておらず、インデックスのカラムのいずれかが NCHAR の場合は、default_nchar テキスト設定オブジェクトが使用されます。それ以外の場合は、default_char テキスト設定オブジェクトが使用されます。

REFRESH 句

テキストインデックスの再表示タイプを指定します。REFRESH 句を指定しない場合、デフォルトとして IMMEDIATE REFRESH が使用されます。次の再表示タイプを指定できます。

IMMEDIATE REFRESH

基本となるテーブルまたはマテリアライズドビューの変更によってテキストインデックスのデータが影響を受けるたびに、テキストインデックスを再表示します。

AUTO REFRESH

内部サーバイベントを使用して自動的にテキストインデックスを再表示します。EVERY サブ句は、再表示間隔を分または時間単位で指定するときに使用します。間隔情報を指定しないで AUTO REFRESH を指定した場合、データベースサーバは 60 分ごとにテキストインデックスを再表示します。sa_text_index_stats システムプロシージャから返された pending_size の値が、最後の再表示のときにテキストインデックスサイズの 20 % を超えるか、deleted_length がテキストインデックスサイズの 50 % を超えると、AUTO REFRESH 句で指定された間隔よりも早くテキストインデックスが再表示される場合があります。1 分ごとに内部イベントが実行され、AUTO REFRESH の全テキストインデックスに対してこの条件がチェックされます。

MANUAL REFRESH

テキストインデックスが手動で再表示されます。

備考

同じカラムを参照している複数のテキストインデックスを作成すると、予期しない結果が生じる可能性があり、おすすめしません。

通常のビューやテンポラリテーブルに対してテキストインデックスを作成することはできません。

マテリアライズドビューに対してテキストインデックスが作成されると、そのテキストインデックスを再表示したりトランケートすることは不可能になり、削除だけが可能になります。マテリアライズドビューに対するテキストインデックスは、基本となるマテリアライズドビューが再表示されたり更新されると常に、データベースサーバによって管理されます。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

ベーステーブルに対する IMMEDIATE REFRESH テキストインデックスは作成時にデータが設定され、この初期再表示の間はテーブルに排他ロックがかけられます。IMMEDIATE REFRESH のテキストインデックスは、スナップショットアイソレーションを使用するクエリを完全にサポートします。

マテリアライズドビューに対する IMMEDIATE REFRESH テキストインデックスは、そのビューにデータが取り込まれる場合には、作成時にデータが設定されます。

MANUAL と AUTO REFRESH のテキストインデックスは、作成後に初期化 (再表示) されます。

AUTO REFRESH のテキストインデックスの再表示では、独立性レベル 0 を使用してテーブルがスキャンされます。

テキストインデックスの作成後に、その定義を IMMEDIATE REFRESH に変更したり、IMMEDIATE REFRESH から変更したりできません。いずれかの変更が必要な場合は、テキストインデックスを削除して再作成します。

REFRESH TEXT INDEX 文を使用して、AUTO REFRESH テキストインデックスを手動で再表示することができます。

権限

テーブルのテキストインデックスを作成するには、テーブルの所有者であるか、次のいずれかの権限を持っている必要があります。

- そのテーブルに対する REFERENCES 権限
- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限

マテリアライズドビューのテキストインデックスを作成するには、マテリアライズドビューの所有者であるか、次のいずれかの権限を持っている必要があります。

- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、サンプルデータベースの MarketingInformation テーブルの Description カラムについてテキストインデックス myTxtIdx を作成します。MarketingTextConfig テキスト設定オブジェクトが使用され、再表示間隔は 24 時間ごとに設定されます。

```
CREATE TEXT INDEX myTxtIdx ON GROUPO.MarketingInformation ( Description )
   CONFIGURATION default_char
   AUTO REFRESH EVERY 24 HOURS;
```

関連情報

REFRESH TEXT INDEX 文 [1255 ページ]
SYSTEXTCONFIG システムビュー [1849 ページ]
ALTER TEXT INDEX 文 [724 ページ]
DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
sa_char_terms システムプロシージャ [1452 ページ]
sa_nchar_terms システムプロシージャ [1574 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]
sa_text_index_stats システムプロシージャ [1641 ページ]

1.4.4.102 CREATE TIME ZONE 文

データベースがサーバのタイムゾーンとは異なるタイムゾーンをシミュレートする場合に使用する、タイムゾーンオブジェクトを作成します。

構文

```
CREATE [ OR REPLACE ] TIME ZONE
name time-zone-option [ ... ]
```

```
time-zone-option :
 { OFFSET offset
 | DST OFFSET offset
 | NO DST
 | [ DST ] STARTING month-day-rule AT { minutes | hours:minutes }
 | [ DST ] ENDING month-day-rule AT { minutes | hours:minutes } }
```

```
offset :
'{ + | -} { minutes | hours:minutes } '
```

```
month-day-rule :
'month / day-rule'
```

```
day-rule :
number
| LAST day
| day >= number
```

パラメータ

OR REPLACE clause

新しいタイムゾーンを作成するか、同じ名前の既存のタイムゾーンを置き換えます。タイムゾーンがすでに存在し、OR REPLACE が指定されていない場合、文は失敗します。

offset clause

タイムゾーンの標準オフセットを UTC より進んでいる時間と分で指定します。オフセットが整数の場合は分を示します。そのため、1:00 と 60 は同じです。オフセットには +23:59 (1439 分) より大きい値や -23:59 (-1439 分) 未満の値を指定することはできません。

DST OFFSET clause

夏時間の使用による標準時間からの差を指定します。DST 句が使用されていない場合、デフォルトは NO DST です。STARTING 句と ENDING 句が指定されていて DST OFFSET 句が指定されていない場合、デフォルトは 1 時間です。

NO DST clause

タイムゾーンで夏時間を使用しないことを指定します。STARTING、ENDING、または DST OFFSET 句は、この句と一緒に使用できません。

STARTING clause

夏時間が始まる日付および時刻を指定します。

ENDING clause

夏時間が終了する日付および時刻を指定します。

month-day-rule

`month` は 1 ～ 12 の数値または英語の 3 文字の省略形 (Jan や Feb など) です。

day-rule

夏時間が開始または終了する月の日付。

number

1 ～ 31 の数字。`number` を指定した場合、`day-rule` は月の日付です。たとえば、`month-day-rule` に **`Mar/12`** を指定した場合、3 月 12 日を表します。

last day

指定した曜日における月の最後の日。たとえば、**`lastFri`** を指定した場合、その月の最終金曜日を表します。

day

1 ～ 7 の数値または英語の 3 文字の省略形 (Mon や Tue など) です。

day >= number

特定の日付以降の最初の指定日。`number` はその月の日数以下である必要があります。たとえば、**`Feb/Mon>=28`** です。

> **注記**
>
> `day>=number` は、別の月の日付と一致する場合があります。たとえば、DST ルール 'Sep / Wed >= 25' (9 月 25 以降の最初の水曜日) は、年によっては 10 月になる場合があります。

備考

DST 句はオプションです。タイムゾーンで夏時間を使用する場合、STARTING 句および ENDING 句を指定する必要があります。

シミュレートされたタイムゾーンオブジェクトは、1 つのデータベースに複数個作成できます。複数のデータベースを稼働させている場合、それぞれのデータベースがそのサーバの実際のタイムゾーンとは異なる個別のシミュレートされたタイムゾーンで稼働することが可能です。目的のタイムゾーンに切り替えるには、SET OPTION PUBLIC.time_zone 文を使用します。

権限

タイムゾーンの作成には、MANAGE TIME ZONE システム権限が必要です。

関連する動作

オートコミット。

この文を実行すると、ISYSTIMEZONE システムテーブルへの移植が行われます。

例

東部標準時を追加するには、次の文を実行します。

```
CREATE TIME ZONE EST5EDT OFFSET '-05:00'
STARTING 'Mar/Sun>=8' AT '2:00'
ENDING 'Nov/Sun>=1' AT '2:00';
```

このタイムゾーンは、UTC より 5 時間遅れています。夏時間は、3 月 8 日以降の最初の日曜日 (3 月の第 2 日曜日) の午前 2 時から 11 月の第 1 日曜日の午前 2 時までです。

オーストラリア東部標準時を追加するには、次の文を実行します。

```
CREATE TIME ZONE NewSouthWales OFFSET '10:00'
STARTING 'Oct/Sun>=1' AT '2:00'
ENDING 'Apr/Sun>=1' AT '2:00';
```

このタイムゾーンは、UTC より 10 時間進んでいます。夏時間は、10 月の第 1 日曜日の午前 2 時から 4 月の第 1 日曜日の午前 2 時までです。

NewSouthWales タイムゾーンに切り替えるには、次の文を実行します。

```
SET OPTION PUBLIC.time_zone='NewSouthWales';
```

サーバのタイムゾーンに戻すには、次の文を実行します。

```
SET OPTION PUBLIC.time_zone=;
```

関連情報

COMMENT 文 [769 ページ]
ALTER TIME ZONE 文 [726 ページ]
DROP TIME ZONE 文 [1078 ページ]
SYSTIMEZONE システムビュー [1851 ページ]

1.4.4.103 CREATE TRIGGER 文

テーブル内にトリガを作成します。

構文

```
CREATE [ OR REPLACE ] TRIGGER trigger-name trigger-type
{ trigger-event-list | UPDATE OF column-list }
[ ORDER integer ] ON table-name
[ REFERENCING [ OLD AS old-name ]
   [ NEW AS new-name ]
   [ REMOTE AS remote-name ] ]
[ FOR EACH { ROW | STATEMENT } ]
[ WHEN ( search-condition ) ]
trigger-body
```

```
column-list :  column-name[, ...]
```

```
trigger-type :
BEFORE
| AFTER
| INSTEAD OF
| RESOLVE
```

```
trigger-event-list : trigger-event[, ...  ]
```

```
trigger-event :
DELETE
| INSERT
| UPDATE
```

```
trigger-body: ブール論理キーワード ({ IF | ELSIF } { INSERTING | UPDATING |
DELETING } THEN some-action) をオプションで含める BEGIN 文
```

パラメータ

OR REPLACE 句

OR REPLACE を指定すると、新しいトリガが作成されるか、同じ名前の既存のトリガが置き換えられます。

trigger-type

ローレベルのトリガを定義して、挿入、更新、または削除の前 (BEFORE)、後 (AFTER) または代わり (INSTEAD OF) に実行できます。文レベルのトリガは、文の INSTEAD OF または AFTER の実行を定義できます。

BEFORE UPDATE トリガは、ローを対象に UPDATE が実行されるたび、および新しい値が古い値と異なるたびに起動されます。つまり、BEFORE UPDATE トリガに `column-list` が指定されている場合は、UPDATE 文の SET 句に `column-list` 内のカラムがあると、トリガが起動します。AFTER UPDATE トリガに `column-list` が指定されている場合は、`column-list` 内のいずれかのカラムの値が UPDATE 文によって変更された場合にのみ、トリガが起動します。

INSTEAD OF トリガは、通常のビューに定義できるトリガの唯一の形式です。INSTEAD OF トリガは、他の動作を、トリガ動作で置換します。INSTEAD OF トリガが起動すると、トリガ動作はスキップされ、指定された動作が実行されます。INSTEAD OF トリガは、ローレベルまたは文レベルのトリガとして定義できます。文レベルの INSTEAD OF トリガは、ローレベルのすべての処理を含め、文全体を置換します。文レベルの INSTEAD OF トリガが起動すると、その文の結果としてローレベルのトリガが起動することはありません。ただし、文レベルのトリガの本文が、その他の処理を実行するため、結果として、その他のローレベルのトリガが実行されます。

INSTEAD OF トリガを定義する場合は、UPDATE OF `column-list` 句、ORDER 句、または WHEN 句は使用できません。

RESOLVE トリガ型は、SQL Remote とともに使用します。これは、ローレベルの UPDATE または UPDATE OF `column-list` の前にのみ起動されます。

trigger-event

トリガを定義する場合、同じ定義内で DELETE、INSERT、および UPDATE イベントを組み合わせることはできますが、UPDATE OF イベントのトリガは個別に定義する必要があります。1 つのテーブル上で任意の数の DELETE、INSERT、および UPDATE を定義することができます。1 つのテーブル上で UPDATE OF イベント のトリガは任意の数だけ定義できますが、カラムごとに 1 つのみです。

トリガは次のイベントによって起動できます。

DELETE イベント

テーブルの 1 つ以上のローが削除されるたびにトリガが起動されされます。

INSERT イベント

テーブルに 1 つ以上のローが挿入されるたびにトリガが起動されされます。

UPDATE イベント

テーブルの 1 つ以上のローが更新されるたびにトリガが起動されされます。

他の SQL ダイアレクトとの互換性を維持するために、この句ではキーワード UPDATING もサポートされます。UPDATING の引数は引用符付きの文字列ですが (たとえば、`UPDATING( 'mycolumn' )`)、UPDATE の引数は識別子 (たとえば、`UPDATE( mycolumn )`) です。

UPDATE OF column-list イベント

このトリガは、関連するテーブルのローが更新され、`column-list` のカラムが修正されると必ず呼び出されます。このタイプのトリガイベントは、`trigger-event-list` では使用できません。このトリガ用に定義された唯一のトリガイベントであることが必要です。この句は、INSTEAD OF トリガでは使用できません。

カラムごとに指定できる UPDATE OF は 1 つのみです。

処理が必要なイベントごとにトリガを個別に作成できます。または、共有するアクションや、イベントに応じたアクションが複数ある場合は、すべてのイベントに対して 1 つのトリガを作成し、IF 文を使用して実行するアクションを区別できます。

ORDER 句

1 つのテーブルで複数のトリガを定義する場合、これらが同じタイプでないとしても、その順序を指定することをお奨めします。これにより、結果が予測可能になり、結果を処理する順序を確定しやすくなります。

同じタイミング (before、after、resolve) で起動する同じタイプ (insert、update、delete) の追加トリガを定義する場合は、ORDER 句を指定してトリガを起動する順序をデータベースサーバに指示します。同じタイミングで起動するように設定された同じタイプのトリガの間では、順序番号をユニークにします。ユニークでない順序番号を指定すると、エラーが返されます。順序番号は、連続した順番にする必要はありません (たとえば、1、12、30 と指定できます)。データベースサーバは、最も小さい番号が付いたトリガから順に起動します。

通常、ORDER 句を省略するか、0 を指定すると、データベースサーバは順序番号 1 を割り当てます。ただし、すでに同じタイプの別のトリガが 1 に設定されている場合は、エラーが返されます。

複数のイベントタイプが含まれる追加トリガを作成するときに、ORDER 句を省略し、1 つ以上のイベントタイプが他のトリガのものと同じである (たとえば、あるトリガの trigger-event-list が UPDATE、INSERT であり、別のトリガの trigger-event-list が UPDATE である) 場合、データベースサーバはエラーを返しません。この場合、データベースサーバは、予期が不可能で変更される可能性がある実装固有の順序でこれらのトリガを処理します。したがって、1 つのテーブルで複数のトリガを定義する場合は、ORDER 句を常に指定することを強くお奨めします。

追加トリガを追加する場合は、トリガの動作が相互に影響するかどうかによって、イベントの同じタイプの既存トリガを修正する必要が生じる可能性があります。相互に影響しない場合、新しいトリガには既存の他のトリガと異なる ORDER 値が必要です。相互に影響する場合は、他のトリガの動作を検討する必要があり、これらのトリガが起動する順序の変更が必要となる可能性があります。

ORDER 句は、INSTEAD OF トリガではサポートされません。テーブルまたはビューに定義されたタイプ (insert、update、delete) ごとに、1 つの INSTEAD OF トリガだけを使用できます。

REFERENCING 句

REFERENCING OLD 句と REFERENCING NEW 句を使用すると、挿入、削除、または更新されたローを参照できます。この句では、UPDATE は削除とそれに続く挿入として取り扱われます。

INSERT は REFERENCING NEW 句を使います。これは挿入されたローを表します。REFERENCING OLD 句はありません。

DELETE は REFERENCING OLD 句を使います。これは削除されたローを表します。REFERENCING NEW 句はありません。

UPDATE は、更新前のローを表す場合は REFERENCING OLD 句を使用し、更新後のローを表す場合は REFERENCING NEW 句を使用します。

REFERENCING OLD と REFERENCING NEW の意味は、トリガがローレベルのトリガなのか、文レベルのトリガなのかによって異なります。ローレベルのトリガの場合、REFERENCING OLD 句を使うと、更新または削除する前のローの値を参照できます。また、REFERENCING NEW 句を使用すると、挿入または更新された値を参照できます。OLD ローと NEW ローは、BEFORE と AFTER トリガの中で参照できます。REFERENCING NEW 句を使うと、BEFORE トリガの新しいローを修正してから、挿入または更新操作を行うことができます。

文レベルのトリガの場合、REFERENCING OLD と REFERENCING NEW 句は、ローの古い値と新しい値を保持している宣言されたテンポラリテーブルを参照します。

REFERENCING REMOTE 句は SQL Remote で使用します。これを使うと、UPDATE 文の VERIFY 句に設定されている値を参照できます。これは、必ずカラムリストのトリガ RESOLVE UPDATE または RESOLVE UPDATE OF と一緒に使用してください。

FOR EACH 句

トリガをローレベルのトリガとして宣言するには、FOR EACH ROW 句を使用します。トリガを文レベルのトリガとして宣言するには、FOR EACH STATEMENT 句を使用するか、または FOR EACH 句を省略します。文レベルのトリガを宣言する場合は、わかりやすくするために、FOR EACH STATEMENT 句を指定することをお奨めします。

WHEN 句

探索条件が真と評価されたローに対してだけ、トリガが起動されます。WHEN 句は、ローレベルトリガと一緒にだけ使用できます。この句は、INSTEAD OF トリガでは使用できません。

trigger-body

トリガの本文には、トリガアクションが生じたときに実行されるアクションが含まれます。これは、BEGIN 文で構成されます。

BEGIN 文にトリガオペレーション条件を指定できます。トリガオペレーション条件は、トリガを起動したトリガイベントに従って動作を行います。たとえば、トリガが更新と削除の両方のために起動するように定義されている場合は、2 つの条件に異なる動作を指定できます。

また、トリガ本文内の条件を使用できる場所であればどこでも、ブール条件 `{ INSERTING | DELETING | UPDATING [ ( 'col-name' ) ] }` を使用できます。この特別な構文を使用して、一定の `trigger-event` の実行時に追加のアクションを指定することができます。たとえば、`IF INSERTING THEN SET msg = msg || 'insert'` のように記述します。

備考

CREATE TRIGGER 文は、データベースのテーブルに関連するトリガを作成し、データベースにトリガを格納します。

マテリアライズドビューにトリガを定義することはできません。定義すると、SQLE_INVALID_TRIGGER_MATVIEW エラーが返されます。

トリガをローレベルのトリガとして宣言すると、各ローを修正する前または後にトリガが実行されます。また、トリガを文レベルのトリガとして宣言すると、トリガ文全体が完了してから、トリガが実行されます。

CREATE TRIGGER はテーブルにテーブルロックを設定して、テーブルを排他的に使用します。

権限

CREATE ANY TRIGGER または CREATE ANY OBJECT のシステム権限が必要です。さらに、そのトリガが作成されるテーブルの所有者であるか、または次のいずれかの権限を持っていることも必要です。

- そのテーブルに対する ALTER 権限
- ALTER ANY TABLE システム権限
- ALTER ANY OBJECT システム権限

別のユーザが所有するビューのトリガを作成するには、CREATE ANY TRIGGER または CREATE ANY OBJECT システム権限のどちらかを持っていて、ALTER ANY VIEW または ALTER ANY OBJECT システム権限のどちらかを持っている必要があります。

既存のトリガを置き換えるには、そのトリガが作成されるテーブルの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- CREATE ANY TRIGGER システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY TRIGGER システム権限。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

CREATE TRIGGER は、オプションの ANSI/ISO SQL 言語機能 BT211、"Basic trigger capability" の一部です。ROW トリガはオプションの ANSI/ISO SQL 言語機能 T212 であり、INSTEAD OF トリガはオプションの ANSI/ISO SQL 言語機能 T213 です。

ソフトウェアの一部のトリガ機能は標準にありません。これらを以下に示します。

- オプションの OR REPLACE 構文。既存のトリガを置き換える場合、新しいトリガインスタンスの作成の認証は省略されます。
- ORDER 句。ANSI/ISO SQL 標準では、トリガは作成順に起動されます。
- RESOLVE トリガ。

Transact-SQL

ROW トリガと RESOLVE トリガは、Adaptive Server Enterprise でサポートされていません。Transact-SQL の INSTEAD OF トリガは、Adaptive Server Enterprise ではサポートされていますが、SQL Anywhere の Transact-SQL ダイアレクトではサポートされていません。Transact-SQL トリガは、異なる構文で定義されます。

例

この例は、文レベルのトリガを作成します。最初に、この CREATE TABLE 文に示されているようにテーブルを作成します (CREATE TABLE システム権限が必要)。

```
CREATE TABLE t0
( id INTEGER NOT NULL,
 times TIMESTAMP NULL DEFAULT CURRENT TIMESTAMP,
 remarks TEXT NULL,
 PRIMARY KEY ( id )
);
```

次に、このテーブルについて文レベルのトリガを作成します。

```
CREATE TRIGGER myTrig AFTER INSERT ORDER 4 ON t0
REFERENCING NEW AS new_name
FOR EACH STATEMENT
BEGIN
  DECLARE @id1 INTEGER;
  DECLARE @times1 TIMESTAMP;
  DECLARE @remarks1 LONG VARCHAR;
  DECLARE @err_notfound EXCEPTION FOR SQLSTATE VALUE '02000';
//declare a cursor for table new_name
  DECLARE new1 CURSOR FOR
   SELECT id, times, remarks FROM new_name;
  OPEN new1;
 //Open the cursor, and get the value
  LoopGetRow:
  LOOP
      FETCH NEXT new1 INTO @id1, @times1,@remarks1;
      IF SQLSTATE = @err_notfound THEN
   LEAVE LoopGetRow
      END IF;
      //print the value or for other use
      PRINT (@remarks1);
  END LOOP LoopGetRow;
  CLOSE new1
END;
```

次の例は、前の例で作成された myTrig トリガを置き換えます。

```
CREATE OR REPLACE TRIGGER myTrig AFTER INSERT ORDER 4 ON t0
REFERENCING NEW AS new_name
FOR EACH STATEMENT
BEGIN
FOR L1 AS new1 CURSOR FOR
   SELECT id, times, remarks FROM new_name
DO
     //print the value or for other use
     PRINT (@remarks1);
END FOR;
END;
```

次の例は、BEFORE UPDATE トリガで REFERENCING NEW を使用する方法を示します。次の例では、新規 Employees テーブルの郵便番号が大文字になるようにしています。この文を実行するには、GROUPO.Employees に対する SELECT、ALTER、UPDATE オブジェクトレベル権限が必要です。

```
CREATE TRIGGER emp_upper_postal_code
BEFORE UPDATE OF PostalCode
ON GROUPO.Employees
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN ( ISNUMERIC( new_emp.PostalCode ) = 0 )
BEGIN
   -- Ensure postal code is uppercase (employee might be
   -- in Canada where postal codes contain letters)
   SET new_emp.PostalCode = UPPER(new_emp.PostalCode)
END;
UPDATE GROUPO.Employees SET state='ON', PostalCode='n2x 4y7' WHERE EmployeeID=191;
SELECT PostalCode FROM GROUPO.Employees WHERE EmployeeID = 191;
```

次の例は、BEFORE DELETE トリガで REFERENCING OLD を使用する方法を示します。この例は、Employees テーブルから、退職していない従業員が削除されないようにします。

```
CREATE TRIGGER TR_check_delete_employee
```

```
BEFORE DELETE
ON Employees
REFERENCING OLD AS current_employee
FOR EACH ROW WHEN ( current_employee.Terminate IS NULL )
BEGIN
    RAISERROR 30001 'You cannot delete an employee who has not been fired';
END;
```

次の例は、BEFORE UPDATE トリガで REFERENCING NEW と REFERENCING OLD を使用する方法を示します。この例は、従業員の給料に減給が生じないようにします。

```
CREATE TRIGGER TR_check_salary_decrease
    BEFORE UPDATE
      ON GROUPO.Employees
    REFERENCING OLD AS before_update
    NEW AS after_update
FOR EACH ROW
BEGIN
   IF after_update.salary < before_update.salary THEN
    RAISERROR 30002 'You cannot decrease a salary';
    END IF;
END;
```

次の例は、BEFORE INSERT トリガと UPDATE トリガで REFERENCING NEW を使用する方法を示します。この例では、SalesOrderItems テーブルのローで挿入や更新が行われる前に起動するトリガを作成します。

```
CREATE TRIGGER TR_update_date
   BEFORE INSERT, UPDATE
     ON GROUPO.SalesOrderItems
   REFERENCING NEW AS new_row
FOR EACH ROW
BEGIN
   SET new_row.ShipDate = CURRENT TIMESTAMP;
END;
```

次のトリガでは、トリガが起動されるきっかけとなったアクションを示すメッセージが、Interactive SQL の結果ウィンドウ枠の履歴タブに表示されます。

```
CREATE TRIGGER tr BEFORE INSERT, UPDATE, DELETE
ON sample_table
REFERENCING OLD AS t1old
FOR EACH ROW
BEGIN
    DECLARE msg varchar(255);
    SET msg = 'This trigger was fired by an ';
    IF INSERTING THEN
        SET msg = msg || 'insert'
    ELSEIF DELETING THEN
        set msg = msg || 'delete'
    ELSEIF UPDATING THEN
        set msg = msg || 'update'
    END IF;
    MESSAGE msg TO CLIENT
END;
```

関連情報

BEGIN 文 [745 ページ]

CREATE TRIGGER 文 [T-SQL] [989 ページ]
DROP TRIGGER 文 [1082 ページ]
ROLLBACK TRIGGER 文 [1287 ページ]
UPDATE 文 [1391 ページ]
ALTER TRIGGER 文 [730 ページ]
RAISERROR 文 [1246 ページ]
CONFLICT 関数 [その他] [277 ページ]

1.4.4.104 CREATE TRIGGER 文 [T-SQL]

Adaptive Server Enterprise 互換の方法で、データベース内に新しいトリガを作成します。

構文

一般的な使用

```
CREATE TRIGGER [owner.]trigger_name
ON [owner.]table_name
FOR { INSERT, UPDATE, DELETE }
AS statement-list
```

検索条件の指定

```
CREATE TRIGGER [owner.]trigger_name
ON [owner.]table_name
FOR {INSERT, UPDATE}
AS
[ IF UPDATE ( column-name )
[ { AND | OR } UPDATE ( column-name ) ] ... ]
  statement-list
[ IF UPDATE ( column-name )
[ { AND | OR} UPDATE ( column-name ) ] ... ]
  statement-list
```

備考

CREATE TRIGGER は、テーブルに対する排他的なテーブルロックを取得します。

削除または挿入されたローは、2 つの宣言されたテンポラリテーブル内に保持されます。Transact-SQL トリガでは、Adaptive Server Enterprise の場合と同様に、テーブル名 "deleted"と "inserted" を使用してこれらのローにアクセスできます。Watcom-SQL CREATE TRIGGER 文では、REFERENCING 句を使用してこれらのローを参照します。

トリガ名はデータベース内でユニークでなければなりません。

Transact-SQL トリガは、そのトリガを起動した文が実行された後に実行されます。

Transact-SQL トリガの作成時には ORDER 句がサポートされていないため、`trigger_order` の値は 1 に設定されます。SYSTRIGGER システムテーブルには、`table_id`、`event`、`trigger_time`、および `trigger_order` というユニークな

インデックスがあります。特定のイベント (挿入、更新、削除) では、文レベルのトリガは常に AFTER であり、trigger_order は設定できません。このため、その他のトリガで 1 以外の順序に設定されないことを前提として、テーブルごとに各タイプのいずれか 1 つのみを設定できます。

権限

テーブルの所有者であるか、テーブルに対する ALTER 権限を持っているか、または ALTER ANY TABLE システム権限を持っていることが必要です。さらに、CREATE ANY TRIGGER または CREATE ANY OBJECT システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

ROW トリガは、Adaptive Server Enterprise でサポートされていません。Transact-SQL の INSTEAD OF トリガは、Adaptive Server Enterprise ではサポートされていますが、SQL Anywhere の Transact-SQL ダイアレクトではサポートされていません。

trigger_name に owner が指定された場合は無視されます。SQL Anywhere トリガは、所有者を持ちません。

関連情報

CREATE TRIGGER 文 [982 ページ]

1.4.4.105 CREATE USER 文

データベースユーザまたはグループを作成します。

構文

```
CREATE USER user-name [ IDENTIFIED BY password ]
 [ LOGIN POLICY policy-name ]
```

```
[ FORCE PASSWORD CHANGE { ON | OFF } ]
```

パラメータ

user-name

作成するユーザの名前。

IDENTIFIED BY clause

ユーザのパスワード。パスワードのないユーザは、データベースに接続できません。

policy-name

ユーザに割り当てるログインポリシーの名前。ログインポリシーが指定されない場合、DEFAULT ログインポリシーが適用されます。

FORCE PASSWORD CHANGE clause

ログイン時にユーザが新しいパスワードを指定する必要があるかどうかを制御します。この設定は、ユーザのポリシーの password_expiry_on_next_login オプション設定を上書きします。

備考

ユーザにパスワードを指定する必要はありません。パスワードのないユーザは、データベースに接続できません。これは、グループを作成し、他のユーザはグループユーザ ID を使用してデータベースに接続させないようにする場合に便利です。ユーザ ID は有効な識別子である必要があります。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワードを指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE ANY USER システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ユーザ SQLTester を作成し、パスワードを welcome に設定します。SQLTester ユーザは、初回ログイン時にパスワードを変更する必要があります。

```
CREATE USER SQLTester IDENTIFIED BY welcome
FORCE PASSWORD CHANGE ON;
```

関連情報

CREATE LOGIN POLICY 文 [852 ページ]
ALTER LOGIN POLICY 文 [657 ページ]
ALTER USER 文 [731 ページ]
COMMENT 文 [769 ページ]
DROP LOGIN POLICY 文 [1043 ページ]
DROP USER 文 [1084 ページ]
GRANT 文 [1131 ページ]
GRANT ROLE 文 [1136 ページ]

1.4.4.106 CREATE VARIABLE 文

接続スコープ変数またはデータベーススコープ変数を作成します。

構文

接続スコープ変数の作成

```
CREATE [ OR REPLACE ] VARIABLE identifier data-type [ { = | DEFAULT } initial-value ]
```

```
initial-value : expression
```

データベーススコープ変数の作成

```
CREATE [ OR REPLACE ] DATABASE VARIABLE [ IF NOT EXISTS ] [ owner.]identifier
data-type [ { = | DEFAULT } initial-value ]
```

```
initial-value : expression
```

パラメータ

OR REPLACE 句

OR REPLACE 句を指定すると、指定された変数がすでに存在する場合はその変数が削除され、新しい定義で置き換えられます。OR REPLACE は、現在の値と新しい値のデータ型が同じ場合にのみ、変数の値を置き換えます。

この句は、IF NOT EXISTS 句と一緒に使用しないでください。

owner

このパラメータは、データベーススコープ変数にのみ適用されます。変数の所有者を設定するには、有効なユーザ ID かロール、または PUBLIC を指定します。ユーザを設定すると、そのユーザのみがデータベース変数を使用できます。ロールを設定すると、そのロールを持つユーザがデータベース変数を使用できます。PUBLIC を設定すると、すべてのユーザが変数を使用できます。

`owner` を指定しない場合は、CREATE VARIABLE 文を実行しているユーザが設定されます。

identifier

変数の有効な識別子。

data-type

変数のデータ型。データ型を明示的に設定するか、%TYPE または %ROWTYPE 属性を使用して、データ型をデータベース内の別のオブジェクトのデータ型に設定します。変数のデータ型、またはテーブルやビューのカラムのデータ型に設定する場合は、%TYPE を使用します。カーソル、テーブル、またはビュー内のローから派生する複合データ型に設定する場合は、%ROWTYPE を使用します。

%ROWTYPE および TABLE REF は、データベーススコープ変数のデータ型としてはサポートされていません。

IF NOT EXISTS 句

この句を指定すると、同じ名前を持つデータベーススコープ変数がすでに存在する場合に、エラーを返さずに文が完了します。このパラメータは、所有しているデータベーススコープ変数でのみ使用できます。

この句は、OR REPLACE 句と一緒に使用しないでください。

{ = | DEFAULT } 句

変数のデフォルト値。データベーススコープ変数の場合は、データベースの再起動後の初期値にもなります。

`initial-value` は、`data-type` で定義されたデータ型と一致する必要があります。`initial-value` を指定しなかった場合、SET 文、SELECT ... INTO 文、または UPDATE 文などを使用して別の値が割り当てられるまで、変数には NULL 値が入っています。式を使用して `initial-value` を設定した場合、その式は作成時に評価され、(式ではなく) 結果の定数が保存されます。

備考

変数は、プロシージャ間で値を共有するのに便利です。また、Embedded SQL プログラムから INSERT または UPDATE 文の大きなテキストまたはバイナリオブジェクトを作成するときにも役立ちます。データベーススコープ変数は、接続やデータベースの再起動をまたいで値を共有するのに便利です。

CREATE VARIABLE 構文は、接続のコンテキスト内で使用可能な接続スコープ変数の作成に使用します。

CREATE DATABASE VARIABLE 構文は、別のユーザや別の接続によって使用可能なデータベーススコープ変数の作成に使用します。

OR REPLACE 句は、SQL スクリプトの VAREXISTS 関数の代わりに使用します。

initial-value の変数名を指定した場合、その変数はデータベース内ですでに初期化されている必要があります。

権限

接続スコープ変数: 接続スコープ変数の作成および置換には、権限は不要です。

データベーススコープ変数: 自分が所有するデータベーススコープ変数の作成または置換には、CREATE DATABASE VARIABLE または MANAGE ANY DATABASE VARIABLE システム権限が必要です。他のユーザまたは PUBLIC が所有するデータベーススコープ変数の作成または置換には、MANAGE ANY DATABASE VARIABLE システム権限が必要です。

関連する動作

接続スコープ変数: 接続スコープ変数の作成に関連する動作はありません。

データベーススコープ変数: データベーススコープ変数の作成および置換により、オートコミットが発生します。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例は、VARCHAR(50) 型の site_name というデータベーススコープ変数を作成 (または更新) します。

```
CREATE OR REPLACE DATABASE VARIABLE @site_name VARCHAR(50);
```

この例は、CHAR(66) 型の database_name という PUBLIC が所有するデータベーススコープ変数を作成 (または更新) し、それを特別値 CURRENT DATABASE に設定します。

```
CREATE OR REPLACE DATABASE VARIABLE PUBLIC.@database_name CHAR(66) DEFAULT
CURRENT DATABASE;
```

この例は、VARCHAR(50) データ型の first_name という接続スコープ変数を作成します。

```
CREATE VARIABLE first_name VARCHAR(50);
```

この例は、DATE データ型の birthday という接続スコープ変数を作成します。

```
CREATE VARIABLE birthday DATE;
```

この例は、初期設定が 5 の INT データ型として、v1 という接続スコープ変数を作成します。

```
CREATE VARIABLE v1 INT = 5;
```

この例は、変数 v1 がすでに存在しているかどうかに関係なく、v1 という接続スコープ変数を作成し、値を 10 に設定します。

```
CREATE OR REPLACE VARIABLE v1 INT = 10;
```

この例は、ProductID という接続スコープ変数を作成し、%TYPE 属性を使用してそのデータ型を、Products テーブルの ID カラムのデータ型に設定します。

```
CREATE VARIABLE ProductID Products.ID%TYPE;
```

この例は、ItemsForSale という接続スコープ変数を作成し、%ROWTYPE 属性を使用してそのデータ型を、Products テーブルに定義されているカラムで構成される複合データ型に設定します。その後、別の変数 ItemID を作成し、そのデータ型を、ItemsForSale 変数の ID カラムのデータ型になるよう宣言します。

```
CREATE VARIABLE ItemsForSale Products%ROWTYPE;
CREATE VARIABLE ItemID ItemsForSale.ID%TYPE;
```

関連情報

SQL 変数 [118 ページ]
特別値 [83 ページ]
%TYPE および %ROWTYPE 属性 [110 ページ]
BEGIN 文 [745 ページ]
SQL データ型 [124 ページ]
DROP VARIABLE 文 [1085 ページ]
SET 文 [1325 ページ]
VAREXISTS 関数 [その他] [590 ページ]
SYSDATABASEVARIABLE システムビュー [1859 ページ]

1.4.4.107 CREATE VIEW 文

データベース上にビューを作成します。

構文

```
CREATE [ OR REPLACE ] VIEW
[ owner.]view-name [ ( column-name, ... ) ]
AS query-expression
[ WITH CHECK OPTION ]
```

パラメータ

OR REPLACE 句

OR REPLACE (CREATE OR REPLACE VIEW) を指定すると、ビューが作成されるか、同じ名前の既存のビューが置き換えられます。OR REPLACE 句を使用する場合、既存の権限は保持されますが、ビューの INSTEAD OF トリガは削除されます。

1 つ以上の INSTEAD OF トリガがあるビューで CREATE OR REPLACE VIEW 文を実行すると、エラーが返されます。ビューを変更または削除するには、その前にトリガを削除します。

AS 句

ビューのベースとなる SELECT 文です。SELECT 文はローカルテンポラリテーブルを参照してはなりません。また、`query-expression` には GROUP BY 句、HAVING 句、WINDOW 句、または ORDER BY 句を指定でき、UNION、EXCEPT、INTERSECT または共通テーブル式を含めることができます。

クエリのセマンティックは、SELECT 文で TOP 句または FIRST 句に ORDER BY 句を組み合わせないと、返されるローの順序が定義されないことを示します。

WITH CHECK OPTION 句

WITH CHECK OPTION 句を指定すると、`query-expression` で定義された条件を満たさないビューへの更新や挿入が拒否されます。

備考

ビューは物理的にはテーブルとしてデータベースの中に存在しません。ビューは使用するたびに引き出されます。ビューは、CREATE VIEW 文で指定された SELECT 文の結果として引き出されます。ビューでは、同じ名前のテーブルと区別するために、テーブルの所有者のユーザ ID を指定することが推奨されます。

SELECT、DELETE、UPDATE、INSERT 文の中では、ビュー名をテーブル名の代わりに使用できます。

SELECT* は、メインクエリ、サブクエリ、派生テーブル、または、CREATE VIEW 文の subselect で使用できます。

クエリでは、ORDER BY 句がなくても、TOP n、FIRST、または LIMIT の句を指定できます。最大で指定数のローが返りますが、返ってくるローの順序は定義されません。そのため、ORDER BY は指定できますが、必須ではありません。クエリにビューが使用されている場合、たとえ FROM 句内に他のテーブルがなくても、ビュー内の ORDER BY は、クエリ内のローの順序に作用しません。このため、ORDER BY をビューに入れるのは、TOP n、FIRST、または LIMIT の句に含めるローを選択する必要がある場合だけにしてください。それ以外の場合、ORDER BY 句に効果はなく、データベースサーバはこの句を無視します。

ビューを定義する `query-expression` が GROUP BY 句、WINDOW 句、集合関数を含まないか、または集合演算子 (UNION、INTERSECT、EXCEPT) を伴わない場合は、ビューを更新できます。ビューを更新すると、基本となるテーブルも更新されます。

ビューのカラムには `column-name` リスト内で指定した名前が与えられます。column-name リストを指定しないと、ビューのカラムは SELECT リスト項目から名前が与えられます。SELECT リストのすべての項目がユニークな名前になるように指定します。SELECT リスト項目からの名前を使用するには、各項目を単純なカラム名にするか、エイリアス名を指定します。

データベースサーバでは、`query-expression` の SELECT リスト内の名前のない式を、CREATE VIEW 文で参照できます。`query-expression` の SELECT リストの名前のない式には、名前 **expression** が割り当てられ、このような式が複数

存在する場合、整数値と連結されます。たとえば、次の文は 3 つのカラム (expression、expression1、および expression2) を持つビュー V を定義し、これらの名前は作成されたビュー V の SYSCOLUMN システムビューに表示されます。

```
CREATE VIEW V AS
   SELECT DATEADD( DAY, 1, NOW() ), DATEADD( DAY, 2, NOW() ), DATEADD( DAY, 2,
NOW() )
   FROM SYS.DUMMY;
```

名前のない SELECT リスト式を持つ他のビューにも同一の名前が割り当てられるため、これらの生成された名前に頼ることはお奨めしません。

一般的に、ビューはカタログに定義されているテーブルやビュー (およびその属性) を参照します。ただし、ビューは SQL 変数も参照できます (TABLE REF 変数を除く)。定義で変数が使用される場合、ビューを参照するクエリが実行されると、変数の代わりに SQL 変数の値が使用されます。SQL 変数を参照するビューは、変数がビューを実行するためのパラメータとして機能することから、パラメータ化されたビューと呼ばれます。パラメータ化されたビューは、等価の SELECT ブロックの本体を FROM 句内の派生テーブルとしてクエリに埋め込む手段を提供します。パラメータ化されたビューは、ビューで参照されている SQL 変数を入力パラメータとして取るようなストアドプロシージャに埋め込まれたクエリで便利です。

CREATE VIEW 文が実行されるときに SQL 変数が存在している必要はありません。ただし、ビューを参照するクエリを実行するときに SQL 変数が定義されていないと、カラム (変数) が見つからないことを示すエラーが返されます。

TABLE REF 型の変数 (テーブル参照変数) は、ビュー定義の最も外側の SELECT リストで許可されていません。

権限

ユーザ本人が所有するビューを作成するには、CREATE VIEW システム権限が必要です。他のユーザが所有するビューを作成するには、CREATE ANY VIEW または CREATE ANY OBJECT のシステム権限が必要です。

既存のビューを置き換えるには、そのビューの所有者であるか、または次のいずれかの権限が必要です。

- CREATE ANY VIEW および DROP ANY VIEW システム権限。
- CREATE ANY OBJECT および DROP ANY OBJECT システム権限。
- ALTER ANY OBJECT または ALTER ANY VIEW システム権限。

ビューによって参照されるテーブルが他のユーザに所有されている場合は、それらのテーブルに対する SELECT オブジェクトレベル権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

コア機能ですが、SELECT 文に組み込まれている一部のビュー機能はオプションの言語機能です。ビュー定義で最上位レベルの SELECT 文で ORDER BY 句を指定する機能は、オプションの言語機能 F852 です。SELECT TOP または

LIMIT を使用したビューの結果セットの制限は、オプションの言語機能 F859 です。ビューの SELECT 文に、集合、DISTINCT が関係する派生テーブル、または集合演算子 (INTERSECT、EXCEPT、UNION) が含まれているなど、更新できないビュー における WITH CHECK OPTION の指定は、オプションの言語機能 T111 です。

パラメータ化されたビュー、オプションの OR REPLACE 構文、および名前のない SELECT リスト式の名前の自動生成といった拡張機能は、ANSI/ISO SQL 標準にありません。

例

次の例は、男性従業員の情報のみを表示するビューを作成します。このビューはベーステーブルと同じカラム名を持ちます。

```
CREATE VIEW MaleEmployees
AS SELECT *
FROM GROUPO.Employees
WHERE Sex = 'M';
```

次の例は、従業員と所属する部署を表示するビューを作成します。

```
CREATE VIEW EmployeesAndDepartments
  AS SELECT Surname, GivenName, DepartmentName
  FROM GROUPO.Employees JOIN GROUPO.Departments
  ON Employees.DepartmentID = Departments.DepartmentID;
```

次の例は、前の例で作成された EmployeesAndDepartments ビューを置き換えます。置き換え後のビューには、各従業員の住所の市、州、国が表示されます。

```
CREATE OR REPLACE VIEW EmployeesAndDepartments
  AS SELECT Surname, GivenName, City, State, Country
  FROM GROUPO.Employees JOIN GROUPO.Departments
  ON Employees.DepartmentID = Departments.DepartmentID;
```

次の例は、パラメータ化されたビューを変数 var1 と var2 に基づいて作成します。この 2 つは Employees テーブルの属性でも Departments テーブルの属性でもありません。

```
CREATE VIEW EmployeesByState
  AS SELECT Surname, GivenName, DepartmentName
  FROM GROUPO.Employees JOIN GROUPO.Departments
  ON Employees.DepartmentID = Departments.DepartmentID
  WHERE Employees.State = var1 and Employees.Status = var2;
```

変数は、変数が式として許可されている任意のコンテキストで、ビューの SELECT 文に出現させることができます。たとえば、次のパラメータ化されたビューでは、パラメータ var1 を LIKE 述部のパターンとして使用しています。

```
CREATE VIEW ProductsByDescription
  AS SELECT *
  FROM GROUPO.Products
  WHERE Products.Description LIKE var1;
```

このビューを使用するには、このビューを参照するクエリを実行する前に変数 var1 を定義します。たとえば、次の BEGIN 文はプロシージャ、関数、またはバッチ文に使用できます。

```
BEGIN
DECLARE var1 CHAR(20);
SET var1 = '%cap%';
SELECT * FROM ProductsByDescription
END
```

関連情報

ALTER VIEW 文 [734 ページ]
SELECT 文 [1291 ページ]

1.4.4.108 DEALLOCATE DESCRIPTOR 文 [ESQL]

SQL 記述子領域に関連付けられているメモリを解放します。

構文

```
DEALLOCATE DESCRIPTOR descriptor-name
```

```
descriptor-name : identifier
```

備考

データ項目、インジケータ変数、構造体そのものなど、記述子領域に関連付けられているすべてのメモリを解放します。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

DEALLOCATE DESCRIPTOR は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部です。

関連情報

ALLOCATE DESCRIPTOR 文 [ESQL] [628 ページ]
SET DESCRIPTOR 文 [ESQL] [1303 ページ]

1.4.4.109 DEALLOCATE 文

この文は SQL Anywhere では機能しないので、無視されます。これは Adaptive Server Enterprise と Microsoft SQL Server との互換性のために用意されています。この文の詳細については、Adaptive Server Enterprise または Microsoft SQL Server のマニュアルを参照してください。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.110 宣言セクション [ESQL]

Embedded SQL プログラムでホスト変数を宣言します。ホスト変数を使って、データベースとデータを交換します。

構文

```
EXEC SQL BEGIN DECLARE SECTION;
C declarations
EXEC SQL END DECLARE SECTION;
```

備考

宣言セクションは、BEGIN DECLARE SECTION と END DECLARE SECTION 文で囲まれた C 変数宣言のセクションです。宣言セクションを使うと、SQL プリプロセッサはホスト変数として使われる C 変数を認識します。すべての C 宣言が宣言セクションの中で有効なわけではありません。

権限

なし。

標準

ANSI/ISO SQL 標準

コア機能。

例

```
EXEC SQL BEGIN DECLARE SECTION;
char *surname, initials[5];
int dept;
EXEC SQL END DECLARE SECTION;
```

関連情報

BEGIN 文 [745 ページ]

1.4.4.111 DECLARE CURSOR 文 [ESQL] [SP]

カーソルを宣言します。

構文

Embedded SQL

```
DECLARE cursor-name
[ UNIQUE ]
[ NO SCROLL
  | DYNAMIC SCROLL
  | SCROLL
  | INSENSITIVE
  | SENSITIVE ]
CURSOR FOR
{ select-statement
| statement-name
| call-statement }
```

ストアドプロシージャ

```
DECLARE cursor-name
[ NO SCROLL
  | DYNAMIC SCROLL
  | SCROLL
  | INSENSITIVE
  | SENSITIVE ]
CURSOR
{ FOR select-statement
| FOR call-statement
| USING variable-name }
```

```
cursor-name :  identifier
```

```
statement-name :  identifier | hostvar
```

```
variable-name :  identifier
```

パラメータ

UNIQUE 句

カーソルが UNIQUE として宣言されている場合、クエリは各ローをユニークに識別するのに必要なすべてのカラムを強制的に返します。このため、プライマリキーまたはユニークなテーブル制約内のカラムのすべてが、確実に返ることになります。必須であってもクエリに指定されていなかったカラムは、結果セットに追加されます。

UNIQUE カーソル上で実行された DESCRIBE は、インジケータ変数の中に次の追加のオプションを設定します。

DT_KEY_COLUMN

カラムはローに対するキーの一部です。

DT_HIDDEN_COLUMN

カラムがクエリに追加されました。カラムはローをユニークに識別するために必要です。

NO SCROLL 句

NOSCROLL として宣言されたカーソルは、FETCHNEXT と FETCHRELATIVE0 の各シーク操作を使用して、結果セットの前方移動に制限されます。

カーソルがローを出た後は、そのローに戻ることができないため、カーソルに sensitivity の制限はありません。NO SCROLL カーソルを要求されると、データベースサーバ は最も効率的な種類のカーソルである asensitive カーソルを提供します。

DYNAMIC SCROLL 句

DYNAMIC SCROLL はデフォルトカーソルタイプです。DYNAMIC SCROLL カーソルでは、FETCH 文のすべてのフォーマットを使用できます。

DYNAMIC SCROLL カーソルを要求されると、データベースサーバは asensitive カーソルを提供します。カーソルを使用する場合、効率と一貫性の間には常にトレードオフ関係があります。Asensitive カーソルを使用すると、パフォーマンスは向上しますが一貫性が低下します。

SCROLL 句

SCROLL として宣言されたカーソルは、FETCH 文のすべてのフォーマットを使用できます。SCROLL カーソルを要求されると、データベースサーバは value-sensitive カーソルを提供します。value-sensitive カーソルを使用すると、前回フェッチの後、基本となるローが更新または削除された場合、次のフェッチに対して、警告またはエラーが返されることがあります。

データベースサーバは、結果セットのメンバーシップが保証されるような方法で value-sensitive カーソルを実行する必要があります。DYNAMIC SCROLL カーソルの方が効率的です。SCROLL カーソルの一貫した動作が必要でない場合は、DYNAMIC SCROLL カーソルを使用してください。

INSENSITIVE 句

INSENSITIVE として宣言されたカーソルには、それを開いたときに決定されるメンバーシップがあります。テンポラリテーブルは、すべてのオリジナルローのコピーから作成されます。INSENSITIVE カーソルからの FETCH では、同時に実行

されているトランザクションの別の INSERT、UPDATE、または DELETE 文、および同じトランザクション内からの別の更新操作の影響を受けません。INSENSITIVE カーソルは更新できません。

SENSITIVE 句

SENSITIVE として宣言されたカーソルは、結果セットのメンバーシップまたは値の変更に依存します。

FOR statement-name 句

PREPARE 文を使用して文に名前を付けます。作成された SELECT または CALL のためにだけカーソルを宣言できます。SQL プリプロセッサの -m HISTORICAL オプションが指定されている場合を除き、PREPARE 文に指定されているカーソル更新可能性がカーソルに使用されます。

USING variable-name 句

ストアドプロシージャの内部でのみ使用します。この変数はカーソルの SELECT 文を含む文字列です。この変数は DECLARE を処理するときに使用するため、プロシージャのパラメータにするか、この変数に値が代入された後に別の BEGIN...END 内にネストされる必要があります。

備考

カーソルはクエリの結果を操作する主要な手段です。DECLARE CURSOR 文は、SELECT 文または CALL 文に対して指定された名前を持つカーソルを宣言します。Watcom SQL のプロシージャ、トリガ、またはバッチの場合、DECLARE CURSOR 文は、BEGIN キーワードの直後に、他の宣言とともに指定してください。カーソル名はユニークにしてください。

カーソルが複合文内で宣言される場合、(Watcom SQL または Transact-SQL 複合文のいずれで宣言されていても) その複合文を実行している間のみ存在します。

単一の文が処理されるときは、重複していないスコープで各カーソルが宣言されていても、すべての DECLARE CURSOR 文でユニークな名前を使用する必要があります。ただし、カーソルはそれを宣言した複合文の中でのみ使用できます。

DECLARE CURSOR 文で指定したカーソルのタイプで、その文のクエリオプティマイザによって選択される実行プランを決定できます。たとえば、SELECT 文の INSENSITIVE カーソルは、カーソルを開いたときに SELECT 文の結果セットが完全に実体化されることを要求します。また、カーソルのタイプは基本となる文の特性と一致している必要があります。カーソルタイプと文に不一致があった場合、カーソルタイプが自動的に変更されることがあります。たとえば、定義上 INSENSITIVE カーソルは読み込み専用のため、INSENSITIVE カーソル宣言は FOR UPDATE を指定している更新可能な SELECT 文とは競合します。このような場合、カーソルタイプは、カーソルが開かれたときに自動的に INSENSITIVE から更新可能な value-sensitive カーソルに変更されます。

カーソル宣言に組み込まれている SELECT 文の更新可能性は、指定されていない場合 ansi_update_constraints オプションの設定によって決定されます。

権限

なし。

関連する動作

基本となる文の要件を満たすようにカーソルタイプを変更する必要がある場合、カーソルが開かれたときに警告が返されます。

標準

ANSI/ISO SQL 標準

DECLARE CURSOR はコア機能です。FOR UPDATE に SCROLL または NO SCROLL を付けて指定する機能は、オプションの ANSI/ISO SQL 言語機能 F831、"Full cursor update" です。Embedded SQL プログラムでの DECLARE CURSOR の使用は、オプションの ANSI/ISO SQL 言語機能 B031 によって構成されています。いくつかのカーソルタイプもオプションの ANSI/ISO SQL 機能です。これらを以下に示します。

- INSENSITIVE カーソルは、ANSI/ISO SQL 標準のオプションの SQL 言語機能 F791 です。
- SENSITIVE カーソルは、ANSI/ISO SQL 標準のオプションの SQL 言語機能 F231 です。
- スクロール可能なカーソルは、ANSI/ISO SQL 標準のオプションの SQL 言語機能 F431 です。

ソフトウェアでは、以下のように多数の DECLARE CURSOR の拡張がサポートされています。

- ソフトウェアでは、ANSI/ISO SQL 標準において DECLARE CURSOR 文の句として定義されている FOR UPDATE 句の複数の拡張がサポートされます。
- WITH HOLD は、この ANSI/ISO SQL 標準の定義に従った DECLARE CURSOR 文の句としてではなく、OPEN 文の句として指定されます。
- ANSI/ISO SQL 標準では、カーソルの感知性とスクロール動作の概念を区別していますが、ソフトウェアでは、歴史的な理由から、この 2 つを組み合わせています。ソフトウェアでは、NO SCROLL として宣言されているカーソルを除き、すべてのカーソルは前後にスクロール可能です。
- DYNAMIC SCROLL と UNIQUE は、ANSI/ISO SQL 標準にはありません。DYNAMIC SCROLL は、ANSI/ISO SQL 標準の ASENSITIVE として宣言されるカーソルと似た動作をします。
- CALL 文を介して、または USING 句を使用してカーソルを宣言する機能は、ANSI/ISO SQL 標準にはありません。

Transact-SQL

DECLARE CURSOR は、Adaptive Server Enterprise によってサポートされますが、いくつかの動作が異なります。Adaptive Server Enterprise は、ANSI/ISO SQL 標準と同様に、スクロール動作と感知性を区別します。Adaptive Server Enterprise では、カーソル感知性オプションは SEMI-SENSITIVE、INSENSITIVE、またはデフォルト (ASENSITIVE と類似) です。Adaptive Server Enterprise では、NO SCROLL カーソルがデフォルトであり、スクロール可能なカーソルはすべて読み込み専用です。DECLARE CURSOR 文のいくつかの機能は、Adaptive Server Enterprise ではサポートされません。これらのコードを以下に示します。

- Adaptive Server Enterprise では、SQL Anywhere のカーソル同時実行性句はサポートされません。フェッチされるローに対するロックを取得するには、HOLDLOCK テーブルヒントを使用してください。
- Adaptive Server Enterprise は、DYNAMIC SCROLL カーソルおよび UNIQUE カーソルをサポートしません。DYNAMIC SCROLL は、Adaptive Server Enterprise のデフォルトカーソルの動作と似ています。
- CALL 文を介して、または USING 句を使用してカーソルを宣言する機能は、Adaptive Server Enterprise ではサポートされません。

Adaptive Server Enterprise では、Transact-SQL プロシージャとファンクションに対して、同じカーソル名を使用する複数の DECLARE CURSOR 文を指定できます。Adaptive Server Enterprise では、DEALLOCATE CURSOR 文は、後

続の OPEN 文が以前宣言された正しいカーソルを参照できるように、現在のスコープからカーソルを削除するために使用されます。この機能は SQL Anywhere ではサポートされていません。SQL Anywhere では、指定されたスコープ内のすべてのカーソルは、ユニークな名前を持つ必要があります。Transact-SQL ダイアレクトのプロシージャに同じ名前の複数のカーソル宣言が含まれている場合、プロシージャが解析されるとき、エラーは発生しません。ただし、実行時に、同じカーソル名の 2 つ目の DECLARE CURSOR 文が実行されると、エラーが発生します。

Open Client 接続および jConnect 接続の TDS Wire Protocol は、完全にスクロール可能な結果セットを実装しないことに注意してください。カーソルで後方にスクロールするときに、TDS クライアントによってすでに取り出されている、プリフェッチされたローのウィンドウ内に必要なローがある場合、FETCH 要求はすぐに履行される可能性があります。ただし、必要なローがこのウィンドウの外にある場合、カーソルの SELECT 文が再実行されることがあります。

例

次の例は、Embedded SQL 内のスクロールカーソルを宣言する方法を示します。

```
EXEC SQL DECLARE cur_employee SCROLL CURSOR
FOR SELECT * FROM GROUPO.Employees;
```

次の例は、Embedded SQL 内の準備文のためのカーソルを宣言する方法を示します。

```
EXEC SQL PREPARE employee_statement
FROM 'SELECT Surname FROM GROUPO.Employees'FOR READ ONLY;
EXEC SQL DECLARE cur_employee CURSOR
FOR employee_statement;
```

次の例は、ストアドプロシージャのカーソルの使用方法を示します。

```
BEGIN
  DECLARE cur_employee CURSOR FOR
      SELECT Surname
      FROM GROUPO.Employees;
  DECLARE name CHAR(40);
  OPEN cur_employee;
  lp: LOOP
    FETCH NEXT cur_employee INTO name;
    IF SQLCODE <> 0 THEN LEAVE lp END IF;
    ...
  END LOOP;
  CLOSE cur_employee;
END
```

この例は、プロシージャへのパラメータを使用した USING 句の使用法を示します。

```
CREATE FUNCTION GetRowCount( IN qry LONG VARCHAR )
RETURNS INT
BEGIN
  DECLARE crsr CURSOR USING qry;
  DECLARE rowcnt INT;
  SET rowcnt = 0;
  OPEN crsr;
  lp: LOOP
    FETCH crsr;
    IF SQLCODE <> 0 THEN LEAVE lp END IF;
    SET rowcnt = rowcnt + 1;
  END LOOP;
  CLOSE crsr;
  RETURN rowcnt;
END;
```

この例は、variable-name に値を代入した後、ネストした BEGIN...END 内で USING 句を使用する方法を示します。

```
CREATE PROCEDURE get_table_name(
  IN id_value INT, OUT tabname CHAR(128)
)
BEGIN
  DECLARE qry LONG VARCHAR;
  SET qry = 'SELECT table_name FROM SYS.SYSTAB ' ||
            'WHERE table_id=' || string(id_value);
  BEGIN
    DECLARE crsr CURSOR USING qry;
    OPEN crsr;
    FETCH crsr INTO tabname;
    CLOSE crsr;
  END
END;
```

次の例は、文の中の 2 つのカーソル名がユニークではないためエラーを返します。

```
BEGIN
   BEGIN
     DECLARE MyCursor DYNAMIC SCROLL CURSOR FOR SELECT 1;
   END;
   BEGIN
     DECLARE MYCursor DYNAMIC SCROLL CURSOR FOR SELECT 2;
   END;
END;
```

次の例は、カーソルを開こうとした文の中でそのカーソルが宣言されていないためエラーを返します。

```
BEGIN
   BEGIN
     DECLARE MyCursor DYNAMIC SCROLL CURSOR FOR SELECT 1;
   END;
   BEGIN
     OPEN MyCursor;
   END;
END;
```

関連情報

PREPARE 文 [ESQL] [1238 ページ]
OPEN 文 [ESQL] [SP] [1218 ページ]
EXPLAIN 文 [ESQL] [1099 ページ]
SELECT 文 [1291 ページ]
CALL 文 [756 ページ]
FOR 文 [1106 ページ]

1.4.4.112 DECLARE LOCAL TEMPORARY TABLE 文

ローカルテンポラリテーブルを宣言します。

構文

```
DECLARE LOCAL TEMPORARY TABLE table-name
( { column-definition [ column-constraint ... ] | table-constraint | pctfree |
like-clause }, ... )
like-clause | as-clause
[ ON COMMIT { DELETE | PRESERVE } ROWS
    | NOT TRANSACTIONAL ]
```

```
pctfree : PCTFREE percent-free-space
```

```
percent-free-space : integer
```

```
like-option :
{ INCLUDING | EXCLUDING } option [ ,... ]
```

```
option :
{ IDENTITY
 | DEFAULTS
 | CONSTRAINTS
 | INDEXES
 | PRIMARY KEY
 | FOREIGN KEYS
 | COMMENTS
 | STORAGE
 | ALL }
```

```
as-clause :
[ (column-name, ... ) ] AS ( select-statement )
```

パラメータ

LIKE 句

`like-option` のデフォルトは、すべてのオプションを除 外 (EXCLUDING) します。`source-table` がビューである場合、`like-option` は適用されていなと無視されます。

元のテーブルから次のカラム属性のいずれかを新しいテーブルに含めるには、INCLUDE を指定します。

IDENTITY デフォルトのオートインクリメント情報を含めます。
DEFAULTS オートインクリメント以外のデフォルトの値式を含めます。
CONSTRAINTS カラムとテーブルの検査、および一意性制約を含めます。
INDEXES プライマリキーと外部キー以外のインデックスを含めます。
PRIMARY KEY プライマリキーを含めます。
FOREIGN KEY 外部キーを含めます。
COMMENTS カラムについてのコメントを含めます。

STORAGE 圧縮設定を含めます。

AS 句 SELECT 文に基づくテーブルを複製するには、この句を使用します。

ON COMMIT 句

デフォルトでは、テンポラリテーブルのローは COMMIT のときに削除されます。ON COMMIT 句を使用すると、COMMIT のときにローを保護できます。

NOT TRANSACTIONAL 句

この句を使用して作成されたテーブルは、COMMIT と ROLLBACK のいずれの影響も受けません。状況によっては、NOT TRANSACTIONAL 句を使用するとパフォーマンスが向上します。これは、トランザクション単位でないテンポラリテーブルでの操作では、ロールバックログにエントリが作成されないためです。たとえば、テンポラリテーブルを使用するプロシージャが COMMIT や ROLLBACK の介入を受けずに繰り返し呼び出される場合、NOT TRANSACTIONAL が有用です。

備考

REFERENCES `column-constraint` および FOREIGN KEY table-constraint は、ローカルテンポラリテーブルに対して使用できません。

DECLARE LOCAL TEMPORARY TABLE 文はテンポラリテーブルを宣言します。

DECLARE LOCAL TEMPORARY TABLE...LIKE 構文は、別のテーブル定義に直接基づく新しいテーブルを宣言します。カラム、制約、および LIKE 句を追加してテーブルを複製したり、SELECT 文に基づいてテーブルを作成したりすることもできます。

DECLARE LOCAL TEMPORARY TABLE を使用して作成されたテーブルは、システムカタログの SYSTABLE ビューに表示されません。

宣言されたテンポラリテーブルのローは、テーブルが明示的に削除されたとき、またはスコープ外になったときに削除されます。TRUNCATE または DELETE を使用して、明示的にローを削除することもできます。

複合文内で宣言されたローカルテンポラリテーブルは、複合文内に存在します。それ以外の場合、宣言されたローカルテンポラリテーブルは接続の終わりまで存在します。

同じスコープ内にある 2 つのローカルテンポラリテーブルは、同じ名前にはできません。ベーステーブルと同じ名前のテンポラリテーブルを作成すると、ローカルテンポラリテーブルのスコープが終了した時点で、そのベーステーブルは、その接続内でのみ参照できるようになります。接続では、既存のテンポラリテーブルと同じ名前のベーステーブルを作成できません。

プロシージャの完了後も保持されるローカルテンポラリテーブルをプロシージャで作成するには、代わりに CREATE LOCAL TEMPORARY TABLE 文を使用します

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

DECLARE LOCAL TEMPORARY TABLE は、オプションの言語機能 F531 の一部です。PCTFREE 句と NOT TRANSACTIONAL 句は、標準にはありません。文に定義されたカラム定義と制約定義には、標準にはない拡張機能構文を含めることもできます。ANSI/ISO SQL 標準では、DECLARE LOCAL TEMPORARY TABLE 文によって作成されるテーブルをシステムカタログに表示することが規定されていますが、ソフトウェアでは表示されません。

Transact-SQL

DECLARE LOCAL TEMPORARY TABLE は、Adaptive Server Enterprise ではサポートされません。SAP Adaptive Server Enterprise では、特殊文字 # で始まるテーブル名を使用して CREATE TABLE 文でテンポラリテーブルを作成します。

例

次の例は、ストアドプロシージャ内のテンポラリテーブルの宣言方法を示します。

```
BEGIN
  DECLARE LOCAL TEMPORARY TABLE TempTab ( number INT );
  ...
END
```

例

次の例の 2 番目の文は、myT2 テーブルを作成し、%TYPE 属性を使用して、その myColumn カラムのデータ型を myT1 の last_name カラムのデータ型に設定します。null 入力可能性などの追加属性は適用されないため、myT2.myColumn には、myT1.last_name が持つ NOT NULL 制限と同じ制限は適用されません。

```
CREATE TABLE myT1
( first_name   CHAR(20),
   last_name   VARCHAR NOT NULL );
CREATE TABLE myT2
( myColumn myT1.last_name%TYPE );
```

次の例では、ローカルテンポラリテーブルを宣言してから、最初のテーブルの定義に基づき 2 番目のローカルテンポラリテーブルを宣言します。

```
DECLARE LOCAL TEMPORARY TABLE table1 ( ID INT NOT NULL DEFAULT AUTOINCREMENT,
NAME LONG VARCHAR ) ;
DECLARE LOCAL TEMPORARY TABLE table2 ( ADDRESS LONG VARCHAR ) ;
```

次の文は、データのない table1 と同様に、ローカルテンポラリテーブルを宣言します。

```
DECLARE LOCAL TEMPORARY TABLE table3 LIKE table1 INCLUDING IDENTITY ;
```

次の文は、table1 と同様にローカルテンポラリテーブルを宣言しますが、追加カラムがあります。

```
DECLARE LOCAL TEMPORARY TABLE table4 ( LIKE table1 INCLUDING IDENTITY, LIKE
table2, phone LONG VARCHAR );
```

次の文は、table1 にデータがあり、さらに各ローの phone カラムが '555-5555' であるローカルテンポラリテーブルを宣言します。

```
DECLARE LOCAL TEMPORARY TABLE table5 AS ( SELECT * , '555-5555' AS phone FROM
table1 ) WITH DATA ;
```

関連情報

CREATE TABLE 文 [952 ページ]
CREATE LOCAL TEMPORARY TABLE 文 [850 ページ]

1.4.4.113 DECLARE 文

複合文 (BEGIN...END) 内で SQL 変数 (接続スコープ) または例外を宣言します。

構文

変数の宣言

```
DECLARE identifier [, ... ] data-type
[ { = | DEFAULT } initial-value ]
```

```
initial-value : expression
```

例外の宣言

```
DECLARE exception-name EXCEPTION
FOR SQLSTATE [ VALUE ] string
```

パラメータ

identifier

変数の有効な識別子。

data-type

変数のデータ型。データ型を明示的に設定するか、%TYPE 属性または %ROWTYPE 属性を使用して設定できます。変数のデータ型、またはテーブルやビューのカラムのデータ型に設定する場合は、%TYPE を使用します。カーソル、テーブル、またはビュー内のローから派生する複合データ型に設定する場合は、%ROWTYPE を使用します。

DEFAULT 句

変数のデフォルト値。initial-value を指定した場合、データ型は、data-type で定義された型と一致する必要があります。

initial-value

変数のデフォルトおよび初期値。initial-value は、data-type で定義されたデータ型と一致する必要があります。initial-value を指定しなかった場合、別の値が割り当てられるまで、変数には NULL 値が入っています。

備考

DECLARE variable-name:プロシージャ、トリガ、またはバッチの本体で使用される変数は、DECLARE を使用して宣言できます。変数は、宣言される複合文の間持続します。

initial-value の変数名を指定した場合、その変数はデータベース内ですでに初期化されている必要があります。

Watcom SQL プロシージャの本文またはトリガは複合文です。また、カーソルの宣言 (DECLARE CURSOR) など他の宣言では、BEGIN キーワードの直後に変数を宣言する必要があります。Transact-SQL プロシージャまたはトリガには、そのような制約はありません。

DECLARE exception-name **EXCEPTION:**この構文を使用して、複合文 (BEGIN...END) 内で SQL 言語の例外の変数を宣言します。この変数は、たとえば、実行中に取得される SQLSTATE との比較に使用したり、SIGNAL 文で使用したり、例外ハンドラ内の例外ケースの一部として使用したりできます。

権限

なし

関連する動作

なし。

標準

ANSI/ISO SQL 標準

変数を宣言する構文 - 永続的ストアドモジュール機能。例外を宣言する構文 - ソフトウェアでサポートされている例外宣言の形式 (つまり DECLARE EXCEPTION 文) は標準にありません。ANSI/ISO SQL 標準では、例外はキーワード DECLARE HANDLER を使用するハンドラ宣言で指定されます。DECLARE...EXCEPTION 構文は、T-SQL プロシージャでは使用できません。

Transact-SQL

例外を宣言する構文は、Transact-SQL 複合文およびプロシージャでは使用できません。

例

次のバッチは、DECLARE 文の使用法を示し、メッセージをデータベースサーバメッセージウィンドウに表示します。

```
BEGIN
  DECLARE varname CHAR(61);
  SET varname = 'Test name';
  MESSAGE varname;
END
```

次の例は、以下の変数を宣言します。

- INT、初期設定値 5 の v1。
- CHAR(10)、初期値 abc の v2 と v3。

```
BEGIN
   DECLARE v1 INT = 5;
   DECLARE v2, v3 CHAR(10) = 'abc';
            // ...
END
```

次のプロシージャは SQLSTATE 比較で使用する例外を宣言します。

```
CREATE PROCEDURE HighSales (IN cutoff INT, OUT HighValues INT)
BEGIN
   DECLARE err_notfound EXCEPTION FOR
      SQLSTATE '02000';
   DECLARE curThisCust CURSOR FOR
      SELECT CAST( sum( SalesOrderItems.Quantity *
            Products.UnitPrice ) AS INTEGER) VALUE
      FROM Customers
            LEFT OUTER JOIN SalesOrders
            LEFT OUTER JOIN SalesOrderItems
            LEFT OUTER JOIN Products
      GROUP BY CompanyName;
   DECLARE ThisValue INT;
   SET HighValues = 0;
   OPEN curThisCust;
   CustomerLoop:
   LOOP
      FETCH NEXT curThisCust
         INTO ThisValue;
      IF SQLSTATE = err_notfound THEN
         LEAVE CustomerLoop;
      END IF;
      IF ThisValue > cutoff THEN
         SET HighValues = HighValues + ThisValue;
      END IF;
END LOOP CustomerLoop;
   CLOSE curThisCust;
END;
```

次の複合文は、SIGNAL および例外ハンドラで使用する例外を宣言します。

```
BEGIN
   DECLARE err_div_by_0 EXCEPTION FOR
      SQLSTATE '22012';
   DECLARE curQuantity CURSOR FOR
      SELECT Quantity
      FROM SalesOrderItems
      WHERE ProductID = 300;
```

```
   DECLARE Quantities INT;
   DECLARE altogether INT;
   SET Quantities = 0;
   SET altogether = 0;
   OPEN curQuantity;
   LOOP
      FETCH NEXT curQuantity
         INTO Quantities;
      IF SQLSTATE = '02000' THEN
         SIGNAL err_div_by_0;
      END IF;
      SET altogether = altogether + Quantities;
   END LOOP;
EXCEPTION
   WHEN err_div_by_0 THEN
    CLOSE curQuantity;
    SELECT altogether;
    return;
   WHEN OTHERS THEN
    RESIGNAL;
END;
```

次の文は、プロシージャ DepartmentsCloseToCustomerLocation を作成し、%TYPE 属性を使用して、その IN パラメータを Customers テーブル内の ID カラムのデータ型に設定します。

```
CREATE OR REPLACE PROCEDURE DepartmentsCloseToCustomerLocation( IN customer_ID
Customers.ID%TYPE )
  BEGIN
    DECLARE cust_rec Customers%ROWTYPE;
    SELECT City, State, Country
    INTO cust_rec.City, cust_rec.State, cust_rec.Country
    FROM Customers
    WHERE ID = customer_ID;
    SELECT Employees.Surname, Employees.GivenName, Departments.DepartmentName
    FROM Employees JOIN Departments
    ON Departments.DepartmentHeadID = Employees.EmployeeID
    WHERE Employees.City = cust_rec.City
    AND Employees.State = cust_rec.State
    AND Employees.Country = cust_rec.Country;
  END;
CALL DepartmentsCloseToCustomerLocation(158);
```

次の例では、%ROWTYPE 属性を使用して変数 @a_product を作成し、Products テーブルから変数にデータを挿入します。

```
CREATE OR REPLACE PROCEDURE CheckStock (
    IN @id Products.ID%TYPE
)
   BEGIN
    DECLARE @a_product Products%ROWTYPE;
    SET (@a_product).ID = 200;
   END;
```

関連情報

特別値 [83 ページ]
%TYPE および %ROWTYPE 属性 [110 ページ]
SQL データ型 [124 ページ]

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
BEGIN 文 [745 ページ]
SQLSTATE 特別値 [105 ページ]

1.4.4.114 DELETE 文 (位置付け) [ESQL] [SP]

カーソルの現在位置のデータを削除します。

構文

```
DELETE [ [FROM ]table ] WHERE CURRENT OF cursor-name
```

```
cursor-name :  identifier | hostvar
```

```
table : [ owner.]table-or-view [ [ AS ] correlation-name ]
```

```
owner : identifier
```

```
table-or-view : identifier
```

```
correlation-name : identifier
```

備考

この形式の DELETE 文は、指定されたカーソルの現在のローを削除します。現在のローは、カーソルからフェッチされた最後のローと定義されます。

ローを削除するテーブルは次のように決定します。

- FROM 句が含まれない場合、カーソルは単一テーブルだけにあります。
- カーソルがジョインされたクエリ用の場合 (ジョインがあるビューの使用を含めて)、FROM 句が使われます。指定したテーブルの現在のローだけが削除されます。ジョインに含まれた他のテーブルは影響を受けません。
- FROM 句が含まれている場合は、table によって、カーソル内の更新可能なテーブルが明確に指定されている必要があります。correlation-name が指定されている場合、サーバは基本となるカーソルで指定されている相関名とその相関名を照合しようとします。DELETE 文に相関名が指定されておらず、テーブル所有者が指定されていない場合、サーバは table-or-view と基本となるカーソルの更新可能なテーブルを照合しようとします。table-or-view は最初に相関名に対して照合されます。
  - 相関名が基本となるカーソルに存在する場合、table-or-view は対応する相関名と照合されることがあります。
  - 相関名が存在しない場合、table-or-view はカーソルのテーブル名と明確に一致する必要があります。
- FROM 句が含まれており、かつテーブル所有者が指定されている場合、table は基本となるカーソルの更新可能なテーブルと明確に一致する必要があります。
- 位置付け DELETE 文はビューでカーソルを開くときに使用できます。ただし、ビューが更新可能である場合にかぎられます。

権限

テーブルの所有者であるか、テーブルに対する DELETE 権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

DELETE 文 (位置付け) は、コア機能です。Embedded SQL プログラム内から位置付け DELETE 文を使用する機能は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部です。

FROM キーワードは、ANSI/ISO SQL 標準では必須ですが、ソフトウェアでは省略可能です。
ansi_update_constraints オプションが Off に設定されている場合、更新可能なカーソルの範囲に、標準にはない拡張を含めることができます。

例

次の文は、データベースから カーソル cur_employee の現在のローを削除します。

```
DELETE
WHERE CURRENT OF cur_employee;
```

関連情報

UPDATE 文 [1391 ページ]
UPDATE (位置付け) 文 [ESQL] [SP] [1386 ページ]
INSERT 文 [1167 ページ]
PUT 文 [ESQL] [1244 ページ]

1.4.4.115 DELETE 文

データベースからローを削除します。

構文

一般的な使用

```
DELETE [ row-limitation ]
[ FROM ] [ owner.]table-or-view [ [ AS ] correlation-name ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ], ... ]
[ OPTION( query-hint, ... ) ]
```

Transact-SQL

```
DELETE [ row-limitation ]
[ FROM ] [ owner.]table-or-view [ [ AS ] correlation-name ]
[ FROM table-expression ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ], ... ]
[ OPTION( query-hint, ... ) ]
```

```
table-or-view : identifier
```

```
row-limitation :
FIRST
| TOP { ALL | limit-expression } [ START AT startat-expression ]
limit-expression : simple-expression
startat-expression : simple-expression
simple-expression :
integer
| variable
| ( simple-expression )
| ( simple-expression { + | - | * } simple-expression )
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| FORCE NO OPTIMIZATION
| option-name = option-value
```

```
table-expression : ジョインを含む完全なテーブル式
```

```
option-name :identifier
```

```
option-value :
hostvar（許可されたインジケータ）
| string
| identifier
| number
```

パラメータ

row-limitation 句

ローを制限する句を使用することによって、削除されるローを WHERE 句を満たすローのサブセットのみに制限できます。TOP 引数と START AT 引数には、ホスト変数、整数定数、または整数変数を使用した簡単な算術演算を指定できます。TOP 引数は 0 以上にします。START AT 引数は 0 より大きい値にします。これらの句を指定する場合は、ローの順序を意味のあるものにするために ORDER BY 句も指定する必要があります。

FROM 句

FROM 句は、ローが削除されるテーブルを示します。Transact-SQL 構文で、DELETE 文に第 2 FROM 句を指定すると、他のテーブルとのジョインに基づいて指定されたテーブルから削除されるローが決定されます。*table-expression* には、抽出テーブル、KEY ジョインおよび NATURAL ジョインなど、任意の複雑なテーブル式を含めることができます。

次の例は、Transact-SQL 構文が使用されたときに、相関名が照合される方法を示します。次の文では、

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_2 AS alias_2
WHERE ...
```

テーブル table_1 の相関名は、第 1 FROM 句にはありませんが、第 2 FROM 句には含まれています。ここでは、最初の句の table_1 が、2 番目の句では alias_1 と識別されます。この文には、table_1 のインスタンスが 1 つしかありません。これは、同じ文の中で相関名を使用する方法と使用しない方法の両方を使ってテーブルを識別する場合、テーブルの 2 つのインスタンスが考慮されるという一般規則の例外として許可されています。

ただし、次の例では、第 2 FROM 句に table_1 のインスタンスが 2 つあります。この文は構文エラーで失敗します。第 2 FROM 句の table_1 のインスタンスのうち、どれが第 1 FROM 句の table_1 の最初のインスタンスと一致するのかが明確ではないためです。

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

WHERE 句

DELETE 文は、WHERE 句の条件を満たすすべてのローを削除します。WHERE 句を指定しない場合、指定したテーブルからすべてのローは削除されます。第 2 FROM 句がある場合、WHERE 句はこの第 2 FROM 句の *table-expression* のローの条件を与えます。

ORDER BY 句

削除されるローのソート順を指定します。通常、ローを更新する順序は重要ではありません。ただし、FIRST 句または TOP 句と一緒に使用する場合は、順序が重要です。

ORDER BY 句ではカラムの順序数を使用できません。

ORDER BY リストの各項目には、昇順の場合 (デフォルト) は ASC、降順の場合は DESC のラベルを付けることができます。

OPTION 句

この句は、文を実行するためのヒントを指定するときに使用します。指定する設定は、現在の文にのみ適用され、ODBC 実行可能アプリケーションによる設定など、パブリックオプションまたはテンポラリオプションの設定よりも優先されます。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- FORCE NO OPTIMIZATION
- `option-name` = `option-value`
  クエリテキスト内の `OPTION( isolation_level = ... )` の指定を使用すると、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。
  クエリテキスト内の `OPTION( parameterization_level = ... )` の指定を使用すると、クエリのパラメータ化レベルよりも優先されます。

備考

DELETE 文を使用してデータを大量に削除する場合も、カラム統計は更新されます。

テーブルのすべてのローを削除する場合、より効率的な TRUNCATE TABLE 文の使用を検討してください。

ビューを定義しているクエリ指定が更新可能な場合は、ビューに対して DELETE 操作を実行できます。ビューは、ビューを定義している SELECT 文の FROM 句のテーブルが 1 つのみであり、かつ、DISTINCT 句、GROUP BY 句、WINDOW 句、集合関数、および集合演算子 (UNION または INTERSECT など) を含んでいない場合に、更新可能です。

権限

テーブルの所有者であるか、テーブルに対する SELECT 権限および DELETE 権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。ただし、次の機能は標準にありません。

- オプションの FROM キーワード。
- `row-limitation` 句と ORDER BY 句。
- OPTION 句。

Transact-SQL 構文は、Transact-SQL のベンダー拡張です。

例

FinancialData テーブルから、2000 年より前のデータすべてを削除します。

```
DELETE
FROM GROUPO.FinancialData
WHERE Year < 2000;
```

SalesOrderItems テーブルの最初から 10 の注文を削除します。各注文の出荷日は 2001-01-01 であり、地域は Central です。

```
DELETE TOP 10
FROM GROUPO.SalesOrderItems
FROM GROUPO.SalesOrders
WHERE SalesOrderItems.ID = SalesOrders.ID
  and ShipDate < '2001-01-01' and Region ='Central'
ORDER BY ShipDate ASC;
```

独立性レベル 3 で文を実行し、データベースから部署 600 を削除します。

```
DELETE FROM GROUPO.Departments
WHERE DepartmentID = 600
OPTION( isolation_level = 3 );
```

関連情報

TRUNCATE 文 [1366 ページ]
INSERT 文 [1167 ページ]
INPUT 文 [Interactive SQL] [1160 ページ]
FROM 句 [1112 ページ]

1.4.4.116 DESCRIBE 文 [ESQL]

データベースから取り出したデータを格納するために必要なホスト変数、またはデータベースにデータを渡すために必要なホスト変数に関する情報を取得します。

構文

```
DESCRIBE
[ USER TYPES ]
[ ALL | BIND VARIABLES FOR | INPUT | OUTPUT
| SELECT LIST FOR ]
[ LONG NAMES [ long-name-spec ] | WITH VARIABLE RESULT ]
[ FOR ] { statement-name | CURSOR cursor-name }
INTO sqlda-name
```

```
long-name-spec :
OWNER.TABLE.COLUMN
```

```
| TABLE.COLUMN
| COLUMN
```

```
statement-name :  identifier | hostvar
```

```
cursor-name :declared cursor
```

```
sqlda-name : identifier
```

パラメータ

USER TYPES 句

USER TYPES 句を持つ DESCRIBE 文は、カラムのドメインに関する情報を返します。通常、このような DESCRIBE は、以前の DESCRIBE が DT_HAS_USERTYPE_INFO のインジケータを返したときに行われます。

返された情報は、sqlname フィールドがカラム名の代わりにドメインの名前を保持すること以外は、USER TYPES キーワードのない DESCRIBE と同じです。

DESCRIBE が LONG NAMES 句を使用する場合は、sqldata フィールドがこの情報を保持します。

ALL 句

DESCRIBE ALL を使用すると、データベースサーバへの 1 回の要求で INPUT と OUTPUT を記述できます。これにはパフォーマンス上の利点があります。OUTPUT 情報が最初に SQLDA に格納され、次に INPUT 情報が格納されます。sqld フィールドには、INPUT と OUTPUT 変数の総数が入ります。インジケータ変数内の DT_DESCRIBE_INPUT ビットは INPUT 変数に対して設定され、OUTPUT 変数に対してクリアします。

BIND VARIABLES FOR 句

INPUT 句と同じです。

SELECT LIST FOR 句

OUTPUT 句と同じです。

INPUT 句

バインド変数は、データベースが文を実行するときにアプリケーションによって提供される値です。バインド変数は、文に対するパラメータと考えることができます。DESCRIBE INPUT は、バインド変数名を使って SQLDA に名前フィールドを格納します。DESCRIBE INPUT は、SQLDA の sqlda フィールドにバインド変数の数も格納します。

DESCRIBE は、SQLDA の中のインジケータ変数を使って情報を追加します。DT_PROCEDURE_IN と DT_PROCEDURE_OUT は、CALL 文を記述するときに、インジケータ変数の中に設定されるビットです。DT_PROCEDURE_IN は IN または INOUT パラメータを示し、DT_PROCEDURE_OUT は INOUT または OUT パラメータを示します。プロシージャ RESULT カラムは、両方のビットをクリアします。DESCRIBE OUTPUT の後、これらのビットは結果セットを持っている文 (OPEN、FETCH、RESUME、CLOSE を使用する必要がある) と持っていない文 (EXECUTE を使用する必要がある) を区別するのに使用できます。DESCRIBE INPUT は、バインド変数が CALL 文に対する引数であるときに、適切に DT_PROCEDURE_IN と DT_PROCEDURE_OUT を設定するだけです。CALL 文の引数である式の中のバインド変数は、ビットを設定しません。

OUTPUT 句

DESCRIBE OUTPUT 文は、SQLDA のそれぞれの SELECT リスト項目のデータ型と長さを埋めます。名前フィールドにも、SELECT リスト項目の名前が入ります。SELECT リスト項目に対してエイリアスを指定する場合、名前はそのエイリア

スになります。エイリアスを指定しない場合、名前を SELECT リスト項目から取り出します。項目が単純なカラム名である場合は、その名前が使用されます。そうでない場合は、式の部分文字列が使用されます。また、DESCRIBE は、SELECT リスト項目の数を SQLDA の sqld フィールドに格納します。

記述されている文が 2 つ以上の SELECT 文の UNION である場合、DESCRIBE OUTPUT に対して返されるカラム名は、最初の SELECT 文に対して返されるカラム名と同じです。

CALL 文を記述すると、DESCRIBE OUTPUT 文は、プロシージャ中の各 INOUT または OUT パラメータに対して、SQLDA の中にデータ型、長さ、名前を格納します。DESCRIBE OUTPUT 文は、SQLDA の sqld フィールドの中に INOUT または OUT パラメータの数も格納します。

結果セットを返す CALL 文を記述すると、DESCRIBE OUTPUT 文は、プロシージャ定義の各 RESULT カラムに対して、SQLDA の中にデータ型、長さ、名前を格納します。DESCRIBE OUTPUT 文は、SQLDA の sqld フィールドの中に結果カラムの数も格納します。

LONG NAMES 句

LONG NAMES 句は、文またはカーソルに対応するカラム名を取得するために用意されています。この句を使用しないと、取得するカラム名の長さは 29 文字に制限されますが、この句を使用すると、任意の長さのカラム名がサポートされます。

LONG NAMES を使用した場合、カーソルからフェッチを行う場合と同様に、長い名前は SQLDA の SQLDATA フィールドに格納されます。これ以外のフィールド (SQLLEN、SQLTYPE など) には入力されません。SQLDA は FETCH SQLDA のようにセットアップしてください。つまり、各カラムには、文字型のデータのエントリを 1 つ含めます。インジケータ変数がある場合、通常の方法でトランケーションが示されます。

長い名前のデフォルトの仕様は、TABLE.COLUMN です。

WITH VARIABLE RESULT 句

この句は、複数の結果セットと異なる数または型のカラムを持つプロシージャの記述に使用します。

WITH VARIABLE RESULT を使用する場合、DESCRIBE 文後にデータベースサーバは SQLCOUNT 値を次のいずれかに設定します。

0

結果セットは変更される場合があります。各 OPEN 文の後でプロシージャコールを記述し直してください。

1

結果セットは固定です。再度記述する必要はありません。

備考

DESCRIBE 文は指定した SQLDA を設定し、指定した文に対して OUTPUT (SELECT LIST FOR と同等) または INPUT (BIND VARIABLES FOR と同等) を記述します。

INPUT の場合、DESCRIBE BIND VARIABLES は SQLDA の中にデータ型を設定しません。アプリケーションがデータ型を設定する必要があります。ALL キーワードを使うと、1 つの SQLDA の中に INPUT と OUTPUT を記述できます。

文名を指定する場合は、同じ文名で PREPARE 文を使用して、文を事前に作成します。また、事前に SQLDA を割り付けます。

カーソル名を指定する場合、カーソルを事前に宣言し、開いておきます。デフォルトの動作は、OUTPUT です。SELECT 文と CALL 文だけが OUTPUT を持っています。他の文または動的カーソルではないカーソルの DESCRIBE OUTPUT は、SQLDA の sqld フィールドを 0 に設定して出力なしを示します。

Embedded SQL の場合、NCHAR、NVARCHAR、LONG NVARCHAR はそれぞれデフォルトで DT_FIXCHAR、DT_VARCHAR、DT_LONGVARCHAR と記述されます。db_change_nchar_charset 関数が呼び出された場合、これらのデータ型はそれぞれ DT_NFIXCHAR、DT_NVARCHAR、DT_LONGNVARCHAR と記述されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

DESCRIBE OUTPUT 文は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" です。DESCRIBE INPUT 文は、オプションの ANSI/ISO SQL 言語機能 B032、"Extended dynamic SQL" です。DESCRIBE 文のその他の多くの句は、標準にはありません。これらを以下に示します。

- USER TYPES、ALL、BIND VARIABLES FOR、LONG NAMES、WITH VARIABLE RESULT の各句。
- DESCRIBE は INTO 句を使用して SQLDA を識別します。ANSI/ISO SQL 標準では、USING キーワードが代わりに使用されます。
- ANSI/ISO SQL 標準では、CURSOR 句はキーワード STRUCTURE で終了します。STRUCTURE は、ソフトウェアではサポートされません。

例

次の例は、DESCRIBE 構造体の使用方法を示します。

```
sqlda = alloc_sqlda( 3 );
EXEC SQL DESCRIBE OUTPUT
  FOR employee_statement
  INTO sqlda;
if( sqlda->sqld  >  sqlda->sqln ) {
  actual_size = sqlda->sqld;
  free_sqlda( sqlda );
  sqlda = alloc_sqlda( actual_size );
  EXEC SQL DESCRIBE OUTPUT
    FOR employee_statement
    INTO sqlda;
}
```

関連情報

ALLOCATE DESCRIPTOR 文 [ESQL] [628 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
OPEN 文 [ESQL] [SP] [1218 ページ]
PREPARE 文 [ESQL] [1238 ページ]
LONG NVARCHAR データ型 [129 ページ]
NCHAR データ型 [130 ページ]
NVARCHAR データ型 [132 ページ]

1.4.4.117 DESCRIBE 文 [Interactive SQL]

指定されたデータベースオブジェクトに関する情報を返します。

構文

データベースオブジェクトを説明します

```
DESCRIBE [ [ INDEX FOR ] TABLE | PROCEDURE ] [ owner.]object-name
```

```
 object-name :
table
| view
| materialized view
| procedure
| function
```

現在の接続を説明します

```
DESCRIBE CONNECTION
```

パラメータ

INDEX FOR 句

指定した object-name のインデックスを表示することを指定します。

TABLE 句

記述する object-name がテーブルまたはビューであることを示します。

PROCEDURE 句

object-name がプロシージャまたはファンクションであることを示します。

備考

DESCRIBE TABLE を使用すると、指定したテーブルまたはビューのすべてのカラムが表示されます。DESCRIBE TABLE 文は 1 つのテーブルカラムにつき 1 つのローを返します。これには、次のものが含まれます。

カラム

カラム名。
タイプ

カラムに格納されているデータ型
NULL 入力可

NULL が許可されているかどうか (1=許可、0=不許可)
プライマリキー

カラムがプライマリキーにあるかどうか (1=ある、0=ない)

DESCRIBE INDEX FOR TABLE を使用すると、指定したテーブルのすべてのインデックスが表示されます。DESCRIBE TABLE 文は 1 つのインデックスにつき 1 つのローを返します。これには、次のものが含まれます。

インデックス名

インデックスの名前。
カラム

インデックス内のカラム。
ユニーク

インデックスがユニークかどうか (1=ユニーク、0=ユニークではない)。
タイプ

インデックスの型。使用可能な値は、次のとおりです。クラスタード、統計、ハッシュ、およびその他。

DESCRIBE PROCEDURE を使用すると、指定したプロシージャまたはファンクションが使用しているすべてのパラメータが表示されます。DESCRIBE PROCEDURE 文は、1 つのパラメータにつき 1 つのローを返します。これには、次のものが含まれます。

パラメータ

パラメータの名前。
タイプ

パラメータのユーザ型。
入力/出力

パラメータに渡すデータまたは返されるデータに関する情報。使用可能な値は、次のとおりです。

入力

パラメータはプロシージャに渡されますが、修正されません。
出力

プロシージャはパラメータの初期値を無視し、プロシージャが返るときに値を設定します。
入力/出力

パラメータはプロシージャに渡され、プロシージャはプロシージャが返るときにパラメータの値を設定します。
結果

パラメータは結果セットを返します。

戻り値

パラメータは宣言済みの戻り値を返します。

DESCRIBE PROCEDURE を使用した場合、返されるパラメータリストは SYSPROCPARM システムビューから取得されます。

TABLE または PROCEDURE を指定していない場合 (たとえば、DESCRIBE `object-name`)、Interactive SQL はオブジェクトをテーブルと仮定します。ただし、そのようなテーブルが存在しない場合、Interactive SQL はプロシージャまたはファンクションとしてオブジェクトを記述しようとします。

DESCRIBE 文を使用して Interactive SQL が接続されているデータベースまたはデータベースサーバに関する情報をリストします。次のプロパティが返されます。

データベース製品

Interactive SQL が接続されているデータベース製品の名前とバージョン番号 (SQL Anywhere17.0.4.1691 など)。

ホスト名

データベースサーバを実行しているコンピュータのネットワーク名。

ホストの TCP/IP アドレス

データベースサーバを実行しているコンピュータの IP アドレス。

ホストのオペレーティングシステム

データベースサーバを実行しているコンピュータで使用されるオペレーティングシステムの名前とバージョン番号。

サーバ名

データベースサーバの名前。

サーバの TCP/IP ポート

現在の接続でデータベースサーバによって使用されるポート番号。

データベース名

Interactive SQL が接続されているデータベースの名前。

データベースの文字セット

データベースの CHAR カラムに使用される文字セット。

接続文字列

データベースまたはデータベースサーバとの接続に使用された接続文字列。3 つのアスタリスクはパスワードに置き換えます。

現在の接続に該当しないプロパティは省略されます。たとえば、共有メモリを使用してデータベースサーバに接続する場合、TCP/IP ポートは省略されます。

権限

なし

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

Departments テーブル内のカラムを記述します。

```
DESCRIBE TABLE GROUPO.Departments;
```

この文の結果セットの例を示します。

Column	Type	Nullable	Primary key
DepartmentID	integer	0	1
DepartmentName	char(40)	0	0
DepartmentHeadID	integer	1	0

Customers テーブルのインデックスを表示します。

```
DESCRIBE INDEX FOR TABLE GROUPO.Customers;
```

この文の結果の例を示します。

Index Name	Columns	Unique	Type
IX_customer_name	Surname,GivenName	0	Clustered

関連情報

ALTER PROCEDURE 文 [669 ページ]

1.4.4.118 DETACH TRACING 文 (廃止)

診断トレース機能は廃止予定です。SQL Anywhere プロファイラを使用して、データベースの問題を診断します。DETACH TRACING セッションによって、診断トレースセッションは終了します。

構文

```
DETACH TRACING { WITH | WITHOUT } SAVE
```

パラメータ

WITH SAVE 句

WITH SAVE を指定すると、保存されていない診断データが診断テーブルに保存されます。

WITHOUT SAVE 句

保存されていないトレーシングデータを保存しない場合は、WITHOUT SAVE を指定します。

備考

プロファイルされているデータベースからこの文を実行すると、診断テーブルに対する診断情報の送信が停止されます。WITHOUT SAVE 句を指定しても、sa_save_trace_data システムプロシージャを使用して後でデータを保存できます (トレーシングデータベースが実行中で、他のトレースセッションが開始されていない場合)。

現在のトレーシングレベルを参照するには、sa_diagnostic_tracing_level テーブルを参照します。

注記

トレーシング情報は、データベースのアンロードまたは再ロード操作の一環としてアンロードされません。トレーシング情報を別のデータベースに転送する場合は、sa_diagnostic_* テーブルの内容をコピーして、手動で転送してください。ただし、この方法はお奨めしません。

権限

MANAGE PROFILING システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

ATTACH TRACING 文 (廃止) [737 ページ]
REFRESH TRACING LEVEL 文 (廃止予定) [1258 ページ]
sa_diagnostic_tracing_level テーブル (廃止予定) [1426 ページ]
sa_save_trace_data システムプロシージャ [1601 ページ]

1.4.4.119 DISCONNECT 文 [ESQL] [Interactive SQL]

データベースへの接続を切断します。

構文

```
DISCONNECT [ connection-name | CURRENT | ALL ]
```

```
connection-name :
identifier
| string
| hostvar
```

備考

DISCONNECT 文はデータベースサーバとの接続を切断し、データベースサーバが使用するすべてのリソースを解放します。切断しようとする接続が CONNECT 文上で指定されている場合、その名前を使って接続を指定できます。ALL を指定すると、すべてのデータベース環境へのアプリケーションの接続がすべて切断されます。CURRENT はデフォルトであり、現在の接続を切断します。

commit_on_exit オプションが On に設定されている場合は、データベース接続を閉じる前に Interactive SQL が自動的に COMMIT を実行します。このオプションが Off に設定されている場合は、Interactive SQL は暗黙の ROLLBACK を実行します。デフォルトでは、commit_on_exit オプションは On に設定されています。

この文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

DISCONNECT は、オプションの ANSI/ISO SQL 言語機能 F771 を構成します。パラメータを使用せずに DISCONNECT を指定する機能、および commit_on_exit option は、標準にはありません。

例

次の文は、Embedded SQL 内の DISCONNECT の使用法を示します。

```
EXEC SQL DISCONNECT :conn_name
```

次の文は、DISCONNECT を使用して、Interactive SQL との接続をすべて切断する方法を示します。

```
DISCONNECT ALL;
```

関連情報

DROP CONNECTION 文 [1030 ページ]
CONNECT 文 [ESQL] [Interactive SQL] [774 ページ]
SET CONNECTION 文 [Interactive SQL] [ESQL] [1302 ページ]

1.4.4.120 DROP CERTIFICATE 文

データベースから証明書を削除します。

構文

```
DROP CERTIFICATE [ IF EXISTS ] certificate-name
```

備考

DROP CERTIFICATE は、ISYSCERTIFICATE システムテーブルから証明書を削除します。

IF EXISTS 句は、存在しない証明書を DROP CERTIFICATE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

MANAGE CERTIFICATES システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

```
DROP CERTIFICATE mycert;
```

関連情報

CREATE CERTIFICATE 文 [779 ページ]

1.4.4.121 DROP CONNECTION 文

データベースへのユーザの接続を削除します。

構文

```
DROP CONNECTION connection-id
```

備考

DROP CONNECTION 文は、データベースへの接続を削除してデータベースからユーザを切断します。

`connection-id` パラメータは整数の定数です。sa_conn_info システムプロシージャを使用して、`connection-id` を取得できます。

この文は、プロシージャ、トリガ、イベント、またはバッチではサポートされていません。

権限

DROP CONNECTION システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のプロシージャは、接続番号で識別される接続を削除します。プロシージャ内から DROP CONNECTION 文を実行するときには、次の例に示すように、EXECUTE IMMEDIATE 文を使用して実行してください。

```
CREATE PROCEDURE drop_connection_by_id( IN conn_number INTEGER )
 BEGIN
   EXECUTE IMMEDIATE 'DROP CONNECTION ' || conn_number;
 END;
```

次の文は、ID 番号 4 の接続を切断します。

```
DROP CONNECTION 4;
```

関連情報

CONNECT 文 [ESQL] [Interactive SQL] [774 ページ]
sa_conn_info システムプロシージャ [1464 ページ]

1.4.4.122 DROP DATABASE 文

データベースに関連付けられているすべてのデータベースファイルを削除します。

構文

```
DROP DATABASE database-name [ KEY key ]
```

備考

DROP DATABASE 文は、関連するすべてのデータベースファイルをディスクから物理的に削除します。データベースファイルが存在しない場合、または起動に適した状態にない場合は、エラーが生成されます。

DROP DATABASE は、ストアドプロシージャ、トリガ、イベント、またはバッチに使用できません。

DROP DATABASE 文を使用する場合、消去するデータベースは実行中にしないでください。削除中のデータベースには接続できません。別のデータベースに接続されている必要があります。たとえば、ユーティリティデータベースに接続します。

強力に暗号化されたデータベースを削除する場合は、キーを指定してください。このキーには文字列 (定数リテラル) または変数参照を使用できます。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

ディスクからデータベースファイルを削除するだけでなく、関連のトランザクションログファイルやトランザクションログミラーファイルも削除されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文を実行してデータベース `temp.db` を削除します。

```
DROP DATABASE 'c:¥temp¥temp.db';
```

次の文を実行して、定数リテラル mykey に基づくデータベースを削除します。

```
DROP DATABASE 'temp.db' KEY 'mykey';
```

次の文を実行して、変数参照 mykeyvar に基づくデータベースを削除します。

```
DROP DATABASE 'temp.db' KEY mykeyvar;
```

関連情報

CREATE DATABASE 文 [781 ページ]

1.4.4.123 DROP DATATYPE 文

データベースからデータ型を削除します。

構文

```
DROP DATATYPE datatype-name
```

備考

DROP DATATYPE よりも DROP DOMAIN を使用することをお奨めします。これは、DROP DOMAIN が ANSI/ISO SQL 標準で使用されている構文であるためです。システム定義のデータ型 (MONEY、UNIQUEIDENTIFIERSTR など) をデータベースから削除することはできません。

権限

データ型の所有者であるか、DROP DATATYPE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

ドメインのサポートは、オプションの ANSI/ISO SQL 言語機能 F251 です。DROP DATATYPE 文は、標準にはありません。

例

次の例は、PhoneNum というデータ型を作成した後に削除します。

```
CREATE DATATYPE PhoneNum CHAR(12) NULL;
DROP DATATYPE PhoneNum;
```

関連情報

DROP DOMAIN 文 [1035 ページ]
CREATE DOMAIN 文 [794 ページ]
ALTER DOMAIN 文 [641 ページ]

1.4.4.124 DROP DBSPACE 文

データベースから DB 領域を削除します。

構文

```
DROP DBSPACE dbspace-name
```

備考

DB 領域を削除するには、その前に DB 領域に含まれるすべてのテーブルを削除する必要があります。DROP DBSPACE 文を使用して、SYSTEM、TEMPORARY、TEMP、TRANSLOG、TRANSLOGMIRROR の事前定義の DB 領域を削除することはできません。

DROP DBSPACE 文は、他の接続で現在使用中のオブジェクトに影響を及ぼす場合は実行できません。

この文を実行するには、データベースに排他的に接続する必要があります。

権限

MANAGE ANY DBSPACE システム権限が必要です。

関連する動作

オートコミット。暗黙的なチェックポイントが生じます。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例は、データベースから架空の DB 領域 MyDBSpace を削除します。

```
DROP DBSPACE MyDBSpace;
```

関連情報

CREATE DBSPACE 文 [789 ページ]
ALTER DBSPACE 文 [639 ページ]

1.4.4.125 DROP DOMAIN 文

データベースからドメイン (データ型) を削除します。

構文

```
DROP DOMAIN domain-name
```

備考

データ型をテーブルカラム、プロシージャの引数、または関数の引数に使用する場合、DROP DOMAIN は許可されません。データ型を削除するには、ドメインを使用して定義されているすべてのカラムのデータ型を変更します。DROP DATATYPE よ

りも DROP DOMAIN を使用することをお奨めします。これは、DROP DOMAIN が ANSI/ISO SQL 標準で使用されている構文であるためです。システム定義のデータ型 (MONEY、UNIQUEIDENTIFIERSTR など) をデータベースから削除することはできません。

権限

ドメインの所有者であるか、DROP DATATYPE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

ドメインのサポートは、オプションの SQL 言語機能 F251 です。

例

次の例は、CustPhoneNumber というドメインを作成した後に削除します。

```
CREATE DOMAIN CustPhoneNumber CHAR(12) NULL;
DROP DOMAIN CustPhoneNumber;
```

関連情報

CREATE DOMAIN 文 [794 ページ]
ALTER DOMAIN 文 [641 ページ]

1.4.4.126 DROP EVENT 文

データベースからイベントを削除します。

構文

```
DROP EVENT [ IF EXISTS ] event-name
```

備考

IF EXISTS 句は、存在しないイベントを DROP EVENT 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

イベントは所有されないため、所有者は指定しません。(所有者を指定しても無視されます。)

権限

MANAGE ANY EVENT または DROP ANY OBJECT のどちらかのシステム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例は、データベースから架空のサンプル DB MyEvent を削除します。

```
DROP EVENT MyEvent;
```

関連情報

CREATE EVENT 文 [805 ページ]
ALTER EVENT 文 [643 ページ]
TRIGGER EVENT 文 [1365 ページ]

1.4.4.127 DROP EXTERNLOGIN 文

データベースから外部ログインを削除します。

構文

外部ログインの削除

```
DROP EXTERNLOGIN login-name TO remote-server
```

外部ログインの削除 (構文の変数を含む)

```
DROP EXTERNLOGIN USER string | variable SERVER string | variable
```

パラメータ

DROP 句

ローカルユーザのログイン ID を指定します。

SERVER 句

リモートサーバ名を指定します。このサーバのローカルユーザの代替ログイン名とパスワードが、削除される外部ログインです。

備考

DROP EXTERNLOGIN はデータベースから外部ログインを削除します。

注記

変数名を受け付ける必須パラメータの場合、次のいずれかの条件にあてはまる場合は、データベースサーバからエラーが返されます。

- 変数が存在しない
- 変数の内容が NULL
- 変数がパラメータで許可されている長さを超えている
- 変数のデータ型がパラメータで要求されているものと一致していない

権限

MANAGE ANY USER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、架空のリモートサーバ sybase1 に対する DBA の外部ログインを削除します。

```
DROP EXTERNLOGIN DBA TO sybase1;
```

関連情報

CREATE EXTERNLOGIN 文 [816 ページ]

1.4.4.128 DROP FUNCTION 文

データベースから関数を削除します。

構文

```
DROP FUNCTION [ IF EXISTS ] [ owner.]function-name
```

備考

IF EXISTS 句は、DROP FUNCTION 文が存在しない関数を削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

関数の所有者であるか、DROP ANY PROCEDURE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

コア機能。IF EXISTS 句は標準にありません。

例

この例は、データベースから架空の関数 MyFunction を削除します。

```
DROP FUNCTION MyFunction;
```

関連情報

CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
ALTER FUNCTION 文 [649 ページ]

1.4.4.129 DROP INDEX 文

データベースからインデックスを削除します。

構文

```
DROP INDEX [ IF EXISTS ] { [ [ owner.]table-name.]index-name |
[ [ owner.]materialized-view-name. ]index-name }
```

備考

IF EXISTS 句は、存在しないインデックスを DROP INDEX 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

IF EXISTS 句を指定し、指定のテーブルが見つからない場合、エラーが返されます。

DROP INDEX 文は、他の接続で現在使用中のオブジェクトに影響を及ぼす場合は実行できません。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、DROP INDEX 文は実行できません。

権限

テーブルのインデックスを削除するには、テーブルの所有者であるか、次のいずれかの権限を持っている必要があります。

- そのテーブルに対する REFERENCES 権限
- DROP ANY INDEX システム権限
- DROP ANY OBJECT システム権限

マテリアライズドビューのインデックスを削除するには、マテリアライズドビューの所有者であるか、次のいずれかの権限を持っている必要があります。

- DROP ANY INDEX システム権限
- DROP ANY OBJECT システム権限

関連する動作

オートコミット。DROP INDEX 文は、現在の接続のすべてのカーソルを閉じます。

ローカルテンポラリテーブルのインデックスを削除するために DROP INDEX 文を使用すると、インデックスが見つからないことを示すエラーが返されます。ローカルテンポラリテーブルの削除には、DROP TABLE 文を使用してください。ローカルテンポラリテーブル上のインデックスは、ローカルテンポラリテーブルがスコープ外になったときに自動的に削除されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例は、データベースから架空のインデックス MyIndex を削除します。

```
DROP INDEX MyIndex;
```

関連情報

CREATE INDEX 文 [842 ページ]
ALTER INDEX 文 [652 ページ]

1.4.4.130 DROP LDAP SERVER 文

LDAP サーバ設定オブジェクトを削除します。

構文

```
DROP LDAP SERVER ldapua-server-name
[ WITH DROP ALL REFERENCES ]
[ WITH SUSPEND ]
```

パラメータ

WITH DROP ALL REFERENCES 句

ログインポリシーから参照される LDAP サーバ設定オブジェクトを削除するには、DROP ALL REFERENCES 句を指定します。

WITH SUSPEND 句

READY または ACTIVE 状態の LDAP サーバ設定オブジェクトを削除するには、WITH SUSPEND 句を指定します。

備考

この文は、LDAP サーバ設定オブジェクトを READY または ACTIVE 状態ではないかチェックした後に、SYSLDAPSERVER から削除します。この文は、LDAP サーバがアクティブ (使用中) ではないことを確認するため、状態が READY または ACTIVE の場合は失敗します。このチェックを無効にするには、WITH SUSPEND 句を使用します。

デフォルトでは、ログインポリシーでの LDAP サーバ設定オブジェクトへの参照によっても、この文は失敗します。ログインポリシーで参照される LDAP サーバ設定オブジェクトを削除するには、DROP ALL REFERENCES 句を追加します。DROP ALL REFERENCES 句を追加しても、ログインポリシーから参照は削除されません。この句は参照が存在する場合に設定オブジェクトを削除できるようにします。その後、ログインポリシーから LDAP サーバ設定オブジェクトへの参照を削除する必要があります。

権限

MANAGE ANY LDAP SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ログインポリシーで参照される apps_primary という名前の LDAP サーバ設定オブジェクトを削除します。

```
DROP LDAP SERVER apps_primary WITH DROP ALL REFERENCES WITH SUSPEND;
```

関連情報

CREATE LDAP SERVER 文 [847 ページ]
ALTER LDAP SERVER 文 [654 ページ]
VALIDATE LDAP SERVER 文 [1398 ページ]
ALTER LOGIN POLICY 文 [657 ページ]

1.4.4.131 DROP LOGIN POLICY 文

ログインポリシーを削除します。

構文

```
DROP LOGIN POLICY policy-name
```

パラメータ

policy-name

ログインポリシーの名前。

備考

ユーザに割り当てられたログインポリシーを削除すると、文が失敗します。ルートログインポリシーは削除できません。ALTER USER 文を使用してユーザのポリシー割り当てを変更します。

権限

MANAGE ANY LOGIN POLICY システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ログインポリシー Test11 を作成し、その後、削除します。

```
CREATE LOGIN POLICY Test11;
DROP LOGIN POLICY Test11;
```

関連情報

ALTER LOGIN POLICY 文 [657 ページ]
ALTER USER 文 [731 ページ]
COMMENT 文 [769 ページ]
CREATE LOGIN POLICY 文 [852 ページ]
CREATE USER 文 [990 ページ]
DROP USER 文 [1084 ページ]

1.4.4.132 DROP MATERIALIZED VIEW 文

データベースからマテリアライズドビューを削除します。

構文

```
DROP MATERIALIZED VIEW [ IF EXISTS ] [ owner.]materialized-view-name
```

備考

テーブル内のすべてのデータは、削除プロセスの一部として自動的に削除されます。マテリアライズドビューのすべてのインデックスとキーも削除されます。

IF EXISTS 句は、存在しないマテリアライズドビューを DROP MATERIALIZED VIEW 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

他の接続で現在使用中のオブジェクトに対して DROP MATERIALIZED VIEW 文を実行することはできません。

DROP MATERIALIZED VIEW 文を実行すると、すべての通常の従属ビューのステータスが INVALID に変わります。マテリアライズドビューを削除する前にビューの依存関係を判断するには、sa_dependent_views システムプロシージャを使用します。

権限

マテリアライズドビューの所有者であるか、DROP ANY MATERIALIZED VIEW または DROP ANY OBJECT システム権限を持っていることが必要です。

関連する動作

オートコミット。マテリアライズドビューにデータが移植されている場合、DROP MATERIALIZED VIEW は自動チェックポイントをトリガします。現在接続しているすべてのカーソルを閉じます。

ビューを削除すると、すべてのプロシージャとトリガがメモリからアンロードされます。これにより、削除されたビューを参照するプロシージャやトリガは、そのビューが存在しないことを反映します。ビューの削除や作成を頻繁に行うと、プロシージャとトリガのアンロードやロードによってパフォーマンスが低下することがあります。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、データベースから架空のマテリアライズドビュー MyMaterializedView を削除します。

```
DROP MATERIALIZED VIEW MyMaterializedView;
```

関連情報

CREATE MATERIALIZED VIEW 文 [854 ページ]
ALTER MATERIALIZED VIEW 文 [659 ページ]
REFRESH MATERIALIZED VIEW 文 [1252 ページ]
sa_dependent_views システムプロシージャ [1487 ページ]

1.4.4.133 DROP MESSAGE 文

データベースからメッセージを削除します。

構文

```
DROP MESSAGE msgnum
```

備考

なし。

権限

所有者であるか、DROP MESSAGE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

DROP MESSAGE には、Adaptive Server Enterprise の sp_dropmessage() システムプロシージャによって提供される機能が用意されています。

例

次の例は、新しいメッセージを作成した後に削除します。この例を実行するには、CREATE MESSAGE システム権限も必要です。

```
CREATE MESSAGE 20000 AS 'End of line reached';
DROP MESSAGE 20000;
```

関連情報

PRINT 文 [T-SQL] [1243 ページ]
CREATE MESSAGE 文 [T-SQL] [856 ページ]
SYSUSERMESSAGE システムビュー [1858 ページ]

1.4.4.134 DROP MIRROR SERVER 文

ミラーサーバを削除します。

構文

```
DROP MIRROR SERVER mirror-server-name
```

備考

指定したミラーサーバ定義をデータベースから削除します。

ミラーデータベースが停止します。サーバ上で実行されているデータベースがミラーデータベースのみの場合は、サーバも停止します。

読み込み専用のスケールアウトとデータベースミラーリングには、それぞれ別途ライセンスが必要です。

権限

MANAGE ANY MIRROR SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例は、scaleout_primary2 という名前のミラーサーバを作成した後に削除します。

```
CREATE MIRROR SERVER "scaleout_primary2"
   AS PRIMARY
   connection_string = 'server=scaleout_primary1;host=winxp-2:6871,winxp-3:6872';
DROP MIRROR SERVER "scaleout_primary2";
```

関連情報

CREATE MIRROR SERVER 文 [858 ページ]
ALTER MIRROR SERVER 文 [662 ページ]
COMMENT 文 [769 ページ]

1.4.4.135 DROP MUTEX 文

指定されたミューテックスを削除します。

構文

```
DROP MUTEX [ IF EXISTS ] [ owner.]mutex-name
```

パラメータ

owner

ミューテックスの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

mutex-name

ミューテックスの名前。mutex-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

IF EXISTS 句

この句は、ミューテックスが存在する場合にのみミューテックスを削除するときに使用します。ミューテックスが存在しない場合にこの句を指定すると、何も実行されずエラーも返されません。

備考

ミューテックスが他の接続によってロックされている場合でも削除操作はブロックせずに行われますが、ミューテックスはリリースされるまでネームスペースに保持されます。ミューテックス上で待機している接続は、オブジェクトが削除されたことを示すエラーをただちに受信します。

権限

DROP ANY MUTEX SEMAPHORE または DROP ANY OBJECT システム権限を持っているか、永続ミューテックスの所有者である必要があります。一時ミューテックスの場合、ミューテックスを作成した接続である必要があります。

関連する動作

オートコミット (永続ミューテックスの場合のみ)。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、protect_my_cr_section ミューテックスを削除します。

```
DROP MUTEX protect_my_cr_section;
```

関連情報

CREATE MUTEX 文 [863 ページ]
LOCK MUTEX 文 [1199 ページ]
RELEASE MUTEX 文 [1259 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.136 DROP ODATA PRODUCER 文

OData プロデューサを削除します。

構文

```
DROP ODATA PRODUCER [ IF EXISTS ] name
```

備考

OData プロデューサの定義とそのすべてのオプションをシステムテーブルから削除します。この文は、OData プロデューサの設定データのみを削除する場合に使用します。OData プロデューサを一時的に無効にする場合は、次の文を実行します。

```
ALTER ODATA PRODUCER name NOT ENABLED;
```

IF EXISTS 句は、存在しない OData プロデューサを DROP ODATA PRODUCER 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

MANAGE ODATA システム権限が必要です。

関連する動作

OData サーバが実行されている場合、指定したプロデューサがシャットダウンされます。

例

prod という名前の OData プロデューサを削除するには、次の文を実行します。

```
DROP ODATA PRODUCER prod;
```

関連情報

ALTER ODATA PRODUCER 文 [666 ページ]
CREATE ODATA PRODUCER 文 [865 ページ]
COMMENT 文 [769 ページ]
SYSODATAPRODUCER システムビュー [1822 ページ]

1.4.4.137 DROP PROCEDURE 文

データベースからプロシージャを削除します。

構文

```
DROP PROCEDURE [ IF EXISTS ] [ owner.]procedure-name
```

備考

IF EXISTS 句は、存在しないプロシージャを DROP PROCEDURE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

プロシージャの所有者であるか、DROP ANY PROCEDURE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

コア機能。IF EXISTS 句は標準にありません。

例

この例は、NewDepartment というプロシージャを作成した後に削除します。この例を実行するには、CREATE PROCEDURE 権限も必要です。

```
CREATE PROCEDURE NewDepartment(
   IN id INT,
   IN name CHAR(35),
   IN head_id INT )
BEGIN
   INSERT
   INTO GROUPO.Departments ( DepartmentID,
       DepartmentName, DepartmentHeadID )
   VALUES ( id, name, head_id );
END;
DROP PROCEDURE NewDepartment;
```

関連情報

CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
ALTER PROCEDURE 文 [669 ページ]

1.4.4.138 DROP PUBLICATION 文 [Mobile Link] [SQL Remote]

パブリケーションを削除します。

構文

```
DROP PUBLICATION [ IF EXISTS ] [ owner.]publication-name
```

```
owner, publication-name : identifier
```

備考

この文は、Mobile Link と SQL Remote にのみ適用されます。

Mobile Link では、パブリケーションが SQL Anywhere リモートデータベース内の同期データを識別します。SQL Remote では、統合データベース内とリモートデータベース内の両方のレプリケートされたデータがパブリケーションによって識別されます。

IF EXISTS 句は、存在しないパブリケーションを DROP PUBLICATION 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

DROP PUBLICATION では、パブリケーションで参照されるすべてのテーブルへの排他アクセスが必要です。

権限

パブリケーションの所有者であるか、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。パブリケーションに対するサブスクリプションがすべて削除されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、pub_contact パブリケーションを削除します。

```
DROP PUBLICATION pub_contact;
```

関連情報

ALTER PUBLICATION 文 [Mobile Link] [SQL Remote] [672 ページ]
CREATE PUBLICATION 文 [Mobile Link] [SQL Remote] [899 ページ]

1.4.4.139 DROP REMOTE MESSAGE TYPE 文 [SQL Remote]

データベースからメッセージタイプの定義を削除します。

構文

```
DROP REMOTE MESSAGE TYPE message-system
```

```
message-system :
FILE
| FTP
```

| *SMTP*

備考

この文は、データベースからメッセージタイプを削除します。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、データベースから FILE メッセージタイプを削除します。

```
DROP REMOTE MESSAGE TYPE FILE;
```

関連情報

CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote] [903 ページ]

1.4.4.140 DROP REMOTE CONNECTION 文

リモートサーバへのリモーデータアクセスの接続を切断します。

構文

```
DROP REMOTE CONNECTION TO server-name
CLOSE { CURRENT | ALL | connection-id }
[ FOR { EFFECTIVE USER | LOGIN USER | USER user-name } ]
```

パラメータ

server-name

CREATE SERVER 文で指定されたリモートデータアクセスサーバ。

CLOSE 句

CLOSE CURRENT は、現在のローカル接続について、リモート接続を切断します。

CLOSE ALL は、すべてのローカル接続について、リモート接続を切断します。

CLOSE `connection-id` は、指定された ID のローカル接続について、リモート接続を切断します。

FOR 句

FOR EFFECTIVE USER は、現在の有効なユーザの externlogin クレデンシャルを使用して作成されたリモート接続を切断します。

FOR LOGIN USER は、現在のログインユーザの externlogin クレデンシャルを使用して作成されたリモート接続を切断します。

FOR USER `user-name` は、`user-name` の externlogin クレデンシャルを使用して作成されたリモート接続を切断します。

FOR 句を省略した場合は、すべてのユーザのリモート接続が切断されます。

備考

DROP REMOTE CONNECTION 文を使用すると、リモートサーバへの接続を明示的に閉じることができます。この句は、リモート接続が非アクティブまたは不要になった場合に役立ちます。

権限

関連する動作

なし

例

有効なユーザを使用した myServer サーバへのすべてのリモート接続を、それらが現在の接続であるかどうかにかかわらず切断します。

```
DROP REMOTE CONNECTION TO myServer CLOSE ALL FOR EFFECTIVE USER;
```

現在のローカル接続について、user2 を使用した myServer サーバへのリモート接続を切断します。

```
DROP REMOTE CONNECTION TO myServer CLOSE CURRENT FOR USER user2;
```

user2 を使用した myServer サーバへのすべてのリモート接続を、それらが現在の接続であるかどうかにかかわらず切断します。

```
DROP REMOTE CONNECTION TO myServer CLOSE ALL FOR USER user2;
```

現在のローカル接続について、ログインユーザを使用した myServer サーバへのリモート接続を切断します。

```
DROP REMOTE CONNECTION TO myServer CLOSE CURRENT FOR LOGIN USER;
```

ID が connectionid で有効なユーザを使用した myServer サーバへのリモート接続を切断します。

```
DROP REMOTE CONNECTION TO myServer CLOSE connectionId FOR EFFECTIVE USER;
```

関連情報

CREATE SERVER 文 [914 ページ]
ALTER SERVER 文 [679 ページ]

1.4.4.141 DROP ROLE 文

データベースからロールを削除するか、ユーザ拡張ロールを変換して通常のユーザに戻します。

構文

```
DROP ROLE [ FROM USER ] role-name
[ WITH { REVOKE | DROP OBJECTS } ]
```

パラメータ

role-name

削除または変換するロールの名前を指定します。

FROM USER 句

ユーザ拡張ロールを変換して通常のユーザに戻すには、この句を指定します。ユーザはすべてのログイン権限、システム権限、保持しているロールをすべて維持します。

WITH REVOKE 句

WITH REVOKE は、`role-name` を付与された他のユーザが存在する場合に指定します。

WITH DROP OBJECTS 句

`role-name` によって所有されているオブジェクトを削除するには、WITH DROP OBJECTS を指定します。オブジェクトが使用中などで削除できない場合、この文はエラーを返します。`role-name` がユーザ拡張ロールの場合、この句は指定できません。

備考

ユーザ定義ロールはデータベースから削除することができ、ユーザ拡張ロールは変換して通常のユーザに戻すことができます。ただし、依存するすべてのロールが、min_role_admin データベースオプションで設定された、アクティブなパスワードを持つ管理者ユーザの最小数の要件を満たす必要があります。

ユーザ拡張ロールを変換して通常のユーザに戻す場合、オブジェクトの所有権は、変換されて通常のユーザに戻されたユーザに残ります。

ユーザ拡張ロールを変換して通常のユーザに戻す場合、`role-name` に対して付与されたすべての権限は、それらが変換された後にユーザに残ります。

ユーザ拡張ロールを変換して通常のユーザに戻す際に、その他のすべてのロールやユーザにそのユーザ拡張ロールが付与されていた場合は、WITH REVOKE 句を指定する必要があります。そうしないと、この文はエラーメッセージを返して失敗します。

削除操作の影響を受けるオブジェクトが使用中の場合、この文はエラーメッセージを返して失敗します。

権限

削除しようとしているロールに対する管理権限が必要です。

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Joe という名前のユーザ拡張ロールを通常のユーザに戻します。ユーザ拡張ロールが所有するオブジェクトは、通常のユーザ、Joe の所有となります。Joe を付与されたユーザまたはロールは、このロールに関連付けられた、基本となる権限を保持します。

```
DROP ROLE FROM USER Joe;
```

次の文は、データベースから Jack という名前のユーザ拡張ロールを削除します。ロール Jack がオブジェクトを所有していた場合、そのオブジェクトの所有権はユーザ Jack に戻されます。Jack を付与されたユーザまたはロールは、ロール Jack に関連付けられた基本となる権限を保持します。

```
DROP ROLE Jack;
```

次の文は、Sam という名前のユーザ拡張ロールを通常のユーザに戻します。Sam を付与されたユーザおよびロールは、Sam に関するすべての基本権限を失います。

```
DROP ROLE FROM USER Sam WITH REVOKE;
```

次の文は、Sales1 という名前のロールを削除します。Sales1 を付与されていたユーザまたはロールは、Sales1 に関連する基本権限を保持します。

```
DROP ROLE Sales1;
```

次の文は、Sales2 という名前のロールを削除します。Sales2 を付与されていたユーザまたはロールは、Sales2 に関連するすべての基本権限を失います。

```
DROP ROLE Sales2 WITH REVOKE;
```

次の文は、Marketing1 という名前のユーザ拡張ロールを、Marketing1 という名前の通常のユーザに変換し、所有していたすべてのオブジェクトを削除します。

```
DROP ROLE FROM USER Marketing1 WITH DROP OBJECTS;
```

次の文は、drops a role named Marketing2 という名前のロールを削除し、このロールが所有していたオブジェクトを削除して、このロールを付与されていたユーザの、ロールの基本となるシステム権限を取り消します。

```
DROP ROLE Marketing2 WITH REVOKE WITH DROP OBJECTS;
```

関連情報

ALTER ROLE 文 [675 ページ]
CREATE ROLE 文 [905 ページ]

1.4.4.142 DROP SEMAPHORE 文

セマフォを削除します。

構文

```
DROP SEMAPHORE [ IF EXISTS ] [ owner.]semaphore-name
```

パラメータ

owner

セマフォの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

semaphore-name

セマフォの名前。semaphore-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

IF EXISTS 句

この句は、セマフォが存在する場合にのみセマフォを削除するときに使用します。セマフォが存在しない場合にこの句を指定すると、何も実行されずエラーも返されません。

備考

エラーは、セマフォの減分を待機している任意の接続に返されます。

権限

DROP ANY MUTEX SEMAPHORE または DROP ANY OBJECT システム権限を持っているか、セマフォの所有者である必要があります。一時セマフォの場合、ミューテックスを作成した接続である必要があります。

関連する動作

オートコミット (永続セマフォの場合のみ)。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、license_counter というセマフォを削除します。

```
DROP SEMAPHORE license_counter;
```

関連情報

CREATE SEMAPHORE 文 [909 ページ]
NOTIFY SEMAPHORE 文 [1215 ページ]
WAITFOR SEMAPHORE 文 [1405 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.143 DROP SEQUENCE 文

シーケンスを削除します。

構文

```
DROP SEQUENCE [ owner.] sequence-name
```

備考

指定したシーケンスが見つからない場合は、エラーメッセージが返されます。シーケンスを削除すると、そのシーケンス名のすべての同義語がデータベースサーバによって自動的に削除されます。

権限

シーケンスの所有者であるか、DROP ANY SEQUENCE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

シーケンスは、ANSI/ISO SQL 言語機能 T176 の一部です。

例

次の例では、Test という名前のシーケンスを作成した後に削除します。

```
CREATE SEQUENCE Test
START WITH 4
INCREMENT BY 2
NO MAXVALUE
NO CYCLE
CACHE 15;
DROP SEQUENCE Test;
```

関連情報

ALTER SEQUENCE 文 [677 ページ]
CREATE SEQUENCE 文 [911 ページ]

1.4.4.144 DROP SERVER 文

カタログからリモートサーバを削除します。

構文

リモートサーバの削除

```
DROP [ REMOTE ] SERVER server-name
```

リモートサーバの削除 (SAP HANA 構文)

```
DROP REMOTE SOURCE remote-source-name
```

パラメータ

DROP SERVER、DROP REMOTE SERVER、および DROP REMOTE SOURCE 文は、セマンティック上同じです。DROP REMOTE SOURCE 構文は、SAP HANA との互換性のために用意されています。

備考

この文を成功させるには、リモートサーバに定義されたプロキシテーブルをすべて削除する必要があります。

権限

SERVER OPERATOR システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、ase_prod という名前の架空のサーバを削除します。

```
DROP SERVER ase_prod;
```

関連情報

ALTER SERVER 文 [679 ページ]
CREATE SERVER 文 [914 ページ]
DROP REMOTE CONNECTION 文 [1055 ページ]

1.4.4.145 DROP SERVICE 文

Web サービスを削除します。

構文

```
DROP SERVICE service-name
```

備考

この文は、ISYSWEBSERVICE システムテーブルに表示されている Web サービスを削除します。

権限

そのサービスの所有者であるか、または MANAGE ANY WEB SERVICE システム権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の SQL 文は、WebServiceTable という名前の架空の Web サービスを削除します。

```
DROP SERVICE WebServiceTable;
```

関連情報

CREATE SERVICE 文 [HTTP Web サービス] [921 ページ]
CREATE SERVICE 文 [SOAP Web サービス] [927 ページ]

ALTER SERVICE 文 [HTTP Web サービス] [683 ページ]
ALTER SERVICE 文 [SOAP Web サービス] [688 ページ]
SYSWEBSERVICE システムビュー [1862 ページ]

1.4.4.146 DROP SPATIAL REFERENCE SYSTEM 文

空間参照系を削除します。

構文

```
DROP SPATIAL REFERENCE SYSTEM [ IF EXISTS ] name
```

備考

IF EXISTS 句は、存在しない空間参照系を DROP SPATIAL REFERENCE SYSTEM 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

所有者であるか、MANAGE ANY SPATIAL OBJECT または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Test という名前の架空の空間参照系を削除します。

```
DROP SPATIAL REFERENCE SYSTEM Test;
```

関連情報

CREATE SPATIAL REFERENCE SYSTEM 文 [933 ページ]
ALTER SPATIAL REFERENCE SYSTEM 文 [694 ページ]

1.4.4.147 DROP SPATIAL UNIT OF MEASURE 文

空間測定単位を削除します。

構文

```
DROP SPATIAL UNIT OF MEASURE [ IF EXISTS ] identifier
```

備考

IF EXISTS 句は、存在しない空間測定単位を DROP SPATIAL UNIT OF MEASURE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

空間測定単位の所有者であるか、MANAGE ANY SPATIAL OBJECT または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Test という名前の架空の空間測定単位を削除します。

```
DROP SPATIAL UNIT OF MEASURE Test;
```

関連情報

CREATE SPATIAL UNIT OF MEASURE 文 [940 ページ]

1.4.4.148 DROP STATEMENT 文 [ESQL]

文のリソースを解放します。

構文

```
DROP STATEMENT [ owner.]statement-name
```

```
statement-name :
identifier
| hostvar
```

備考

DROP STATEMENT 文は、指定した文によって使われているリソースを解放します。これらのリソースは、PREPARE 文を実行して割り付けられます。通常、データベース接続が解放されるまで、リソースは解放されません。

文を削除するには、最初にその文を準備する必要があります。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。ANSI/ISO SQL 標準では、この機能はオプションの ANSI/ISO SQL 言語機能 B032、"Extended dynamic SQL" の一部である DEALLOCATE PREPARE 文 によって提供されます。

例

次は、DROP STATEMENT の使用例です。

```
EXEC SQL DROP STATEMENT S1;
EXEC SQL DROP STATEMENT :stmt;
```

関連情報

PREPARE 文 [ESQL] [1238 ページ]

1.4.4.149 DROP STATISTICS 文

指定したカラムのすべての統計情報を消去します。

構文

```
DROP STATISTICS [ ON ] [owner.]object-name [ ( column-list ) ]
```

```
object-name :
table-name
| materialized-view-name
| temp-table-name
```

備考

データベースサーバオプティマイザは、カラムの統計情報を基にして、各文の実行に最適な方式を判別します。データベースサーバは、それらの統計を自動的に収集、更新します。カラムの統計情報は、データベースのシステムテーブル SYSCOLSTAT に永久的に格納されます。ある文の処理中に収集されたカラムの統計情報は、以降の文の効率的な実行方法を見いだすときに使用できます。

場合によっては、カラムの統計情報が不正確になったり、関連統計情報が使用不能になったりすることがあります。このような状況がもっとも発生しやすいのは、大量のデータが追加、更新、または削除されてから実行されたクエリが少ない場合です。

DROP STATISTICS 文は、システムテーブル ISYSCOLSTAT から、指定されたカラムに関する内部統計データをすべて削除します。これは強力な処理で、オプティマイザは必要な統計情報にアクセスできなくなります。これらの統計情報がない場合、オプティマイザは非効率的なデータアクセスプランを生成し、データベースパフォーマンスの低下を招くことがあります。

DROP STATISTICS 文には、実行対象のテーブルに排他ロックをかける必要があります。テーブルを参照するその他すべての接続がコミットまたはロールバックされるか、テーブルを参照している開いたカーソルをすべて閉じるまで、文の実行は進行しません。

この文は、問題を追究するときや、元のデータとは大幅に異なるデータをデータベースに再ロードする場合のみ使用してください。

CREATE STATISTICS 文と DROP STATISTICS 文は、統計情報に関するコミットを必要としませんし、コミットしても意味はありません。DROP STATISTICS だけは関連する動作としてコミットが行われます。ただし、コミットしても統計情報の変更が保存されるわけではありません (統計ガバナーが保守を行います)。したがって、DROP STATISTICS の関連する動作としてオートコミットが行われるとは言っても、コミットされるのは統計情報の消去を除いたすべての変更です。

権限

テーブル所有者であるか、MANAGE ANY STATISTICS または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE STATISTICS 文 [942 ページ]
SYSCOLSTAT システムビュー [1796 ページ]

1.4.4.150 DROP SUBSCRIPTION 文 [SQL Remote]

パブリケーションからユーザのサブスクリプションを削除します。

構文

```
DROP SUBSCRIPTION TO publication-name [ ( subscription-value ) ]
 FOR subscriber-id, ...
```

```
subscription-value : string
```

```
subscriber-id : string
```

パラメータ

publication-name

このパブリケーション名でユーザのサブスクリプションが作成されます。パブリケーションの所有者を含めることもできます。

subscription-value

パブリケーションのサブスクリプション式と照合される文字列。この値は、削除するサブスクリプションにサブスクリプション値が含まれる場合にのみ必要です。

subscriber-id

パブリケーションに対するサブスクライバのユーザ ID。

備考

現在のデータベースのパブリケーションに対するユーザ ID の SQL Remote サブスクリプションを削除します。今後パブリケーションのデータが変更されても、ユーザ ID は更新を受け付けません。

SQL Remote では、パブリケーションとサブスクリプションは双方向の関係です。統合データベース上のパブリケーションに対するリモートユーザのサブスクリプションを削除する場合、リモートデータベースでも統合データベースのサブスクリプションを削除してください。これは、リモートデータベースに対する更新が統合データベースに送信されるのを防ぐためです。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、pub_contact publication に対するユーザ ID SamS のサブスクリプションを削除します。

```
DROP SUBSCRIPTION TO pub_contact
FOR Sam_Singer;
```

関連情報

CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]
STOP SUBSCRIPTION 文 [SQL Remote] [1353 ページ]
SYSSUBSCRIPTION システムビュー [1839 ページ]

1.4.4.151 DROP SYNCHRONIZATION PROFILE 文 [Mobile Link]

SQL Anywhere 同期プロファイルを削除します。

構文

```
DROP SYNCHRONIZATION PROFILE [ IF EXISTS ] name
```

パラメータ

name

削除する同期プロファイルの名前。

備考

同期プロファイルは、同期の制御に使用できる同期オプションの名前付きコレクションです。IF EXISTS 句は、存在しない同期プロファイルを DROP SYNCHRONIZATION PROFILE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link] [946 ページ]
ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link] [700 ページ]

1.4.4.152 DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]

リモートデータベース内の同期サブスクリプションを削除します。

構文

```
DROP SYNCHRONIZATION SUBSCRIPTION { subscription-name |
TO publication-name
[ FOR ml-username, ... ] }
```

パラメータ

subscription-name

削除するサブスクリプションの名前を指定します。

TO 句

パブリケーション名を指定します。

FOR 句

1 人以上のユーザを指定します。

この句を省略すると、パブリケーションに対するデフォルトの設定が削除されます。

備考

パブリケーションで参照されるすべてのテーブルへの排他アクセスが必要です。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、sales という名前のサブスクリプションを削除します。

```
DROP SYNCHRONIZATION SUBSCRIPTION sales;
```

次の例では、Mobile Link ユーザ SSinger とパブリケーション sales_publication との間のサブスクリプションを削除します。

```
DROP SYNCHRONIZATION SUBSCRIPTION
```

```
    TO user.sales_publication
    FOR "SSinger";
```

次の例では、FOR 句を省略し、パブリケーション sales_publication のデフォルトの設定を削除します。

```
DROP SYNCHRONIZATION SUBSCRIPTION
    TO user.sales_publication;
```

関連情報

ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [701 ページ]
CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link] [948 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.153 DROP SYNCHRONIZATION USER 文 [Mobile Link]

SQL Anywhere リモートデータベースから 1 人以上の同期ユーザを削除します。

構文

```
DROP SYNCHRONIZATION USER ml-username, ...
```

```
ml-username : identifier
```

備考

Mobile Link リモートデータベースから、1 人以上の同期ユーザを削除します。

ユーザによってサブスクライブされるパブリケーションで参照されるすべてのテーブルへの排他アクセスが必要です。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

ユーザに関連するすべてのサブスクリプションも削除されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

データベースから Mobile Link ユーザ SSinger を削除します。

```
DROP SYNCHRONIZATION USER SSinger;
```

関連情報

ALTER SYNCHRONIZATION USER 文 [Mobile Link] [704 ページ]
CREATE SYNCHRONIZATION USER 文 [Mobile Link] [950 ページ]
SYSSYNC システムビュー [1839 ページ]

1.4.4.154 DROP TABLE 文

データベースからテーブルを削除します。

構文

```
DROP TABLE [ IF EXISTS ] [ owner.]table-name
```

備考

テーブルを削除すると、テーブル内のすべてのデータが削除プロセスの一部として自動的に削除されます。テーブルのすべてのインデックスとキーも削除されます。

IF EXISTS 句は、存在しないテーブルを DROP TABLE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

DROP TABLE 文は、他の接続で現在使用中のテーブルに影響を及ぼす場合は実行できません。テーブルに依存するマテリアライズドビューがある場合も、DROP TABLE 文を実行できません。

DROP TABLE 文を実行すると、すべての通常の従属ビューのステータスが INVALID に変わります。テーブルを削除する前にビューの依存関係を判断するには、sa_dependent_views システムプロシージャを使用します。

グローバルテンポラリテーブルは、このテンポラリテーブルを参照したすべてのユーザが切断されるまで削除できません。

権限

テーブルの所有者であるか、DROP ANY TABLE または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。DROP TABLE によって自動チェックポイントが生じる場合もあります。DROP TABLE 文を実行すると、現在の接続のすべてのカーソルが閉じられます。

DROP TABLE 文は、ローカルテンポラリテーブルを削除するために使用できます。

標準

ANSI/ISO SQL 標準

IF EXISTS 句はコア機能ですが、標準にありません。宣言されたローカルテンポラリテーブルを DROP TABLE 文で削除する機能は、標準にありません。

例

この例では、データベースから架空の MyTable テーブルが削除されています。

```
DROP TABLE MyTable;
```

この例では、データベースから架空の MyTable テーブルが削除されています。IF EXISTS が指定されているため、テーブルが存在しない場合はエラーが返されません。

```
DROP TABLE IF EXISTS MyTable;
```

関連情報

CREATE TABLE 文 [952 ページ]
ALTER TABLE 文 [705 ページ]
sa_dependent_views システムプロシージャ [1487 ページ]

1.4.4.155 DROP TEXT CONFIGURATION 文

テキスト設定オブジェクトを削除します。

構文

```
DROP TEXT CONFIGURATION [ owner.]text-config-name
```

備考

依存するテキストインデックスがあるテキスト設定オブジェクトを削除しようとすると、エラーになります。依存するテキストインデックスを削除してからテキスト設定オブジェクトを削除してください。

テキスト設定オブジェクトは、ISYSTEXTCONFIG システムテーブルに格納されています。

権限

テキスト設定オブジェクトの所有者であるか、DROP ANY TEXT CONFIGURATION または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

次の各文は、mytextconfig というテキスト設定オブジェクトをそれぞれ作成および削除します。

```
CREATE TEXT CONFIGURATION mytextconfig FROM default_char;
DROP TEXT CONFIGURATION mytextconfig;
```

関連情報

DROP TEXT INDEX 文 [1077 ページ]
SYSTEXTCONFIG システムビュー [1849 ページ]
CREATE TEXT CONFIGURATION 文 [975 ページ]
ALTER TEXT CONFIGURATION 文 [720 ページ]

1.4.4.156 DROP TEXT INDEX 文

データベースからテキストインデックスを削除します。

構文

```
DROP TEXT INDEX text-index-name
ON [ owner.]table-name
```

パラメータ

ON 句

この句は、テキストインデックスが構築されたテーブルまたはマテリアライズドビューを指定するときに使用します。

備考

依存するテキストインデックスを削除してから、テキスト設定オブジェクトを削除できます。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

権限

テーブルのテキストインデックスを削除するには、テーブルの所有者であるか、次のいずれかの権限を持っている必要があります。

- そのテーブルに対する REFERENCES 権限
- DROP ANY INDEX システム権限
- DROP ANY OBJECT システム権限

マテリアライズドビューのテキストインデックスを削除するには、マテリアライズドビューの所有者であるか、次のいずれかの権限を持っている必要があります。

- DROP ANY INDEX システム権限
- DROP ANY OBJECT システム権限

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

次の各文は、TextIdx というテキストインデックスをそれぞれ作成および削除します。

```
CREATE TEXT INDEX TextIdx ON GROUPO.MarketingInformation ( Description )
DROP TEXT INDEX TextIdx ON GROUPO.MarketingInformation;
```

関連情報

SYSTEXTCONFIG システムビュー [1849 ページ]
CREATE TEXT INDEX 文 [976 ページ]
ALTER TEXT INDEX 文 [724 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]

1.4.4.157 DROP TIME ZONE 文

データベースからタイムゾーンを削除します。

構文

```
DROP TIME ZONE [ IF EXISTS ] name
```

備考

IF EXISTS 句は、存在しないタイムゾーンを DROP TIME ZONE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

タイムゾーンが time_zone データベースオプションを使用して設定された場合は、そのタイムゾーンは削除できません。

権限

MANAGE TIME ZONE または DROP ANY OBJECT のどちらかのシステム権限が必要です。

関連する動作

オートコミット。

例

この例は、データベースから架空のタイムゾーン MyTimeZone を削除します。

```
DROP TIME ZONE MyTimeZone;
```

関連情報

ALTER TIME ZONE 文 [726 ページ]
COMMENT 文 [769 ページ]
CREATE TIME ZONE 文 [979 ページ]
SYSTIMEZONE システムビュー [1851 ページ]

1.4.4.158 DROP TRACE EVENT 文

ユーザ定義トレースイベントを削除します。

構文

```
DROP TRACE EVENT [ IF EXISTS ] trace-event-name
```

備考

この文はユーザ定義のトレースイベントのみを削除します。IF EXISTS 句は、存在しないトレースイベントを DROP TRACE EVENT 文が削除しようとしたときにエラーを返さないようにする場合に使用します。1 つ以上のイベントトレースセッションがトレースイベントを参照している場合、参照元のトレースセッションが削除されるまでトレースイベントを削除できません。

システム権限

MANAGE ANY TRACE SESSION システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

my_event という名前のトレースイベントを削除します。

```
DROP TRACE EVENT my_event;
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]

sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.4.4.159 DROP TRACE EVENT SESSION 文

トレースイベントセッションを削除します。

構文

```
DROP TRACE EVENT SESSION [ IF EXISTS ] session-name UNCONDITIONALLY
[ ON SERVER ]
```

備考

UNCONDITIONALLY を指定した場合は、アクティブなセッションを削除すると、その定義が削除される前にセッションが自動的に停止します。UNCONDITIONALLY を指定しない場合は、エラーが返されます。

ON SERVER 句を指定すると、データベースサーバ上のすべてのデータベースのトレースイベントを記録しているトレースイベントセッションが削除されます。この句を指定しない場合、現在のデータベース上のトレースイベントセッションが削除されます。

システム権限

MANAGE ANY TRACE SESSION システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、my_session という名前のトレースイベントセッションを削除します。

```
DROP TRACE EVENT SESSION my_session;
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.4.4.160 DROP TRIGGER 文

データベースからトリガを削除します。

構文

```
DROP TRIGGER [ IF EXISTS ] [ owner.] [ table-name.]trigger-name
```

備考

DROP 文が存在しないデータベースオブジェクトを削除しようとする場合にエラーが返されないようにするには、IF EXISTS 句を使用します。

権限

テーブルのトリガを削除するには、次のいずれかの条件に該当する必要があります。

- そのテーブルの所有者である。
- そのテーブルに対する ALTER 権限を持っている。
- ALTER ANY TABLE システム権限を持っている。
- ALTER ANY OBJECT システム権限を持っている。

他のユーザが所有するビュートリガを削除するには、ALTER ANY VIEW または ALTER ANY OBJECT システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

DROP TRIGGER は、オプションの ANSI/ISO SQL 言語機能 T211、"Basic trigger capability" の一部です。IF EXISTS 句は標準にありません。

例

この例は、Employees テーブルの更新前に郵便番号を大文字にする emp_upper_postal_code というトリガを作成した後、削除します。このトリガが存在しない場合は、エラーが返されます。

```
CREATE TRIGGER emp_upper_postal_code
BEFORE UPDATE OF PostalCode
ON GROUPO.Employees
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN ( ISNUMERIC( new_emp.PostalCode ) = 0 )
BEGIN
   -- Ensure postal code is uppercase (employee might be
   -- in Canada where postal codes contain letters)
   SET new_emp.PostalCode = UPPER(new_emp.PostalCode)
END;
DROP TRIGGER MyTrigger;
```

関連情報

CREATE TRIGGER 文 [982 ページ]
ALTER TRIGGER 文 [730 ページ]
ROLLBACK TRIGGER 文 [1287 ページ]

1.4.4.161 DROP USER 文

ユーザを削除します。

構文

```
DROP USER user-name
```

パラメータ

user-name

削除するユーザの名前。

備考

ユーザを削除すると、そのユーザが所有するすべてのデータベースオブジェクト (テーブルやプロシージャなど) とそのユーザの外部ログインも削除されます。さらに、このユーザが、いずれかのサービスの USER 句で指定されている場合は、それらのサービスも削除されます。

この文の実行中は、対象ユーザはデータベースに接続できません。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワードを指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE ANY USER システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、SQLTester というユーザを作成した後に削除します。

```
CREATE USER SQLTester IDENTIFIED BY pass1234;
DROP USER SQLTester;
```

関連情報

ALTER LOGIN POLICY 文 [657 ページ]
ALTER USER 文 [731 ページ]
COMMENT 文 [769 ページ]
CREATE LOGIN POLICY 文 [852 ページ]
CREATE USER 文 [990 ページ]
DROP LOGIN POLICY 文 [1043 ページ]

1.4.4.162 DROP VARIABLE 文

SQL 変数を削除します。

構文

接続スコープ変数の削除:

```
DROP VARIABLE [ IF EXISTS ] identifier
```

データベーススコープ変数の削除

```
DROP DATABASE VARIABLE [ IF EXISTS ] [ owner.]identifier
```

パラメータ

identifier

変数の有効な識別子。

owner

データベーススコープ変数の所有者を指定します。`owner` を指定しない場合、データベースサーバはこの文を実行しているユーザが所有する `identifier` という名前のデータベーススコープ変数を検索します。見つからない場合は、PUBLIC が所有する `identifier` という名前のデータベーススコープ変数を、データベースサーバが検索します。

IF EXISTS 句

この句を指定すると、指定した名前 (および/または指定した場合は所有者) を持つ変数が見つからない場合に、エラーを返さずに文が完了します。

備考

DROP VARIABLE 文は SQL 変数を削除します。

データベース接続が終了すると、接続スコープ変数も自動的に削除されます。データベーススコープ変数は明示的に削除する必要があります。

そのデータベーススコープ変数が削除されたときに文がそのデータベーススコープ変数にアクセスしている場合は、その文のみが変数をメモリ内で引き続き使用できます。

変数は大規模なオブジェクトのためによく使われます。したがって、使用後にこれらを削除したり、NULL に設定したりすると、ディスク領域やメモリなど、相当量のリソースが解放されることがあります。

権限

接続スコープ変数:接続スコープ変数の削除には、権限は不要です。

データベーススコープ変数:自分が所有するデータベーススコープ変数の削除には、権限は不要です。他のユーザまたは PUBLIC が所有するデータベーススコープ変数の削除には、MANAGE ANY DATABASE VARIABLE システム権限が必要です。

関連する動作

接続スコープ変数:接続スコープ変数の削除に関連する動作はありません。

データベーススコープ変数:データベーススコープ変数の削除により、オートコミットが発生します。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

CREATE VARIABLE 文 [992 ページ]

SET 文 [1325 ページ]

1.4.4.163 DROP VIEW 文

データベースからビューを削除します。

構文

```
DROP VIEW [ IF EXISTS ] [ owner.]view-name
```

備考

IF EXISTS 句は、存在しないビューを DROP VIEW 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

DROP VIEW 文を実行すると、すべての通常の従属ビューのステータスが INVALID に変わります。ビューを削除する前にビューの依存関係を判断するには、sa_dependent_views システムプロシージャを使用します。

1 つ以上の INSTEAD OF トリガがあるビューに対して DROP VIEW 文を実行すると、エラーが返されます。トリガを削除してからでないと、ビューを削除または変更できません。

権限

ビューの所有者であるか、DROP ANY VIEW または DROP ANY OBJECT システム権限を持っている必要があります。

関連する動作

オートコミット。DROP VIEW 文を実行すると、現在の接続のすべてのカーソルが閉じられます。

ビューを削除すると、すべてのプロシージャとトリガがメモリからアンロードされます。これにより、削除されたビューを参照するプロシージャやトリガは、そのビューが存在しないことを反映します。ビューの削除や作成を頻繁に行うと、プロシージャとトリガのアンロードやロードによってパフォーマンスが低下することがあります。

標準

ANSI/ISO SQL 標準

コア機能。IF EXISTS 句は標準にありません。

例

次の例は、MyView というビューを作成した後に削除します。この例の CREATE VIEW 文を実行するには、Employees テーブルの SELECT 権限が必要です。

```
CREATE VIEW MyView
   AS SELECT * FROM GROUPO.Employees;
DROP VIEW MyView;
```

関連情報

CREATE VIEW 文 [995 ページ]
ALTER VIEW 文 [734 ページ]
sa_dependent_views システムプロシージャ [1487 ページ]

1.4.4.164 EXCEPT 文

2 つのクエリブロックの集合差を返します。

構文

```
[ WITH temporary-views ] main-query-block
EXCEPT [ ALL | DISTINCT ] except-query-block
[ ORDER BY [ integer | select-list-expression-name ] [ ASC | DESC ], ... ]
[ FOR XML xml-mode ]
[ OPTION( query-hint, ... ) ]
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

```
main-query-block : query-block
```

```
except-query-block :query-block
```

```
option-name : identifier
```

```
option-value :
hostvar （許可されたインジケータ）
| string
| identifier
| number
```

パラメータ

main-query-block

SELECT 文またはクエリ式を構成するクエリブロック (ネストされている可能性あり)。クエリブロックについては、SQL の共通要素のマニュアルを参照してください。

except-query-block

SELECT 文またはクエリ式を構成するクエリブロック (ネストされている可能性あり)。クエリブロックについては、SQL の共通要素のマニュアルを参照してください。

FOR XML 句

この句については、SELECT 文のマニュアルを参照してください。

OPTION 句

この句は、文を実行するためのヒントを指定するときに使用します。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- `option-name` = `option-value`.クエリテキスト内の `OPTION( isolation_level = ... )` の指定は、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。

備考

EXCEPT 文は、main-query-block 内のすべてのロー (except-query-block 内にも存在するローを除く) を返します。main-query-block での重複が、結果で重複として表示されないようにする場合は、EXCEPT または EXCEPT DISTINCT を指定します。それ以外の場合は、EXCEPT ALL を指定します。クエリブロックはネストできます。

EXCEPT 単独での使用は、EXCEPT DISTINCT と同じです。

`main-query-block` および `except-query-block` は和両立する必要があります。それぞれの SELECT リストには同数の項目が含まれており、かつ、各項目の型は比較可能である必要があります。2 つの SELECT リスト内の対応する項目が異なるデータ型の場合、データベースサーバは結果の中から対応するカラムのデータ型を選択し、各 `query-block` のカラムを自動的にそれぞれ変換します。

EXCEPT ALL では、集合差ではなく、バッグ差を実装します。たとえば、`main-query-block` 内に特定の値を持つ 5 つの (重複) ローがあり、`except-query-block` 内に同一の値を持つ 2 つの重複ローがある場合、EXCEPT ALL は 3 つのローを返します。

`main-query-block` 内に重複ローがない場合、EXCEPT の結果は EXCEPT ALL の結果と同じになります。

表示されるカラム名は最初の `query-block` に表示されるカラム名と同じであり、これらの名前は ORDER BY 句と一致する式の名前を判断するために使用されます。結果セットのカラム名をカスタマイズする別の方法として、共通のテーブル式 (WITH 句) を使用する方法があります。

権限

`query-block` で参照されているテーブルを所有しているか、SELECT ANY TABLE 権限を持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

EXCEPT DISTINCT はコア機能です。EXCEPT ALL はオプションの ANSI/ISO SQL 言語機能 F304 を構成します。EXCEPT とともに DISTINCT キーワードを明示的に指定する方法は、オプションの ANSI/ISO SQL 言語機能 T551 です。EXCEPT とともに ORDER BY 句を指定する方法は、ANSI/ISO SQL 言語機能 F850 です。ORDER BY 句を含む `query-block` は、ANSI/ISO SQL 機能 F851 の構成要素です。ロー制限句 (SELECT TOP または LIMIT) を含むクエリブロックは、コンテキストに応じて、オプションの ANSI/ISO SQL 言語機能 F857 または F858 を構成します。FOR XML 句と OPTION 句は標準にありません。

Transact-SQL

EXCEPT は、Adaptive Server Enterprise ではサポートされていません。ただし、SQL Anywhere でサポートされている Transact-SQL ダイアレクトでは、EXCEPT ALL と EXCEPT DISTINCT の両方を使用できます。

関連情報

一般的な SQL 構文要素 [608 ページ]
SELECT 文 [1291 ページ]
INTERSECT 文 [1177 ページ]
UNION 文 [1373 ページ]
SELECT 文 [1291 ページ]

1.4.4.165 EXECUTE IMMEDIATE 文 [SP]

動的に作成された文をプロシージャの中から実行できるようにします。

構文

```
EXECUTE IMMEDIATE [ execute-option ] string-expression
```

```
execute-option :
WITH QUOTES [ ON | OFF ]
| WITH ESCAPES { ON | OFF }
| WITH BATCH { ON | OFF }
| WITH RESULT SET { ON | OFF }
```

パラメータ

WITH QUOTES 句

WITH QUOTES または WITH QUOTES ON を指定すると、文字列式にある二重引用符はすべて識別子を区切るものと見なされます。WITH QUOTES を指定しない場合、または WITH QUOTES OFF を指定する場合、文字列式内の二重引用符は、quoted_identifier オプションの現在の設定に応じて処理されます。

WITH QUOTES は、ストアドプロシージャに渡されるオブジェクト名が、実行される文の一部として使用される場合に役立ちます。ただし、名前に二重引用符が必要な場合や、quoted_identifier オプションが Off に設定されているときにプロシージャが呼び出される場合があります。

WITH ESCAPES 句

WITH ESCAPES OFF を指定すると、文字列式のエスケープシーケンス (¥n、¥x、¥¥ など) がすべて無視されます。たとえば、連続する 2 つの円記号は、1 つの円記号に変換されるのではなく、2 つの円記号として残ります。デフォルトの設定は、WITH ESCAPES ON です。

WITH ESCAPES OFF には、動的に作成された文が円記号を含むファイル名を参照する場合に、この文の実行を容易にするという用途があります。

コンテキストによっては、`string-expression` のエスケープシーケンスは、EXECUTE IMMEDIATE 文が実行される前に変換されます。たとえば、複合文は実行される前に解析されますが、この解析中に WITH ESCAPES の設定とは関係なく、エスケープシーケンスの変換が行われます。これらのコンテキストでは、WITH ESCAPES OFF を指定すると、これ以上は変換されません。例:

```
BEGIN
   DECLARE String1 LONG VARCHAR;
   DECLARE String2 LONG VARCHAR;
   EXECUTE IMMEDIATE
        'SET String1 = ''One backslash: ¥¥¥¥ ''';
     EXECUTE IMMEDIATE WITH ESCAPES OFF
        'SET String2 = ''Two backslashes: ¥¥¥¥ ''';
   SELECT String1, String2
END
```

WITH BATCH 句

WITH BATCH 句を使用して、EXECUTE IMMEDIATE 文でバッチの実行を管理できます。WITH BATCH OFF を設定すると、プロシージャの実行時に誤って SQL を挿入するのを防ぐことができます。

WITH BATCH ON は dbo が所有するプロシージャ以外ではデフォルトです。

WITH BATCH OFF が使用されると、`string-expression` で指定された文は 1 つの文になるはずです。

WITH RESULT SET 句

WITH RESULT SET 句を使用すれば、サーバでこの句を含むプロシージャを正しく定義できます。WITH RESULT SET ON または WITH RESULT SET OFF を指定することは、プロシージャが作成されるときだけではなく、プロシージャが実行されるときの動作についても影響をもたらすことになります。デフォルトオプションは WITH RESULT SET OFF です。

WITH RESULT SET ON を指定することによって、EXECUTE IMMEDIATE 文で結果を返すことができます。この句を使用すると、この句を含むプロシージャが結果セットを返すように指定されます。

備考

EXECUTE IMMEDIATE 文は、プロシージャとトリガの中から実行できる文の種類を増やします。これを使って、プロシージャに渡されたパラメータを使って作成される文のような動的に作成される文を実行できます。

文中のリテラル文字列は一重引用符で囲んでください。文字列リテラルでは、複数行をスキャンできません。

EXECUTE IMMEDIATE によって実行される文では、グローバル変数だけが参照できます。

EXECUTE IMMEDIATE で実行される文には、キャッシュされたプランがありません。

権限

なし。

関連する動作

なし。ただし、文が関連する動作としてのオートコミットを伴うデータ定義文である場合、そのコミットが起こります。

標準

ANSI/ISO SQL 標準

EXECUTE IMMEDIATE は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" です。`execute-option` 構文は標準にありません。ANSI/ISO SQL 標準では、結果セットを返す EXECUTE IMMEDIATE の使用が禁止されています。

例

次の文は、テーブル名がパラメータとしてプロシージャに提供されているテーブルを作成します。

```
CREATE PROCEDURE CreateTableProc(
                  IN tablename char(30)
                  )
BEGIN
   EXECUTE IMMEDIATE
   'CREATE TABLE ' || tablename ||
   ' ( column1 INT PRIMARY KEY)'
END;
```

プロシージャを呼び出して、テーブル mytable を作成します。

```
CALL CreateTableProc( 'mytable' );
```

関連情報

名前付きパラメータ [123 ページ]
CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
BEGIN 文 [745 ページ]
EXECUTE 文 [ESQL] [1093 ページ]
EXECUTE 文 [T-SQL] [1096 ページ]

1.4.4.166 EXECUTE 文 [ESQL]

準備された SQL 文を実行します。

構文

一般的な使用

```
EXECUTE statement
[ USING { hostvar-list | [ SQL ] DESCRIPTOR sqlda-name } ]
[ INTO { into-hostvar-list | [ SQL ] DESCRIPTOR into-sqlda-name } ]
[ ARRAY :row-count ]
```

```
row-count : integer | hostvar
```

```
statement :   identifier | hostvar | string
```

```
sqlda-name : identifier
```

```
into-sqlda-name :  identifier
```

Execute immediately

```
EXECUTE IMMEDIATE statement
```

```
statement :  string | hostvar
```

パラメータ

USING 句

SELECT 文または CALL 文から返される結果は、変数リストの変数の中、または指定した SQLDA が記述するプログラムデータ領域の中のいずれかに入ります。OUTPUT (select リストまたはパラメータ) 対ホスト変数リスト、または OUTPUT 対 SQLDA 記述子配列は、どちらも対応が 1 対 1 です。

INTO 句

EXECUTE INTO を INSERT 文と一緒に使用する場合、挿入されたローは 2 番目の記述子に返されます。たとえば、オートインクリメントプライマリキーを使用するとき、またはプライマリキーの値を生成する BEFORE INSERT トリガを使用するとき、EXECUTE 文により、即座にローを再フェッチし、そのローに割り当てられているプライマリキーの値を決定するためのメカニズムが提供されます。オートインクリメントキーと一緒に @@identity を使用する場合にも同じ結果が得られます。

ARRAY 句

オプションの ARRAY 句を提供された INSERT 文と一緒に使用して、ワイド挿入ができます。ワイド挿入は複数のローを同時に挿入し、パフォーマンスを改善します。整数値は、挿入するローの数です。SQLDA には各エントリの変数 (ロー数 * カラム数) を入れます。最初のローは SQLDA の変数 0 から (ロー当たりのカラム数) -1 に入り、以後のローも同様です。同様に、ARRAY 句は準備済の UPDATE 文、DELETE 文、および MERGE 文によるワイド更新、ワイド削除、およびワイドマージにも使用できます。

備考

EXECUTE 文を使用して SQL 文を作成できます。データベースから多数のローを返す SELECT 文または CALL 文にはカーソルを使用します。

INSERT、UPDATE、DELETE、または MERGE の実行が成功した後、SQLCA (SQLCOUNT) の `sqlerrd[2]` フィールドに、演算によって影響を受けるローの数が入ります。

Execute

以前に作成された名前付きの動的文を実行します。要求についての情報を提供するホスト変数プレースホルダ (バインド変数) が動的文にある場合は、`sqlda-name` に C 変数を指定します。この C 変数は SQLDA へのポインタであり、SQLDA には動的文のすべてのバインド変数のための記述子を含めておきます。この方法以外に、バインド変数を `hostvar-list` で指定する方法もあります。

Execute immediately

バインド変数または出力がない文を PREPARE して EXECUTE するための省略形です。文字列またはホスト変数の中にある SQL 文を即座に実行し、完了時に削除します。

ワイド挿入、ワイド更新、およびワイド削除の操作を実行する場合 (ARRAY 句)、ホスト変数は簡易名とし、ワイド更新のローは同一のデータ型にする必要があります。

権限

必要な権限は、実行する文によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

EXECUTE 文は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部を構成します。INTO 句は、オプションの ANSI/ISO SQL 言語機能 B032、"Extended dynamic SQL" の一部を構成します。ARRAY 句は標準にありません。

また、Embedded SQL でサポートされている EXECUTE IMMEDIATE 文も、オプションの ANSI/ISO SQL 言語機能 B031 の一部です。

例

この例は DELETE 文を実行します。

```
EXEC SQL EXECUTE IMMEDIATE
'DELETE FROM Employees WHERE EmployeeID = 105';
```

この例は準備された DELETE 文を実行します。

```
EXEC SQL PREPARE del_stmt FROM
'DELETE FROM Employees WHERE EmployeeID = :a';
EXEC SQL EXECUTE del_stmt USING :employee_number;
```

この例は準備されたクエリを実行します。

```
EXEC SQL PREPARE sel1 FROM
'SELECT Surname FROM Employees WHERE EmployeeID = :a';
EXEC SQL EXECUTE sel1 USING :employee_number INTO :surname;
```

関連情報

名前付きパラメータ [123 ページ]
EXECUTE IMMEDIATE 文 [SP] [1090 ページ]
PREPARE 文 [ESQL] [1238 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]

1.4.4.167 EXECUTE 文 [T-SQL]

CALL 文の代わりとなる Adaptive Server Enterprise と互換性のあるプロシージャを呼び出して、準備された SQL 文を Transact-SQL で実行します。

構文

ストアドプロシージャの呼び出し

```
[ EXECUTE | EXEC ][ @return_status = ] [creator.]procedure_name
[ argument, ... ]
```

```
argument :
[ @parameter-name = ] expression
| [ @parameter-name = ] @variable [ output ]
```

T-SQL ストアドプロシージャとトリガの中での動的文の実行

```
EXECUTE ( string-expression )
```

備考

ストアドプロシージャを呼び出す構文は、Transact-SQL 互換性のために用意されています。EXECUTE はストアドプロシージャを呼び出します。オプションとしてプロシージャパラメータを指定し、出力された値を取得し、ステータス情報を返します。Watcom SQL では、CALL 文または EXECUTE IMMEDIATE 文を使用します。

文を実行する構文を使用すると、Transact-SQL のストアドプロシージャとトリガの中で動的文を実行できます。EXECUTE 文は、プロシージャとトリガの中から実行できる文の種類を増やします。これを使って、プロシージャに渡されるパラメータを使って作成された文のように、動的に作成された文を実行できます。文の中のリテラル文字列は、一重引用符で囲みます。また、文は 1 行にしてください。この構文は、Transact-SQL 互換性のために用意されていますが、Transact-SQL または Watcom-SQL のバッチとプロシージャにも使えます。

Transact-SQL の EXECUTE 文には、結果セットを予期することを示す方法はありません。Transact-SQL プロシージャが結果セットを返すことを示すには、一例として次のような情報を指定する方法があります。

```
IF 1 = 0
      SELECT 1 AS a
```

Transact-SQL のストアドプロシージャとトリガの中で文を実行することもできます。

権限

プロシージャを呼び出すときは、そのプロシージャの所有者であるか、EXECUTE ANY PROCEDURE システム権限を持っている必要があります。

T-SQL ストアドプロシージャとトリガの中で動的文を実行する場合、必要な権限は実行する文によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

ストアドプロシージャを呼び出す構文は、標準にありません。

動的文を実行する構文は、ANSI/ISO SQL 標準のオプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の EXECUTE IMMEDIATE 文と同じ機能を提供します。ただし、動的文を実行する構文は、ANSI/ISO SQL 標準の動的文を実行する構文とは異なります。

例

次のプロシージャは、ストアドプロシージャの実行方法を示します。

```
CREATE PROCEDURE p1( @var INTEGER = 54 )
AS
PRINT 'on input @var = %1!', @var
DECLARE @intvar integer
SELECT @intvar=123
SELECT @var=@intvar
PRINT 'on exit @var = %1!', @var;
```

次の文は、パラメータに 23 の入力値を提供するプロシージャを実行します。Open Client または JDBC アプリケーションから接続している場合は、クライアントのウィンドウに PRINT メッセージが表示されます。ODBC または Embedded SQL アプリケーションから接続している場合、このメッセージはデータベースサーバメッセージウィンドウに表示されます。

```
EXECUTE p1 23;
```

次の文はプロシージャを実行する代替方法です。これは、パラメータがいくつかあるときに便利です。

```
EXECUTE p1 @var = 23;
```

次の文は、パラメータにデフォルト値を使用するプロシージャを実行します。

```
EXECUTE p1;
```

次の文は、プロシージャを実行し、リターン状況をチェックするために変数へのリターン値を保管します。

```
EXECUTE @status = p1 23;
```

関連情報

名前付きパラメータ [123 ページ]
CALL 文 [756 ページ]

EXECUTE 文 [ESQL] [1093 ページ]
EXECUTE IMMEDIATE 文 [SP] [1090 ページ]

1.4.4.168 EXIT 文 [Interactive SQL]

Interactive SQL を終了します。

構文

```
{ EXIT | QUIT | BYE } [ return-code ]
```

```
return-code : number | connection-variable
```

備考

この文は、Interactive SQL がウィンドウプログラムとして実行している場合には Interactive SQL ウィンドウを閉じ、コマンドプロンプト (バッチ) モードで実行している場合には Interactive SQL を完全に終了します。どちらの場合でも、データベース接続も閉じられます。commit_on_exit オプションが On に設定されている場合は、データベース接続を閉じる前に Interactive SQL が自動的に COMMIT を実行します。このオプションが Off に設定されている場合は、Interactive SQL は暗黙の ROLLBACK を実行します。デフォルトでは、commit_on_exit オプションは On に設定されています。

オプションのリターンコードをバッチファイルでチェックすると、Interactive SQL スクリプトファイル内の文の成功または失敗を確認できます。デフォルトのリターンコードは 0 です。

権限

なし。

関連する動作

この文は、オプション commit_on_exit が On (デフォルト) に設定されている場合は自動的にコミットを実行し、そうでない場合は暗黙のロールバックを実行します。

Windows オペレーティングシステムでは、オプションの戻り値を ERRORLEVEL として使用できます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、テーブル T にローが存在する場合は Interactive SQL の戻り値を 1 に設定し、テーブル T にローがない場合は 0 に設定します。

```
CREATE VARIABLE rowCount INT;
CREATE VARIABLE retcode INT;
SELECT COUNT(*) INTO rowCount FROM GROUPO.Products;
IF( rowCount > 0 ) THEN
    SET retcode = 1;
ELSE
    SET retcode = 0;
END IF;
EXIT retcode;
```

注記

次の文は指定できません。これは、EXIT が (SQL 文ではなく) Interactive SQL 文であり、Interactive SQL 文を他の SQL ブロック文に含めることはできないためです。

```
CREATE VARIABLE rowCount INT;
SELECT COUNT(*) INTO rowCount FROM T;
IF( rowCount > 0 ) THEN
    EXIT 1    //  <-- not allowed
ELSE
    EXIT 0    //  <-- not allowed
END IF;
```

関連情報

SET OPTION 文 [1310 ページ]

1.4.4.169 EXPLAIN 文 [ESQL]

特定のカーソルに使用されている最適化方法のテキスト仕様を取得します。

構文

```
EXPLAIN PLAN FOR CURSOR cursor-name
{  INTO hostvar | USING DESCRIPTOR sqlda-name }
```

```
cursor-name : identifier | hostvar
```

```
sqlda-name :identifier
```

備考

EXPLAIN 文は、指定したカーソルに対する最適化方法のテキスト表現を検索します。事前にカーソルを宣言し開いておきます。

hostvar または sqlda-name 変数には、文字列型を使用してください。最適化文字列は、どのような順序でテーブルを検索するかを指定し、もしあればどのインデックスを検索に使用するかを指定します。

この文字列はクエリによっては長くなることがあり、次のようなフォーマットに従います。

```
table (index), table (index), ...
```

テーブルに相関名が付いている場合、相関名がテーブル名の代わりになります。テーブル名がリストに並ぶ順序は、データベースサーバがそれらにアクセスする順序です。それぞれのテーブルの後には、カッコで囲んだインデックス名が置かれます。これは、テーブルへのアクセスに使用するインデックスです。インデックスを使わない場合 (テーブルを連続的にスキャンします)、英字 "seq" がインデックス名の代わりになります。特別の SQL SELECT 文が部分文字列を伴う場合、コロン (:) でそれぞれのサブクエリの最適化文字列が分割されます。これらのサブクエリセクションは、データベースサーバがクエリを実行する順序で表示されます。

EXPLAIN 文が正常に実行された後で、SQLCA (SQLIOESTIMATE) の sqlerrd フィールドに、クエリのすべてのローをフェッチするのに必要な入出力操作の数の推定値が格納されます。

この文は開いているカーソルでのみ実行できます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、EXPLAIN の使用方法を示します。

```
EXEC SQL BEGIN DECLARE SECTION;
char plan[300];
EXEC SQL END DECLARE SECTION;
EXEC SQL DECLARE employee_cursor CURSOR FOR
   SELECT EmployeeID, Surname
   FROM Employees
   WHERE Surname like :pattern;
EXEC SQL OPEN employee_cursor;
EXEC SQL EXPLAIN PLAN FOR CURSOR employee_cursor INTO :plan;
printf( "Optimization Strategy: '%s'.n", plan );
```

計画変数には以下の文字列が含まれます。

```
'Employees <seq>'
```

関連情報

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
PREPARE 文 [ESQL] [1238 ページ]
FETCH 文 [ESQL] [SP] [1101 ページ]
CLOSE 文 [ESQL] [SP] [767 ページ]
OPEN 文 [ESQL] [SP] [1218 ページ]

1.4.4.170 FETCH 文 [ESQL] [SP]

カーソルを特定のローに配置 (再配置) し、そのローの式の値を、ストアドプロシージャまたはアプリケーション内からアクセスできる変数にコピーします。

構文

ストアドプロシージャ

```
FETCH [ cursor-position ] cursor-name
INTO variable-list [ FOR UPDATE ]
```

Embedded SQL

```
FETCH [ cursor-position ] cursor-name
[ INTO { hostvar-list } | USING [ SQL ] DESCRIPTOR sqlda-name ]
[ PURGE ]
[ BLOCK n ]
[ FOR UPDATE ]
[ ARRAY fetch-count ]
```

```
cursor-position :
    NEXT | PRIOR | FIRST | LAST
| { ABSOLUTE | RELATIVE } row-count
```

```
row-count : number | hostvar
```

```
cursor-name : identifier | hostvar
```

```
hostvar-list: インジケータ変数を含めることができる
```

```
variable-list: ストアドプロシージャ変数
```

```
sqlda-name :  identifier
```

```
fetch-count :  integer | hostvar
```

パラメータ

INTO 句

INTO 句はオプションです。この句が指定されていない場合、FETCH 文はカーソルのみを配置します。hostvar-list は、Embedded SQL でのみ使用されます。

> **注記**
>
> variable-list に含まれる項目が 1 つだけで、カーソルの結果セットに複数の項目がある場合、variable-list がロー変数と見なされ、返されるカラム値が順番にロー変数のフィールドに割り当てられます。

カーソル位置 (cursor position)

オプションの位置パラメータを指定して、カーソルを移動してからローをフェッチできます。指定しない場合、NEXT が使用されます。FETCH 文に配置パラメータがあり、その場所が許可されたカーソルの範囲の外側である場合、SQLE_NOTFOUND の警告が表示され、SQLCOUNT フィールドには有効な位置からのオフセットが設定されます。

OPEN 文は、まず、先頭のローの前にカーソルを配置します。

NEXT 句

デフォルトの位置付けは NEXT です。これはローをフェッチする前に、カーソルを次のローに進めます。

PRIOR 句

ローをフェッチする前に、カーソルを 1 つ前のローに戻します。

RELATIVE 句

RELATIVE 配置を使用すると、フェッチする前にいずれかの方向へ指定した数のローだけカーソルを移動できます。正の数は前進を示し、負の数は後退を示します。したがって、NEXT は RELATIVE 1 と等しく、PRIOR は RELATIVE -1 と等しくなります。RELATIVE 0 は、このカーソル上の最後の FETCH 文と同じローを取り出します。

ABSOLUTE 句

ABSOLUTE 配置パラメータを使用すると、特定のローに移動できます。0 は先頭のローの前の位置を示します。

1 は先頭のローを示し、残りのローがそれに続きます。負の数を使ってカーソルの最後からの絶対位置を指定します。-1 はカーソルの最後のローを示します。

FIRST 句

ABSOLUTE 1 の省略形。

LAST 句

ABSOLUTE -1 の省略形。

> **注記**
>
> DYNAMIC SCROLL カーソルに挿入や更新をいくつか行うと、カーソルの位置の問題が生じます。SELECT 文に ORDER BY 句を指定しないかぎり、データベースサーバはカーソル内の予測可能な位置にはローを挿入しません。場合によっては、カーソルを閉じてもう一度開かないと、挿入したローが表示されないことがあります。
>
> この動作は、カーソルを開くためにテンポラリテーブルを作成する必要がある場合に起こります。
>
> UPDATE 文は、カーソル内でローを移動させることがあります。これは、既存のインデックスを使用する ORDER BY 句がカーソルに指定されている場合に発生します (テンポラリテーブルは作成されません)。

BLOCK 句

クライアントアプリケーションは一度に複数のローをフェッチできます。これはブロックフェッチ、プリフェッチ、またはマルチローフェッチと呼ばれます。最初のフェッチによって、いくつかのローがデータベースサーバから送り返されます。クライアントはこれらのローをバッファに格納します。後に続くフェッチは、データベースサーバへ新しい要求を行わないで、これらのバッファからローを取り出します。

BLOCK 句は Embedded SQL でのみ使用されます。この句は、クライアントとサーバに、アプリケーションがフェッチできるローの個数に関する情報を与えます。0 という特別値は、要求をデータベースサーバに送信し、シングルローを返すことを示します (ローブロックはありません)。BLOCK 句を指定すると、BLOCK 値を次にプリフェッチするときに含められるローの数が削減されます。プリフェッチされるローの数を増やすには、PrefetchRows 接続パラメータを使用します。

BLOCK 句を指定しない場合は、OPEN に指定した値が使用されます。

FETCH RELATIVE 0 は常にローを再フェッチします。

カーソルのプリフェッチが無効な場合、BLOCK 句は無視され、ローは一度に 1 つずつフェッチされます。ARRAY も指定している場合、ARRAY で指定されたロー数がフェッチされます。

PURGE 句

PURGE 句は Embedded SQL でのみ使用されます。この句によって、クライアントはすべてのローのバッファをフラッシュし、次にフェッチ要求をデータベースサーバに送信します。このフェッチ要求はローのブロックを返します。

FOR UPDATE 句

FOR UPDATE 句は、フェッチされたローが続いて UPDATE WHERE CURRENT OF CURSOR 文によって更新されることを示します。この句は、データベースサーバがローに対して意図的なロックを設定するようにします。ロックは、現在のトランザクションの終わりまで保持されます。

ARRAY 句

ARRAY 句は Embedded SQL でのみ使用されます。この句を使うと、いわゆるワイドフェッチができるようになります。これは複数のローを同時に取り出し、パフォーマンスを改善します。

Embedded SQL でワイドフェッチを使用するには、次のような FETCH 文をコードに含めます。

```
EXEC SQL FETCH ... ARRAY nnn
```

ARRAY nnn は FETCH 文の最後の項目です。フェッチ回数を示す nnn にはホスト変数も使用できます。SQLDA には nnn * (ローあたりのカラム数) 変数を指定する必要があります。最初のローは SQLDA の変数 0 から (ロー当たりのカラム数) -1 に入り、以後のローも同様です。

備考

FETCH 文は指定したカーソルからローを 1 つ取り出します。事前にカーソルを開いておきます。

テーブル参照変数 (タイプ TABLE REF として定義される変数) の FETCH 文での使用は、サポートしていません。

Embedded SQL での使用

Embedded SQL の FETCH 文は配列をサポートしていません。

DECLARE CURSOR 文は、C ソースコード内の FETCH 文の前に置きます。また、OPEN 文を実行してから FETCH 文を実行します。ホスト変数をカーソル名の代わりに使用している場合、DECLARE 文は実際にコードを生成するため、FETCH 文の前に実行します。

サーバは SQLCOUNT にフェッチされたレコードの数を返し、エラーがない場合は 0 より大きい SQLCOUNT を常に返します。

フェッチ時に SQLSTATE_NOTFOUND 警告が返される場合、SQLCA (SQLCOUNT) の sqlerrd[2] フィールドには、フェッチの試みが許可されるカーソル位置を超えたときのローの数が含まれます。ローが見つからなくても位置が有効な場合は、値は 0 です。たとえば、カーソル位置が最後のローのときに FETCH RELATIVE 1 を実行した場合です。カーソルの最後を超えてフェッチしようとした場合、値は正の数です。カーソルの先頭を超えてフェッチしようとした場合、値は負の数です。カーソルの最後を超えてフェッチしようとした場合、カーソルの位置は最後のローになります。カーソルの先頭より前をフェッチしようとした場合、カーソルの位置は先頭のローになります。

FETCH 文の実行が成功した後、SQLCA (SQLIOCOUNT) の sqlerrd[1] フィールドはフェッチを実行するのに必要な入出力操作の数だけ増加します。このフィールドは、実際にはデータベース文が発行されるたびに増加します。

シングルローフェッチ

SELECT 文の結果からの 1 つのローは、変数リストの変数の中に置かれます。SELECT リスト対ホスト変数リストは、1 対 1 に対応します。

マルチローフェッチ

SELECT 文の結果からの 1 つまたは複数のローは、variable-list の変数の中、または指定した sqlda-name が記述するプログラムデータ領域の中のいずれかに置かれます。SELECT リスト対 hostvar-list または対 sqlda-name 記述子配列は、どちらも 1 対 1 で対応します。

権限

カーソルが開いている必要があり、テーブルに対する SELECT オブジェクトレベル権限を持っているか、カーソルの宣言で参照されているテーブルの所有者であるか、SELECT ANY TABLE システム権限を持っている必要があります。

関連する動作

プリフェッチが有効な場合、FETCH 文によって、複数のローがサーバからクライアントに取得されるようになる場合があります。

標準

ANSI/ISO SQL 標準

若干の例外がありますが、FETCH 文のストアド プロシージャ構文は ANSI/ISO SQL のコア機能です。NEXT 以外のスクロールオプションは、オプションの ANSI/ISO SQL 言語機能 F431、"Read-only scrollable cursors" を構成します。ソフトウェアでは、SQL/2008 標準に記載されているように、FETCH 文のオプションの FROM 句はサポートされていません。

Embedded SQL 構文は標準にありません。

FOR UPDATE、PURGE、ARRAY、BLOCK、USING [SQL] DESCRIPTOR 句は標準にありません。

例

次は、Embedded SQL の例です。

```
EXEC SQL DECLARE cur_employee CURSOR FOR
SELECT EmployeeID, Surname FROM Employees;
EXEC SQL OPEN cur_employee;
EXEC SQL FETCH cur_employee
INTO :emp_number, :emp_name:indicator;
```

次は、プロシージャの例です。

```
BEGIN
   DECLARE cur_employee CURSOR FOR
      SELECT Surname
      FROM Employees;
   DECLARE name CHAR(40);
   OPEN cur_employee;
   lp: LOOP
      FETCH NEXT cur_employee into name;
      IF SQLCODE <> 0 THEN LEAVE lp END IF;
       ...
   END LOOP;
   CLOSE cur_employee;
END
```

関連情報

SELECT 文 [1291 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
FOR 文 [1106 ページ]

PREPARE 文 [ESQL] [1238 ページ]
OPEN 文 [ESQL] [SP] [1218 ページ]
RESUME 文 [1271 ページ]

1.4.4.171 FOR 文

文リストの実行をカーソル内のローごとに 1 回繰り返します。

構文

```
[ statement-label : ]
FOR for-loop-name AS cursor-name [ cursor-type ] CURSOR
 { FOR statement [ FOR { UPDATE [ cursor-concurrency ] | READ ONLY } ]
   | USING variable-name }
DO statement-list
END FOR [ statement-label ]
```

```
cursor-type :
NO SCROLL
| DYNAMIC SCROLL
| SCROLL
| INSENSITIVE
| SENSITIVE
```

```
cursor-concurrency : BY { VALUES | TIMESTAMP | LOCK }
```

```
variable-name : identifier
```

パラメータ

NO SCROLL 句

NOSCROLL として宣言されたカーソルは、FETCHNEXT と FETCHRELATIVE0 の各シーク操作を使用して、結果セットの前方移動に制限されます。

カーソルがローを出た後は、そのローに戻ることができないため、カーソルに sensitivity の制限はありません。NO SCROLL カーソルを要求されると、データベースサーバ は最も効率的な種類のカーソルである asensitive カーソルを提供します。

DYNAMIC SCROLL 句

DYNAMIC SCROLL はデフォルトカーソルタイプです。DYNAMIC SCROLL カーソルでは、FETCH 文のすべてのフォーマットを使用できます。

DYNAMIC SCROLL カーソルを要求されると、データベースサーバは asensitive カーソルを提供します。カーソルを使用する場合、効率と一貫性の間には常にトレードオフ関係があります。Asensitive カーソルを使用すると、パフォーマンスは向上しますが一貫性が低下します。

SCROLL 句

SCROLL として宣言されたカーソルは、FETCH 文のすべてのフォーマットを使用できます。SCROLL カーソルを要求されると、データベースサーバは value-sensitive カーソルを提供します。

データベースサーバは、結果セットのメンバーシップが保証されるような方法で value-sensitive カーソルを実行する必要があります。DYNAMIC SCROLL カーソルの方が効率的です。SCROLL カーソルの一貫した動作が必要でない場合は、DYNAMIC SCROLL カーソルを使用してください。

INSENSITIVE 句

INSENSITIVE として宣言されたカーソルは、存続期間中、値とメンバーシップが固定されます。SELECT 文の結果セットはカーソルが開かれると、実体化されます。INSENSITIVE カーソルからの FETCHING は、そのカーソルを開いた接続を含むどの接続からの INSERT、UPDATE、MERGE、PUT、DELETE 文の影響も受けません。

SENSITIVE 句

SENSITIVE として宣言されたカーソルは、結果セットのメンバーシップまたは値の変更に依存します。

FOR UPDATE 句

FOR UPDATE はデフォルトです。ORDER BY 句なしの単一テーブルのクエリの場合、または ansi_update_constraints オプションが Off に設定されている場合、カーソルのデフォルトは FOR UPDATE です。ansi_update_constraints オプションが Cursors または Strict に設定されている場合、ORDER BY 句を含むクエリ上のカーソルのデフォルトは READ ONLY です。ただし、FOR UPDATE 句を使用して、カーソルを更新可能と明示的に示すこともできます。

FOR READ ONLY 句

FOR READ ONLY として宣言されたカーソルは、UPDATE (位置付け) 文、DELETE (位置付け) 文、または PUT 文には使用できません。ORDER BY 句またはジョインを使用してカーソルに対する更新を可能にするにはコストがかかるため、2 つ以上のテーブルのジョインを含むクエリに対するカーソルは READ ONLY であり、ansi_update_constraints データベースオプションが Off でないかぎり、更新可能にすることはできません。FOR UPDATE を指定したカーソル要求への応答では、データベースサーバは value-sensitive カーソルまたは sensitive カーソルを提供します。insensitive カーソルと asensitive カーソルは更新できません。

備考

FOR 文は制御文です。これを使うと、カーソル内の各ローに対して一度だけ、SQL 文のリストを実行できます。FOR 文は、カーソルの DECLARE と、カーソルの結果セットのそれぞれのカラムに対する変数の DECLARE があり、その後には、カーソルから 1 つのローをフェッチしてローカル変数に格納し、カーソル内の各ローに対して一度だけ `statement-list` を実行するループが続く複合文と同じです。

有効なカーソルタイプには、dynamic scroll (デフォルト)、scroll、no scroll、sensitive、insensitive があります。

それぞれのローカル変数の名前とデータ型は、カーソル内で使用する `statement` から取り出されます。SELECT 文の場合、データ型は SELECT リスト内の式のデータ型です。名前は、SELECT リスト項目のエイリアスが存在する場合はエイリアスになります。そうでない場合は、カラムの名前となります。SELECT リスト項目のカラム参照が簡単でない場合は、エイリアスを指定します。CALL 文の場合、名前とデータ型はプロシージャ定義内の RESULT 句から取ります。

LEAVE 文を使用して、END FOR に続く最初の文から実行を再開できます。終了の `statement-label` を指定する場合は、開始の `statement-label` と一致させる必要があります。

FOR 文によって作成されるカーソルは暗黙的に WITH HOLD を使用して開かれるため、ループ内で実行され、COMMIT を発生させる文によってこのカーソルは閉じられません。

⚠ 警告

cursor-name を指定しない場合、cursor-type は cursor-name として使用されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

FOR 文は、オプションの ANSI/ISO SQL 言語機能 P002、"Computational completeness" の一部です。FOR 文の USING 句は標準にありません。DECLARE CURSOR 文では、カーソル感知性とカーソルスクロール動作のオプションの組み合わせと同様、cursor-concurrency の使用は標準にありません。

例

次のフラグメントは、FOR ループの使い方を示します。

```
FOR names AS curs INSENSITIVE CURSOR FOR
SELECT Surname
FROM Employees
DO
   CALL search_for_name( Surname );
END FOR;
```

このフラグメントは、FOR ループの使い方も示します。

```
BEGIN
    FOR names AS curs SCROLL CURSOR FOR
    SELECT EmployeeID, GivenName FROM Employees where EmployeeID < 130
    FOR UPDATE BY VALUES
    DO
        MESSAGE 'emp: ' || GivenName;
    END FOR;
END
```

次の例は、プロシージャ myproc の内部で使用している FOR ループを示します。このプロシージャは、プロシージャを呼び出したときに指定したソート順に従って、Employees テーブルから上位 10 名の従業員を返します (asc は昇順、desc は降順)。

```
CALL sa_make_object( 'procedure', 'myproc' ) ;
```

```
ALTER PROCEDURE myproc (
    IN @order_by VARCHAR(20) DEFAULT NULL
)
RESULT ( Surname person_name_t )
BEGIN
    DECLARE @sql LONG VARCHAR;
    DECLARE @msg LONG VARCHAR;
    DECLARE LOCAL TEMPORARY TABLE temp_names( surnames person_name_t );
    SET @sql = 'SELECT TOP(10) * FROM Employees AS t ' ;
    CASE @order_by
    WHEN 'asc' THEN
        SET @sql = @sql || 'ORDER BY t.Surname ASC';
        SET @msg = 'Sorted ascending by last name: ';
    WHEN 'desc' THEN
        SET @sql = @sql || 'ORDER BY t.Surname DESC';
        SET @msg = 'Sorted ascending by last name: ';
    END CASE;
    FOR loop_name AS curs SCROLL CURSOR USING @sql
    DO
        INSERT INTO temp_names( surnames ) VALUES( Surname );
        MESSAGE( @msg || Surname ) ;
    END FOR;
    SELECT * FROM temp_names;
END ;
```

asc を指定して myproc プロシージャを呼び出すと (たとえば、`CALL myproc( 'asc' );`)、次のような結果が返されます。

Surname
Ahmed
Barker
Barletta
Bertrand
Bigelow
Blaikie
Braun
Breault
Bucceri
Butterfield

関連情報

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
FETCH 文 [ESQL] [SP] [1101 ページ]
CONTINUE 文 [778 ページ]
LOOP 文 [1203 ページ]

1.4.4.172 FORWARD TO 文

ネイティブ構文の SQL 文をリモートサーバに送信します。

構文

1 つの文の転送

```
FORWARD TO server-name sql-statement
```

パススルーモードの入力

```
FORWARD TO [ server-name ]
```

パススルーモードの終了

```
FORWARD TO
```

パラメータ

server-name

リモートサーバの名前。

sql-statement

リモートサーバのネイティブ SQL 構文によるコマンド。コマンドまたは一連のコマンドは、波括弧 ({}) または一重引用符で囲まれます。

備考

FORWARD TO 文は、ユーザがパススルー接続が必要なサーバを指定できるようにします。この文は、次の 2 つの方法で使用できます。

1 つの文を転送する構文

1 つの文をリモートサーバに送信します。

パススルーモードの構文

データベースサーバをパススルーモードにして、一連の文をリモートサーバに送信します。それ以降の文はすべて直接リモートサーバに渡されます。パススルーモードをオフにするには、`server-name` を指定しないで FORWARD TO を実行します。

パススルーモードでリモートサーバからエラーが返された場合は、FORWARD TO 文を実行してパススルーモードをオフにしてください。

プロキシテーブル:ユーザが構文を使用してリモートサーバに接続するには、ログインする必要があります。別のユーザが所有するプロシージャ内に FORWARD TO 文が出現する場合、またはイベント (またはイベントによって呼び出される

プロシージャ) 内に FORWARD TO 文が出現する場合、認証は行われず、パスワードは転送されません。プロキシテーブルを使用する唯一の方法は、EXTERNLOGIN を定義することです。

ユーザのために server-name への接続を確立する場合、データベースサーバは次のいずれかを使用します。

- CREATE EXTERNLOGIN を使用するリモートログインエイリアスセット
- リモートログインエイリアスが設定されない場合は、データベースサーバとの通信に使用される名前とパスワード

指定のサーバに接続を確立できない場合は、その理由を示すメッセージがユーザに返されます。

文が要求されたサーバに渡されると、結果はすべて、クライアントプログラムで認識できる形式に変換されます。

注記

FORWARD TO 文はサーバディレクティブであり、ストアドプロシージャ、トリガ、イベント、またはバッチ内では使用できません。しかしながら、dbo.sp_forward_to_remote_server() システムプロシージャは、ストアドプロシージャ、トリガ、イベント、またはバッチで使用できます。

権限

なし

関連する動作

リモート接続は、FORWARD TO セッションの間、AUTOCOMMIT (非連鎖) モードに設定されます。FORWARD TO 文の前に保留中であった作業はすべて自動的にコミットされます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、SQL 文をリモートサーバ RemoteASE に送信します。

```
FORWARD TO RemoteASE { SELECT * FROM titles };
```

次の例は、リモートサーバ aseprod でのパススルーセッションを示します。

```
FORWARD TO aseprod;
    SELECT * FROM titles;
    SELECT * FROM authors;
FORWARD TO;
```

関連情報

PASSTHROUGH 文 [SQL Remote] [1230 ページ]
sp_forward_to_remote_server システムプロシージャ [1679 ページ]

1.4.4.173 FROM 句

DELETE、SELECT、または UPDATE 文に必要なデータベーステーブルまたはビューを指定します。SELECT 文内で使用される場合、FROM 句は MERGE 文または INSERT 文でも使用できます。

構文

```
FROM table-expression,...
```

```
table-expression :
table-name
| view-name
| procedure-name
| derived-table
| lateral-derived-table
| pivoted-derived-table
| unpivoted-derived-table
| join-expression
| ( table-expression,... )
| openstring-expression
| apply-expression
| contains-expression
| dml-derived-table
| openxml-operator
| array-operator
```

```
table-name :
[ userid.]table-name
[ [ AS ] correlation-name ]
[ WITH ( hint [...] ) ]
[ FORCE INDEX ( index-name ) ]
```

```
view-name :
[ userid.]view-name [ [ AS ] correlation-name ]
[ WITH ( table-hint ) ]
```

```
procedure-name :
[ owner.]procedure-name ( [ parameter,... ] )
[ WITH ( column-name data-type,... ) ]
[ [ AS ] correlation-name ]
```

```
derived-table :
( select-statement )
[ AS ] correlation-name [ ( column-name,... ) ]
```

```
lateral-derived-table :
LATERAL ( select-statement | table-expression )
[ AS ] correlation-name [ ( column-name,... ) ]
```

```
pivoted-derived-table :
pivot-source-table PIVOT [ XML ] ( pivot-clause ) [ AS ] pivoted-correlation-name
```

```
unpivoted-derived-table :
unpivoted-derived-table :
unpivot-source-table UNPIVOT [ { INCLUDE | EXCLUDE } NULLS ]  ( unpivot-clause )
[ AS ] correlation-name
```

```
join-expression :
table-expression join-operator table-expression
[ ON join-condition ]
```

```
join-operator :
[ KEY | NATURAL ] [ join-type ] JOIN
| CROSS JOIN
```

```
join-type :
INNER
| LEFT [ OUTER ]
| RIGHT [ OUTER ]
| FULL [ OUTER ]
```

```
hint :
table-hint | index-hint
```

```
table-hint :
READPAST
| UPDLOCK
| XLOCK
| FASTFIRSTROW
| HOLDLOCK
| NOLOCK
| READCOMMITTED
| READUNCOMMITTED
| REPEATABLEREAD
| SERIALIZABLE
```

```
index-hint :
NO INDEX
| INDEX ( [ PRIMARY KEY | FOREIGN KEY ] index-name [, ..] ) [ INDEX ONLY { ON |
OFF } ]
| CLUSTERED INDEX [ INDEX ONLY { ON | OFF } ]
```

```
openstring-expression :
OPENSTRING ( { FILE | VALUE } string-expression )
WITH ( rowset-schema )
 [ OPTION ( scan-option ...  ) ]
 [ AS ] correlation-name
```

```
apply-expression :
table-expression { CROSS | OUTER } APPLY table-expression
```

```
contains-expression :
{ table-name | view-name } CONTAINS ( column-name [,...], contains-query )
[ [ AS ] score-correlation-name ]
```

```
rowset-schema :
column-schema-list
| TABLE [owner.]table-name [ ( column-list ) ]
```

```
column-schema-list :
{ column-name user-or-base-type |  filler( ) } [ ,... ]
```

```
column-list :
{ column-name | filler( ) } [ ,... ]
```

```
scan-option :
BYTE ORDER MARK { ON | OFF }
| COMMENTS INTRODUCED BY comment-prefix
| { COMPRESSED | AUTO | NOT COMPRESSED }
| DELIMITED BY string
| ENCODING encoding
| { ENCRYPTED KEY key-expression | NOT ENCRYPTED }
| ESCAPE CHARACTER character
| ESCAPES { ON | OFF }
| FORMAT { TEXT | BCP }
| HEXADECIMAL { ON | OFF }
| QUOTE string
| QUOTES { ON | OFF }
| ROW DELIMITED BY string
| SKIP integer
| STRIP { ON | OFF | LTRIM | RTRIM | BOTH }
```

```
key-expression : string | variable
```

```
contains-query : string
```

```
dml-derived-table :
( dml-statement ) REFERENCING ( [ table-version-names | NONE ] )
```

```
dml-statement :
insert-statement
delete-statement
update-statement
merge-statement
```

```
table-version-names :
OLD [ AS ] correlation-name [ FINAL [ AS ] correlation-name ]
| FINAL [ AS ] correlation-name
```

パラメータ

table-name

ベーステーブルまたはテンポラリテーブル。ユーザ ID を指定すると、別のユーザが所有するテーブルを修飾できます。ユーザ ID を指定しないと、ユーザが被付与者となるユーザ定義ロールに所有されるテーブルがデフォルトで検索されます。

view-name

クエリに含めるビューを指定します。テーブルの場合と同様、ユーザ ID を指定すると、別のユーザが所有するビューを修飾できます。ユーザ ID を指定しない場合は、デフォルトで、現在のユーザの所属グループが所有するビューを参照します。構文ではビューに対してテーブルヒントを指定できますが、これらのヒントの効果はありません。

procedure-name

結果セットを返すストアドプロシージャです。この句は SELECT 文の FROM 句にのみ適用されます。プロシージャにパラメータを指定しない場合でも、プロシージャ名の後のカッコは必要です。オプションのパラメータの代わりに DEFAULT を指定できます。

引数リストは、引数の順序を守って指定するか、キーワードフォーマットを使うことにより指定できます。順序を守って指定した場合、引数はプロシージャのパラメータリスト内の対応するパラメータと一致します。キーワードを使用する場合、引数名にはパラメータ名を使用します。

ストアドプロシージャが複数の結果セットを返す場合、最初の結果セットだけが使用されます。

WITH 句を指定すると、プロシージャの結果セットにカラム名のエイリアスを指定できます。WITH 句を指定する場合、カラム数はプロシージャの結果セットのカラム数と一致し、データ型はプロシージャの結果セットのデータ型と互換性がある必要があります。WITH 句を指定しない場合、カラム名と型はプロシージャ定義で設定された名前と型です。

Embedded SQL アプリケーションの場合、RESULT 句を使用しないでプロシージャを作成し、そのプロシージャが変数結果セットを返す場合には、プロシージャを参照する SELECT 文の DESCRIBE が失敗することがあります。DESCRIBE の失敗を防ぐには、予測される結果セットスキーマを記述する WITH 句を含めます。

プロシージャを選択するとテンポラリテーブルが生成されます。たとえば、次のクエリはローカルのテンポラリテーブルを作成します。

```
BEGIN
...
    my: LOOP
      BEGIN
        SELECT TOP 1 NUMBER INTO conn_id FROM sa_conn_info( ) ORDER BY NUMBER
DESC;
      END;
...
   END LOOP my;
END
```

derived-table

FROM 句の中で、テーブル名またはビュー名の代わりに SELECT 文を指定できます。このように使用する SELECT 文を派生テーブルと呼び、エイリアスを指定する必要があります。たとえば、次の文には、派生テーブル MyDerivedTable が含まれています。この派生テーブルは、Products テーブル内の製品を UnitPrice でランキングしています。

```
SELECT TOP 3 *
       FROM ( SELECT Description,
                     Quantity,
                     UnitPrice,
                     RANK() OVER ( ORDER BY UnitPrice ASC )
                 AS Rank
                 FROM GROUPO.Products ) AS MyDerivedTable
ORDER BY Rank;
```

lateral-derived-table

親の文のオブジェクトへの参照 (外部参照) が含まれている場合がある、派生テーブル、ストアドプロシージャ、またはジョインしたテーブル。FROM 句の外部参照を使用する場合は、ラテラル派生テーブルを使用します。

外部参照を使用できるのは、FROM 句のラテラル派生テーブルの前のテーブルに対してだけです。たとえば、SELECT リストの項目に外部参照は使用できません。

テーブルと外部参照は、カンマで区切ります。たとえば、次のクエリは有効です。

```
SELECT *
 FROM A, LATERAL( B LEFT OUTER JOIN C ON ( A.x = B.x ) ) myLateralDT;
```

```
SELECT *
 FROM A, LATERAL( SELECT * FROM B WHERE A.x = B.x ) myLateralDT;
```

```
SELECT *
 FROM A, LATERAL( procedure-name( A.x ) ) myLateralDT;
```

LATERAL (`table-expression`) の指定は、LATERAL (SELECT * FROM `table-expression`) の指定と同等です。

pivoted-derived-table FROM `table-name` 式のデータを行列変換された派生テーブルに変換します。この句の完全な構文は、PIVOT 句のトピックで説明されています。

unpivoted-derived-table FROM `table-name` 式のデータを列行変換された派生テーブルに変換します。この句の完全な構文は、UNPIVOT 句のトピックで説明されています。

openstring-expression

OPENSTRING 句は、ファイルまたは BLOB 内を問い合わせ、これらのソースの内容をローセットとして処理するときに指定します。テーブルやビューなどの定義された構造をクエリするわけではないため、この句を指定するときは、ファイルまたは BLOB のスキーマ情報も指定して結果セットを生成します。この句は SELECT 文の FROM 句に適用されます。UPDATE 文または DELETE 文ではサポートされません。

ROWID 関数は、OPENSTRING 式で生成されたテーブルの結果セットでサポートされます。

次に示す OPENSTRING 句のサブ句とパラメータは、ファイルと BLOB 内のデータを定義し、問い合わせるために使用します。

FILE 句と VALUE 句

FILE 句は、問い合わせるファイルを指定するときに使用します。VALUE 句は、問い合わせる BLOB 式を指定するときに使用します。BLOB 式のデータ型は、LONG BINARY と見なされます。VALUE 句の値として、READ_CLIENT_FILE 関数を指定できます。

FILE キーワードと VALUE キーワードをどちらも指定しないと、VALUE が指定されたと見なされます。

FORMAT SHAPEFILE を使用した場合、FILE のみが指定されたと見なされます。

WITH 句

この句は、問い合わせられるデータのローセットスキーマ (カラム名とデータ型) を指定するときに使用します。カラムは直接指定できます (たとえば、`WITH ( Surname CHAR(30), GivenName CHAR(30) )`)。また、TABLE サブ句を使用して、スキーマ情報の取得元に使用するテーブルを参照できます (たとえば、`WITH ( TABLE dba.Employees ( Surname, GivenName ) )`)。指定するテーブルの所有者であるか、または SELECT 権限が必要です。

カラムを指定する場合は、入力データ内でスキップするカラムに filler( ) を指定できます (たとえば、`WITH ( filler( ), Surname CHAR(30), GivenName CHAR(30) )`)。

OPTION 句

OPTION 句は、エスケープ文字、デリミタ、エンコードなど、入力ファイルに使用する解析オプションを指定するときに使用します。サポートされるオプションは、入力ファイルの解析を制御する LOAD TABLE 文のオプションで構成されます。

scan-option

各スキャンオプションの詳細については、LOAD TABLE 文のロードオプションを参照してください。

apply-expression

この句は、左の table-expression のローごとに、右の table-expression を評価するジョイン条件を指定するときに使用します。たとえば、適用式を使用して、テーブル式のローごとに関数、プロシージャ、または派生テーブルを評価できます。

contains-expression

テーブル名の後に付ける CONTAINS 句は、テーブルをフィルタして、contains-query で指定した全文クエリに一致するローのみを返すときに使用します。テーブルの一致するすべてのローが、score-correlation-name を使用して参照できるスコアカラムとともに返されます。score-correlation-name を指定しない場合は、デフォルトの相関名 contains によってスコアカラムを参照できます。

オプションの相関名の引数を例外として、CONTAINS 句は、CONTAINS 検索条件と同じ引数を取ります。

CONTAINS 句にリストされるカラムには、テキストインデックスが必要です。

contains-query は、NULL または空の文字列に設定できません。テキスト設定オブジェクトの設定により contains-query 内のすべての単語が削除されるようになっている場合、contains-expression によって参照されるベーステーブルのローは返されません。

correlation-name

correlation-name は、FROM 句の中でテーブルまたはビューに代替名を指定するときに使用します。この代替名は、文のどこからでも参照できます。たとえば、emp と dep はそれぞれ、Employees テーブルと Departments テーブルの相関名です。

```
SELECT Surname, GivenName, DepartmentName
   FROM GROUPO.Employees emp, GROUPO.Departments dep,
   WHERE emp.DepartmentID=dep.DepartmentID;
```

相関名を使用して、同じテーブルの異なるインスタンスを区別できます。たとえば、次のクエリは Employee テーブルをそれ自体にジョインし、相関名 Mgr を使用して、従業員の各マネージャの姓を結果に含めます。

```
SELECT Emp.EmployeeID, Emp.Surname, Dept.DepartmentName, Mgr.Surname AS
ManagerName
    FROM GROUPO.Employees AS Emp, GROUPO.Departments AS Dept, GROUPO.Employees
AS Mgr
    WHERE Emp.DepartmentID = Dept.DepartmentID AND Emp.ManagerID =
Mgr.EmployeeID;
```

dml-statement

dml-statement を使用して、ローの選択に使用する DML 文 (INSERT、DELETE、UPDATE、または MERGE) を指定します。実行時には、dml-derived-table で指定された DML 文が最初に実行され、その DML の影響を受けるローが、REFERENCING 句で記述されているカラムを含むテンポラリテーブルにマテリアライズされます。テンポラリテーブルは、dml-derived-table の結果セットを表します。

クエリ内で参照しないために結果をテンポラリテーブルにマテリアライズする必要がない場合は、REFERENCING ( ) または REFERENCING (NONE) を使用します。

REFERENCING ( ) または REFERENCING (NONE) を指定した場合、更新されたローはクエリで参照されないため、dml-derived-table の結果セットを表すテンポラリテーブルにマテリアライズされません。この場合のテンポラリテー

ブルは空のテーブルになります。メインの文が実行される前に dml-statement が実行されるようにするには、この機能を使用します。

結果の OLD カラムには、更新操作の対象にするローを検索するスキャンで見つかった値が格納されます。FINAL カラムには、参照整合性チェックが実行され、計算カラムとデフォルトカラムが更新され、すべてのトリガ (FOR STATEMENT タイプの AFTER トリガを除く) が起動された後に、値が格納されます。

文	サポートされているテーブルバージョン
INSERT	FINAL
DELETE	OLD
UPDATE	FINAL か OLD またはその両方
MERGE	FINAL か OLD またはその両方

OLD と FINAL の両方の名前を指定した場合、2 つの相関名が使用されます。ただし、これらの名前はどちらも同じ結果セットを示すため、本当の相関ではありません。REFERENCING (OLD AS O FINAL AS F) と指定した場合、暗黙的なジョイン述部 O.rowid = F.rowid が使用されます。

INSERT 文では、FINAL のみがサポートされています。したがって、INSERT ON EXISTING UPDATE 文によって変更された更新ローの値は、派生テーブルの結果セットには含まれません。代わりに、MERGE 文を使用して insert-else-update 処理を実行します。

dml-derived-table 文は更新可能なテーブルを 1 つのみ参照できます。複数のテーブルに対する更新の場合は、エラーが返されます。また、DML 文が相関サブクエリまたは共通テーブル式に含まれている場合、これらの構成体のセマンティックが不明確になる可能性があるため、dml-statement からの選択は許可されません。

openxml-operator

OPENXML 演算子を使用して XML ドキュメントから結果セットを返すには、このパラメータを使用します。

array-operator

UNNEST などの配列演算子を使用して ARRAY から結果セットを返すには、このパラメータを使用します。

WITH table-hint 句

WITH table-hint 句を使用すると、このテーブルでのみ使用できる動作と、この文でのみ使用できる動作を指定できます。この句を使用すると、独立性レベルを変更せずに、またデータベースまたは接続のオプションを設定せずに、動作を変更できます。テーブルヒントは、ベーステーブル、テンポラリテーブル、およびマテリアライズドビューに使用できます。

> ⚠ 警告
>
> WITH table-hint 句は高度な機能です。必要な場合にかぎり、経験を有するデータベース管理者のみが使用してください。また、この設定はすべての状況で適用されるとはかぎりません。

独立性レベル関連のテーブルヒント

独立性レベルのテーブルヒントは、テーブルのクエリを実行するときに独立性レベルの動作を指定する場合に使用します。また、指定したテーブルと現在のクエリにのみ使用するロック方法を指定します。スナップショットの独立性レベルをテーブルヒントとして指定することはできません。

テーブルヒント	説明
HOLDLOCK	独立性レベル 3 と同等な動作を設定します。このテーブルヒントは SERIALIZABLE と同義です。
NOLOCK	独立性レベル 0 と同等な動作を設定します。このテーブルヒントは READUNCOMMITTED と同義です。
READCOMMITTED	独立性レベル 1 と同等な動作を設定します。
READPAST	書き込みロックがかけられたローをブロックするのではなく、無視するようにデータベースサーバに指示します。このテーブルヒントは独立性レベル 1 でのみ使用できます。READPAST ヒントは、FROM 句の中の相関名がベーステーブルまたは共有されたグローバルテンポラリテーブルを参照する場合のみ考慮されます。それ以外の場合 (ビュー、プロキシテーブル、およびテーブル関数)、READPAST ヒントは無視されます。ベーステーブルの相関名にヒントを指定すると、ビュー内のクエリから READPAST を使用する場合があります。READPAST テーブルヒントを使用すると、サーバ内でロックと述部評価が相互に影響し合うため、異常事態が発生する可能性があります。また、DELETE 文、INSERT 文、または UPDATE 文のターゲットとなるテーブルに READPAST ヒントは使用できません。
READUNCOMMITTED	独立性レベル 0 と同等な動作を設定します。このテーブルヒントは NOLOCK と同義です。
REPEATABLEREAD	独立性レベル 2 と同等な動作を設定します。
SERIALIZABLE	独立性レベル 3 と同等な動作を設定します。このテーブルヒントは HOLDLOCK と同義です。
UPDLOCK	ヒントが指定されたテーブルの文によって処理されるローを意図的ロックを使用してロックすることを指定します。影響を受けるローは、トランザクションの終わりまでロックされたままになります。UPDLOCK はすべての独立性レベルで機能し、意図的ロックを使用します。
XLOCK	ヒントが指定されたテーブルの文によって処理されるローを排他的にロックすることを指定します。影響を受けるローは、トランザクションの終わりまでロックされたままになります。XLOCK はすべての独立性レベルで機能し、書き込みロックを使用します。

注記

Mobile Link 同期に参加しているデータベースにクエリを書き込んでいる場合、同期スクリプトに READPAST テーブルヒントを使用しないことをお奨めします。

アプリケーションの更新回数が多すぎてダウンロードのパフォーマンスに影響が出ているために READPAST を考慮する場合、代替策としてスナップショットアイソレーションを使用する方法があります。

最適化テーブルヒント (FASTFIRSTROW)

FASTFIRSTROW テーブルヒントを使用すると、optimization_goal オプションを First-row に設定することなく、クエリの最適化ゴールを設定できます。FASTFIRSTROW を使用すると、データベースサーバは、クエリ結果の最初のローをフェッチする時間を短縮するためのアクセスプランを選択します。

WITH (index-hint) 句

WITH (index-hint) 句を使用すると、クエリオプティマイザプラン選択アルゴリズムを無効にするインデックスヒントを指定できます。また、インデックスを使用してテーブルにアクセスする方法をオプティマイザに正確に指示できます。インデックスヒントは、ベーステーブル、テンポラリテーブル、およびマテリアライズドビューに使用できます。

NO INDEX

この句は、強制的にテーブルを逐次スキャンするときに使用します (インデックスは不使用)。逐次スキャンのコストは非常に高くなる場合があります。

INDEX ([PRIMARY KEY | FOREIGN KEY] index-name [,...])

この句は、オプティマイザがクエリを処理するために使用する必要のあるインデックスを、最大 4 つまで指定するときに使用します。

指定したインデックスが 1 つでも使用できなかった場合はエラーが返されます。

PRIMARY KEY または FOREIGN KEY を指定して、テーブルの PRIMARY KEY インデックスと FOREIGN KEY インデックスの名前が同じ場合のあいまいさを排除できます。

PRIMARY キーまたは FOREIGN キーを指定しないでインデックスヒントにインデックス名を指定すると、テーブルに同じ名前を持つ複数のインデックスが存在する場合、オプティマイザは標準のインデックスを選択します。標準のインデックスが存在しない場合、オプティマイザはプライマリキーインデックスを選択します。プライマリキーインデックスが存在しない場合は、外部キーインデックスが代わりに使用されます。

index-name は、インデックスのユーザ ID とテーブル名を指定することで修飾できます。

INDEX 句で指定するインデックスは、そのテーブルに定義されているインデックスにする必要があります。それ以外の場合、エラーが返されます。たとえば、FROM Products WITH(INDEX (Products.xx)) は、インデックス xx が Products テーブルに定義されていない場合、エラーを返します。同様に、FROM Products WITH(INDEX (sales_order_items.sales_order_items)) は、sales_order_items.sales_order_items インデックスが存在していても、Products テーブルに定義されていないため、エラーを返します。

INDEX ONLY { ON | OFF }

この句は、データのインデックス専用取得を実行するかどうかを制御するときに使用します。INDEX ONLY ON を使用して INDEX (index-name...) 句を指定すると、データベースサーバは、指定されたインデックスを使用してインデックス専用取得を実行しようとします。インデックス専用取得の処理で、指定したインデックスが 1 つでも使用できなかった場合は、エラーが返されます (たとえば、インデックスがない場合、または既存のインデックスがクエリを処理できない場合)。

INDEX ONLY OFF を指定すると、インデックス専用取得は行われません。

FORCE INDEX (index-name)

FORCE INDEX (index-name) 構文は互換性のために用意されています。また、複数インデックスの指定はサポートされません。この句は、WITH (INDEX (index-name)) と同じです。

CLUSTERED INDEX

この句は、オプティマイザでクラスタドインデックスを使用する必要があること (存在する場合) を指定するときに使用します。ベーステーブル用に存在できるクラスタドインデックスは 1 つのみのため、インデックス名は指定しません。クラスタドインデックスが存在しないか使用できない場合は、エラーが返されます。

備考

サブクエリと subselect は、FROM 句でのストアドプロシージャおよびテーブル関数の引数としてサポートされます。たとえば、次の FROM 句は有効です。

```
SELECT *, ( SELECT 12 x ) D
FROM sa_rowgenerator( 1,( SELECT 12 x ) );
```

SELECT 文、UPDATE 文、DELETE 文には、文が使用するテーブルを指定するテーブルリストが必要です。

> **注記**
>
> FROM 句の説明はテーブルについてのものですが、特に注意書きがなければビューと派生テーブルにも適用します。

FROM 句は、指定した全テーブルのすべてのカラムで構成される結果セットを作成します。最初に、コンポーネントテーブルのすべてのローの組み合わせが結果セットの中に入ります。次に、JOIN 条件か WHERE 条件、またはその両方の分だけ、通常は組み合わせの数が減ります。

CROSS JOIN には ON フレーズを使用できません。

権限

`openstring-expression` の FILE 句では、READ FILE 権限が必要です。

`openstring-expression` の TABLE 句では、ユーザが参照先のテーブルを所有しているか、SELECT ANY TABLE 権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

FROM 句は、ANSI/ISO SQL 標準の基礎部分です。FROM 句の複雑さは、FROM 句の個々のコンポーネントを標準の該当部分に照らしてチェックする必要があることを意味します。次に、ソフトウェアでサポートされている、オプションの ANSI/ISO SQL 言語機能の一部のリストを示します。

- CROSS JOIN、FULL OUTER JOIN、NATURAL JOIN は、オプションの ANSI/ISO SQL 機能 F401 を構成します。
- INTERSECT と INTERSECT ALL は、オプションの ANSI/ISO SQL 機能 F302 を構成します。
- EXCEPT ALL はオプションの ANSI/ISO SQL 言語機能 F304 です。
- 派生テーブルは、ANSI/ISO SQL 言語機能 T591 です。

- FROM 句のプロシージャ (テーブル関数) は、ANSI/ISO SQL 機能 T326 です。ANSI/ISO SQL 標準では TABLE キーワードを使用してプロシージャの出力をテーブル式として識別する必要があるのに対し、ソフトウェアでは TABLE キーワードは不要です。
- 共通テーブル式は、オプションの ANSI/ISO SQL 言語機能 T121 です。共通テーブル式の中でネストされた派生テーブルで別の共通テーブル式を使用する方法は、言語機能 T122 です。
- 再帰テーブル式は、ANSI/ISO SQL 機能 T131 です。共通テーブル式の中でネストされた派生テーブルでの、再帰テーブル式の使用は、オプションの ANSI/ISO SQL 言語機能 T132 です。

次に示す FROM 句のコンポーネントは、標準にありません。

- KEY JOIN。
- CROSS APPLY と OUTER APPLY。
- OPENSTRING。
- CONTAINS を使用した `table-expression` (全文検索)。
- `dml-statement` を派生テーブルとして指定。
- すべてのテーブルヒント (WITH、FORCE INDEX、READPAST、独立性レベルのヒントの使用を含む)。
- LATERAL (`table-expression`)。LATERAL (`select-statement`) は、ANSI/ISO SQL 標準のオプションの ANSI/ISO SQL 言語機能 T491 です。

例

次は、有効な FROM 句です。

```
...
FROM GROUPO.Employees
...
```

```
...
FROM GROUPO.Employees NATURAL JOIN GROUPO.Departments
...
```

```
...
FROM GROUPO.Customers
KEY JOIN GROUPO.SalesOrders
KEY JOIN GROUPO.SalesOrderItems
KEY JOIN GROUPO.Products
...
```

```
...
FROM GROUPO.Employees CONTAINS ( Street, ' Way ' )
...
```

次のクエリは、クエリ内での派生テーブルの使い方を示します。

```
SELECT Surname, GivenName, number_of_orders
FROM GROUPO.Customers JOIN
     ( SELECT CustomerID, COUNT(*)
       FROM GROUPO.SalesOrders
        GROUP BY CustomerID )
     AS sales_order_counts( CustomerID,
                            number_of_orders )
ON ( Customers.ID = sales_order_counts.CustomerID )
WHERE number_of_orders > 3;
```

次のクエリは、ストアドプロシージャの結果セットからローを選択する方法を示します。

```
SELECT t.ID, t.QuantityOrdered AS q, p.name
FROM GROUPO.ShowCustomerProducts( 149 ) t JOIN GROUPO.Products p
ON t.ID = p.ID;
```

次の例は、ファイルを問い合わせる OPENSTRING 句を使用して、クエリを実行する方法を示します。CREATE TABLE 文は、2 つのカラム column1 と columns2 を持つテーブル testtable を作成します。UNLOAD 文は、RowGenerator テーブルからローをアンロードして、ファイル `testfile.dat` を作成します。SELECT 文では FROM 句の中で OPENSTRING 句を指定し、testtable テーブルと RowGenerator テーブルの両方のスキーマ情報を使用して `testfile.dat` をクエリします。このクエリは値が 49 のローを 1 つ返します。

```
CREATE TABLE testtable( column1 CHAR(10), column2 INT );
UNLOAD SELECT * FROM RowGenerator TO 'testfile.dat';
SELECT A.column2
  FROM OPENSTRING( FILE 'testfile.dat' )
  WITH ( TABLE testtable( column2 ) ) A, RowGenerator B
  WHERE A.column2 = B.row_num
  AND A.column2 < 50
  AND B.row_num > 48;
```

次の例は、文字列値を問い合わせる OPENSTRING 句を使用して、クエリを実行する方法を示します。SELECT 文では FROM 句の中で OPENSTRING 句を使用し、WITH 句で与えられるスキーマ情報を使用して文字列値を問い合わせます。クエリは、3 つのローを持つ 2 つのカラムを返します。

```
SELECT *
  FROM OPENSTRING( VALUE '1,"First"$2,"Second"$3,"Third"')
  WITH (c1 INT, c2 VARCHAR(30))
  OPTION ( DELIMITED BY ',' ROW DELIMITED BY '$')
  AS VALS
```

次の例は、データ操作文によって修正されたローを選択するためのクエリを実行する方法を示します。この例では、青色の品目の在庫が半分以上減ると、警告が発行されます。

```
SELECT old_products.name, old_products.quantity, final_products.quantity
FROM
( UPDATE GROUPO.Products SET quantity = quantity - 10 WHERE color = 'Blue' )
REFERENCING ( OLD AS old_products FINAL AS final_products )
WHERE final_products.quantity < 0.5 * old_products.quantity;
```

次のクエリは、ShowCustomerProducts という架空のプロシージャからの選択で WITH 句を使用する方法を示します。

```
SELECT sp.ident, sp.quantity, Products.name
FROM GROUPO.ShowCustomerProducts( 149 )
   WITH ( ident INT, description CHAR(20), quantity INT ) sp
JOIN GROUPO.Products
ON sp.ident = Products.ID;
```

関連情報

%TYPE および %ROWTYPE 属性 [110 ページ]
PIVOT 句 [1232 ページ]
UNPIVOT 句 [1382 ページ]

名前付きパラメータ [123 ページ]
CONTAINS 探索条件 [72 ページ]
UNNEST 配列演算子 [26 ページ]
OPENXML 演算子 [19 ページ]
LOAD TABLE 文 [1182 ページ]
DELETE 文 [1016 ページ]
SELECT 文 [1291 ページ]
UPDATE 文 [1391 ページ]
INSERT 文 [1167 ページ]
MERGE 文 [1204 ページ]

1.4.4.174 GET DATA 文 [ESQL]

カーソルの現在のローの 1 つのカラムに対する文字列またはバイナリデータを取得します。

構文

```
GET DATA cursor-name
COLUMN column-num
OFFSET start-offset
[ WITH TEXTPTR ]
{ USING DESCRIPTOR sqlda-name | INTO hostvar, ... }
```

```
cursor-name : identifier | hostvar
```

```
column-num : integer | hostvar
```

```
start-offset : integer | hostvar
```

```
sqlda-name : identifier
```

パラメータ

COLUMN 句

column-num の値は 1 から始まり、どのカラムのデータがフェッチされるかを示します。そのカラムは、文字列型またはバイナリ型で指定します。

OFFSET 句

start-offset は、フィールド値の中でスキップされるバイト数を示します。通常、これは、以前にフェッチされたバイトの数です。この GET DATA 文に対してフェッチされるバイト数は、ターゲットのホスト変数の長さによって決定されます。

WITH TEXTPTR 句

WITH TEXTPTR 句がある場合、テキストポインタが SQLDA の第 2 ホスト変数または SQLDA の第 2 フィールドに取り出されます。このテキストポインタは、Transact-SQL READ TEXT と WRITE TEXT 文と一緒に使えます。テキストポインタは 16 ビットのバイナリ値で、次のように宣言できます。

```
DECL_BINARY( 16 ) textptr_var;
```

WITH TEXTPTR 句は、長いデータ型 (LONG BINARY、LONG VARCHAR、TEXT、IMAGE) でのみ使用できます。それ以外のデータ型で使用すると、エラー INVALID_TEXTPTR_VALUE が返されます。

データの合計の長さが、SQLCA 構造体の SQLCOUNT フィールドに返されます。

USING DESCRIPTOR 句

sqlda-name では、フェッチされたデータを受信する SQLDA (SQL Descriptor Area) を指定します。USING DESCRIPTOR 句を使用すると、フェッチされたデータを受信するためのホスト変数を動的に指定できます。

INTO 句

INTO 句は、フェッチされたデータを受信するホスト変数を指定するときに使用します。ターゲットのホスト変数のインジケータ値は a_sql_len 型であり、現在は 16 ビット値です。このため、必ずしもトランケートされたバイト数が入るほど大きいわけではありません。代わりに、フィールドに NULL 値がある場合は負の値が入り、値がトランケートされた場合は (トランケートされたバイトの数とはかぎりません) 正の値が入ります。また、NULL 値以外の値がトランケートされない場合は、0 が入ります。

また、LONG VARCHAR、LONG NVARCHAR、または LONG BINARY ホスト変数で 0 より大きいオフセットを使用すると、untrunc_len フィールドにはトランケート前のサイズが正確に示されません。

備考

現在のカーソルの位置でローから 1 つのカラム値を取得します。カーソルを開き、FETCH を使ってローに配置しておいてください。

GET DATA を使って、LONG BINARY または LONG VARCHAR フィールドをフェッチします。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、GET DATA を使用してバイナリの大規模なオブジェクト (BLOB とも呼ばれます) をフェッチします。

```
EXEC SQL BEGIN DECLARE SECTION;
DECL_BINARY(1000) piece;
short ind;
EXEC SQL END DECLARE SECTION;
int size;
/* Open a cursor on a long varchar field */
EXEC SQL DECLARE big_cursor CURSOR FOR
SELECT long_data FROM some_table
WHERE key_id = 2;
EXEC SQL OPEN big_cursor;
EXEC SQL FETCH big_cursor INTO :piece;
for( offset = 0; ; offset += piece.len ) {
   EXEC SQL GET DATA big_cursor COLUMN 1
   OFFSET :offset INTO :piece:ind;
   /* Done if the NULL value */
   if( ind < 0 ) break;
   write_out_piece( piece );
   /* Done when the piece was not truncated */
   if( ind == 0 ) break;
}
EXEC SQL CLOSE big_cursor;
```

関連情報

FETCH 文 [ESQL] [SP] [1101 ページ]
READTEXT 文 [T-SQL] [1250 ページ]
SET 文 [1325 ページ]

1.4.4.175 GET DESCRIPTOR 文 [ESQL]

記述子領域内の変数に関する情報を取り出すか、その値を取り出します。

構文

```
GET DESCRIPTOR descriptor-name
{ hostvar = COUNT | VALUE { integer | hostvar } assignment, ... }
```

```
assignment :
 hostvar =
TYPE
```

```
| LENGTH
| PRECISION
| SCALE
| DATA
| INDICATOR
| NAME
| NULLABLE
| RETURNED_LENGTH
```

```
descriptor-name : identifier
```

備考

GET DESCRIPTOR 文は、記述子領域内の変数に関する情報を取り出すか、その値を取り出すために使用します。

{ integer | hostvar } 値には、情報を取り出す記述子領域内の変数を指定します。GET...DATA の実行中に型チェックが実行され、ホスト変数と記述子変数のデータ型が同じかどうか確認されます。LONG VARCHAR と LONG BINARY は GET DESCRIPTOR...DATA ではサポートされていません。

エラーが発生すると、戻り値は SQLCA に格納されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

GET DESCRIPTOR は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部を構成します。

例

次の例は、sqlda の位置 col_num にあるカラムの型を返します。

```
int get_type( SQLDA *sqlda, int col_num )
{
    EXEC SQL BEGIN DECLARE SECTION;
    int ret_type;
```

```
    int col = col_num;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL GET DESCRIPTOR sqlda VALUE :col :ret_type = TYPE;
    return( ret_type );
}
```

関連情報

ALLOCATE DESCRIPTOR 文 [ESQL] [628 ページ]
DEALLOCATE DESCRIPTOR 文 [ESQL] [999 ページ]
SET DESCRIPTOR 文 [ESQL] [1303 ページ]

1.4.4.176 GET OPTION 文 [ESQL]

オプションの現在の設定を取得します。この文の代わりに、CONNECTION_PROPERTY 関数を使用することをお奨めします。

構文

```
GET OPTION [ userid.]option-name
{ INTO hostvar | USING DESCRIPTOR sqlda-name }
```

```
userid : identifier, string | hostvar
```

```
option-name :
identifier
| string
| hostvar
```

```
hostvar: 許可されたインジケータ変数
```

```
sqlda-name : identifier
```

備考

GET OPTION 文は、このソフトウェアの旧バージョンとの互換性のために用意されています。オプションの値を取得するには、CONNECTION_PROPERTY システム関数の使用をお奨めします。

GET OPTION 文は、option-name のオプション設定を取得します。このオプション設定は userid に対するものか、または userid を指定していない場合は接続しているユーザに対するものです。これはユーザ個人の設定であるか、または、接続しているユーザに設定がない場合は PUBLIC 設定となります。指定したオプションがデータベースオプションであり、ユーザがそのオプションに対してテンポラリ設定を持っている場合、テンポラリ設定が取得されます。

option-name が存在しない場合、GET OPTION 文は警告 SQLE_NOTFOUND を返します。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、GET OPTION の使い方を示します。

```
EXEC SQL GET OPTION 'date_format' INTO :datefmt;
```

関連情報

システムプロシージャのアルファベット順リスト [1438 ページ]
SET OPTION 文 [1310 ページ]
CONNECTION_PROPERTY 関数 [システム] [281 ページ]

1.4.4.177 GOTO 文

ラベルの付いた文に制御を分岐します。

構文

```
label-name:
sql-statement(s)
GOTO label-name
```

備考

プロシージャ、トリガ、またはバッチ内の文には、有効な識別子とそれに続くコロンを使用して、ラベルを付けることができます (mylabel: など)。ラベルは、ループ、条件、またはブロックの先頭に付けます。ラベルは、GOTO 文内で参照することができます。その結果、実行ポイントがループ/コンディションの一番上またはブロック内の最初の文に移動します。

GOTO 文内でラベルを参照する場合は、コロンを指定しないでください。

複合文をネストする場合、現在の複合文およびその上位の複合文内のラベルに進むだけです。上位の複合文内でネストされている他の複合文にあるラベルに進むことはできません。

データベースサーバでは、Transact-SQL プロシージャ、トリガ、またはバッチでの GOTO 文の使用をサポートしています。Transact-SQL では、ラベルの使用はループ、条件、またはブロックの先頭に制限されておらず、任意の文で実行できます。ただし、ネストされた複合文内の GOTO 文の使用には、同じ制限が適用されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、GotoTest プロシージャが実行されると、GOTO lbl1 によって実行が SET i2 = 200 文に再配置されます。その結果、カラム i2 の戻り値は、結果セット内の 5 すべてのローで 203 になります。

```
CREATE OR REPLACE PROCEDURE GotoTest()
RESULT ( id INT, i INT, i2 INT )
BEGIN
    DECLARE LOCAL TEMPORARY TABLE gotoTable( id INT DEFAULT AUTOINCREMENT, i INT,
i2 INT ) NOT TRANSACTIONAL;
    DECLARE i INT;
    DECLARE i2 INT;
    SET i = 100;
    lbl1: WHILE i < 105 LOOP
        SET i2 = 200;
        SET i2 = i2 + 1;
        lbl2: BEGIN
            SET i2 = i2 + 1;
            lbl3: BEGIN
```

```
                SET i2 = i2 + 1;
                INSERT INTO gotoTable(i, i2) VALUES(i, i2);
                SET i = i + 1;
                IF( i < 110 ) THEN
                    GOTO lbl1
                END IF;
            END
        END
    END LOOP;
    SELECT id, i, i2 FROM gotoTable ORDER BY id;
END;
CALL GotoTest();
```

表 4: 結果

id	i	i2
1	100	203
2	101	203
3	102	203
4	103	203
5	104	203

GotoTest プロシージャが GOTO lbl1 の代わりに GOTO lbl2 を使用するよう変更された場合、GOTO 文により実行が lbl2: BEGIN 文の後ただちに SET i2 = i2 + 1 文に再配置され、カラム i2 の戻り値は、203、205、207 となり、221 まで続きます。

GotoTest プロシージャが GOTO lbl3 を使用するよう変更された場合、GOTO 文により実行が lbl3: BEGIN 文の後ただちに SET i2 = i2 + 1 文に再配置され、カラム i2 の戻り値は、203、204、205 となり、212 まで続きます。

次の Transact-SQL バッチは、データベースサーバメッセージウィンドウに 4 回 "yes" のメッセージを表示します。

```
DECLARE @count SMALLINT
SELECT @count = 1
restart:
   PRINT 'yes'
   SELECT @count = @count + 1
   WHILE @count <=4
    GOTO restart;
```

1.4.4.178 GRANT 文

システム権限とオブジェクトレベル権限をユーザとロールに付与します。

構文

システム権限の付与

```
GRANT system-privilege [,...]
TO grantee [ ,... ]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

オブジェクトレベル権限の付与

```
GRANT object-level-privilege,...
 ON [ owner.]object-name
 TO to-userid,...
[ WITH GRANT OPTION ]
```

```
object-level-privilege :
ALL [ PRIVILEGES ]
| ALTER
| DELETE
| INSERT
| LOAD
| REFERENCES [ ( column-name,... ) ]
| SELECT [ ( column-name,... ) ]
| TRUNCATE
| UPDATE [ ( column-name,... ) ]
```

SET USER システム権限の付与

```
GRANT SET USER [ ( user-list | ANY [ WITH ROLES role-list ] ) ]
TO grantee [,...]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

CHANGE PASSWORD システム権限の付与

```
GRANT CHANGE PASSWORD [ ( user-list | ANY [ WITH ROLES role-list ] ) ]
TO grantee [,...]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

パラメータ

grantee

ユーザのユーザ ID またはロールの名前。権限を互換ロールに付与することはできません。権限をどのシステムロールに付与することもできますが、ログインをサポートするシステムロールは PUBLIC、dbo、diagnostics、rs_systabgroup、SA_DEBUG だけです。

object-level-privilege

ALL privilege

この権限は、ALTER、DELETE、INSERT、REFERENCES、SELECT、PDATE の各権限をテーブルに付与します。この権限は、DELETE、INSERT、UPDATE の各権限をビューに付与します。

ALTER privilege

この権限を使用すると、ユーザが ALTER TABLE 文を使用して指定のテーブルを変更できます。この権限は、ビューには使えません。

DELETE privilege

この権限を使用すると、ユーザは指定したテーブルまたはビューからローを削除できます。

INSERT privilege

この権限を使用すると、ユーザは指定したテーブルまたはビューにローを挿入できます。

LOAD privilege

この権限を使用すると、ユーザは名前付きテーブルまたはビューをロードできます。

REFERENCES privilege

この権限を使用すると、ユーザは指定したテーブルのインデックスを作成できます。また、指定したテーブルを参照する外部キーを作成できます。カラム名を指定する場合、ユーザはそれらのカラムのみを参照できます。カラムの REFERENCES 権限は、ビューに対しては付与されません。テーブルに対してのみ付与されます。INDEX は REFERENCES の同意語です。

SELECT privilege

この権限を使用すると、ユーザはビューまたはテーブル内の情報を参照できます。カラム名を指定する場合、ユーザはそれらのカラムのみを参照できます。カラムの SELECT 権限は、ビューに対しては付与されません。テーブルに対してのみ付与されます。

TRUNCATE privilege

この権限を使用すると、ユーザは名前付きオブジェクトをトランケートできます。

UPDATE privilege

この権限を使用すると、ユーザはビューまたはテーブル内のローを更新できます。カラム名を指定する場合、ユーザはそれらのカラムのみを更新できます。

WITH GRANT OPTION

WITH GRANT OPTION を指定する場合、指定した名前のユーザ ID にも権限が与えられ、他のユーザ ID に同じ権限を GRANT (付与) できます。ロールを行使できるユーザは、ロールに付与されている WITH GRANT OPTION を継承しません。

FROM internal-id

この句は内部でのみ使用されます。

GRANT SET USER

user-list

`grantee-list` 内のユーザによって同一化できるすべてのユーザ ID (ターゲット) のカンマ区切りのリストを指定します。例:`GRANT SET USER(u1, u2, u3)...`

ANY [WITH ROLES target_role_list] clause

`grantee` によって特定のユーザ ID を指定せずに同一化できるユーザを指定します。

ANY が指定されている場合、ユーザは任意のユーザを同一化できます。これがデフォルト値です。

ANY WITH ROLES `role-list` を指定すると、`grantee-list` 内のユーザは `role-list` にリストされているロールを 1 つ以上持っているユーザを同一化できます (`role-list` はカンマ区切りのロールのリスト)。

WITH ADMIN [ONLY] OPTION option

WITH ADMIN OPTION 句と WITH ADMIN ONLY OPTION 句を指定できるのは、ANY 句を使用する場合だけです。

GRANT CHANGE PASSWORD

user-list

`grantee` によってパスワードを変更できるユーザのカンマ区切りのリストを指定します。

ANY [WITH ROLES role-list]

`grantee` によって特定のユーザ ID を指定せずにパスワードを変更できるユーザを指定します。

ANY が指定されている場合、ユーザは任意のユーザのパスワードを変更できます。これがデフォルト値です。

ANY WITH ROLES `role-list` を指定すると、`grantee` は `role-list` にリストされているロールを 1 つ以上持っている任意のユーザのパスワードを変更できます (`role-list` はカンマ区切りのロールのリスト)。

WITH ADMIN [ONLY] OPTION option

WITH ADMIN OPTION 句と WITH ADMIN ONLY OPTION 句を指定できるのは、ANY 句を使用する場合だけです。

備考

無効にされたオブジェクトに対する権限を付与できます。無効化されたオブジェクトに対する権限はデータベースに保存され、オブジェクトが有効になると使用できるようになります。

SYS ロールという例外はありますが、can grant/revoke 追加の権限をシステムロールに追加したり、システムロールから取り消したりできます。ただし、追加や取り消しを実行する権限に対する管理権限がある場合に限ります。

SET USER を特定のユーザに 2 回以上付与し、同一化できる異なるユーザ ID を指定すると、(以前の付与情報が上書きされるのではなく) 同一化できるリストに追加のユーザが付加されます。

同一化の権限を付与すること (GRANT SET USER) は、ユーザが別のユーザを正しく同一化できるかどうかを示すものではありません。ユーザが別のユーザを同一化できるかどうかの評価は、SETUSER 文を実行することにより、ユーザ ID が別のユーザの同一化を開始しようとするときに行われます。同一化するユーザは、SET USER システム権限を持っていて、同一化に必要な最低の基準を満たしている必要があります。

バージョン 16.0 以前のソフトウェアで使用された権限、パーミッション、グループに関連する GRANT 構文は、引き続きサポートされますが推奨されません。

MANAGE ROLES システム権限にシステム権限を付与する場合、MANAGE ROLES、つまり SYS_MANAGE_ROLES_ROLE (`GRANT privilege-name TO SYS_MANAGE_ROLES_ROLE;` など) に対する特別な内部表現を使用する必要があります。これは、

権限

付与する権限ごとに管理権限を持っている必要があります。

オブジェクトレベルの権限を付与する場合は、MANAGE ANY OBJECT PRIVILEGE システム権限 (管理権限付き) も持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

ロールへのシステム権限の付与

RoleA に管理権限を与えずに、CREATE ANY OBJECT システム権限をロール RoleA に付与するには、次の文を実行します。

```
GRANT CREATE ANY OBJECT TO RoleA;
```

他のユーザやロールに対してシステム権限の付与や取り消しを実行する機能とともに、CREATE ANY OBJECT システム権限を RoleA に付与するには、次の文を実行します。

```
GRANT CREATE ANY OBJECT TO RoleA WITH ADMIN OPTION;
```

RoleA 管理権限を BACKUP DATABASE システム権限に与えるが、BACKUP DATABASE 権限を使えないようにするには、次の分を実行します。

```
GRANT BACKUP DATABASE TO RoleA WITH ADMIN ONLY OPTION;
```

ユーザへの SET USER の付与

次の例は、User4 および User5 が User1、User2、User3 を同一化できることを指定します。

```
GRANT SET USER ( User1, User2, User3 ) TO User4, User5;
```

次の例は、User1 がデータベース内のすべてのユーザを同一化できることを指定します。同様に、User1 は SET USER システム権限を他のユーザに付与できます。

```
GRANT SET USER (ANY) TO User1 WITH ADMIN OPTION;
```

次の例は、User1 が、SYS_AUTH_BACKUP_ROLE 互換ロールを付与されているすべてのユーザを同一化できることを指定します。

```
GRANT SET USER (ANY WITH ROLES SYS_AUTH_BACKUP_ROLE) TO User1;
```

関連情報

SETUSER 文 [1329 ページ]
REVOKE 文 [1278 ページ]

1.4.4.179 GRANT ROLE 文

ユーザおよびロールにロールを付与します。

構文

システムロールの付与

```
GRANT ROLE system-role
TO grantee [ ,... ]
```

```
system-role :
dbo
| DIAGNOSTICS
| PUBLIC
| rs_systabgroup
| SA_DEBUG
| SYS
| SYS_REPLICATION_ADMIN_ROLE
| SYS_RUN_REPLICATION_ROLE
| SYS_SAMONITOR_ADMIN_ROLE
| SYS_SPATIAL_ADMIN_ROLE
```

ユーザ定義ロールの付与

```
GRANT ROLE user-defined-role [,...]
TO grantee [ ,... ]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

互換ロールの付与

```
GRANT ROLE compatibility-role-name [,...]
TO grantee [ ,... ]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
[ WITH NO SYSTEM PRIVILEGE INHERITANCE ]
```

```
compatibility-role-name :
SYS_AUTH_BACKUP_ROLE
| SYS_AUTH_DBA_ROLE
| SYS_AUTH_PROFILE_ROLE
| SYS_AUTH_READCLIENTFILE_ROLE
| SYS_AUTH_READFILE_ROLE
| SYS_AUTH_RESOURCE_ROLE
| SYS_AUTH_SA_ROLE
| SYS_AUTH_SSO_ROLE
| SYS_AUTH_VALIDATE_ROLE
| SYS_AUTH_WRITECLIENTFILE_ROLE
| SYS_AUTH_WRITEFILE_ROLE
```

パラメータ

role-name

システムロール、互換ロール、ユーザ拡張ロール、またはユーザ定義ロールの名前。

grantee

ユーザのユーザ ID またはロールの名前。ロールを互換ロールに付与することはできません。ロールをどのシステムロールに付与することもできますが、ログインをサポートするシステムロールは PUBLIC、dbo、diagnostics、rs_systabgroup、SA_DEBUG だけです。

WITH [NO] ADMIN OPTION clause

この句は、非システム権限を付与する場合にのみ適用できます。システムロールの管理権限を付与することはできません。システムロールを管理 (付与および取り消し) できるのは、MANAGE ROLES システム権限を持つユーザだけです。

デフォルトは WITH NO ADMIN OPTION です。つまり、`grantee` にはロールが付与されますが、それらを管理できる機能は付与されません。

WITH ADMIN OPTION を指定した場合、各 `grantee` には各 `role-granted` に対する管理者権限が付与されます。

WITH ADMIN ONLY OPTION を指定した場合、`grantee` にはロールに対する管理権限だけが付与され、ロールそのものは付与されません。WITH NO SYSTEM PRIVILEGE INHERITANCE 句を WITH ADMIN ONLY OPTION と一緒に使用することはできません。

WITH ADMIN OPTION 句を WITH NO SYSTEM PRIVILEGE INHERITANCE 句と一緒に使用できるのは、SYS_AUTH_DBA_ROLE を付与する場合に限られます。

WITH NO SYSTEM PRIVILEGE INHERITANCE

この句は、ロールの被付与者が、そのロールのシステム権限を継承するのを回避します。通常は、互換ロールをユーザまたはロールに付与する場合は、そのロールとその被付与者の両方が、その互換ロールのシステム権限を使用できます。互換ロールのシステム権限の継承を無効にした場合は、被付与者でなく、ロールのみがシステム権限を使用できます。

WITH NO SYSTEM PRIVILEGE INHERITANCE 句は下位互換性を確保するために用意されています。互換ロールに対するシステム権限の継承を無効にすると、バージョン 12 以前のデータベースの継承不可能な権限の動作と同様になります。互換ロールに対してシステム権限の継承を有効にすると、すべてのシステムロールおよびユーザ定義ロールの動作と同様になります。

ユーザ、ユーザ拡張ロール、またはシステムロールに対して次のいずれかのロールを付与するときは、システム権限の継承を無効にすることができます。

- SYS_AUTH_DBA_ROLE
- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE
- SYS_RUN_REPLICATION_ROLE

同様に、WITH ADMIN OPTION 句を WITH NO SYSTEM PRIVILEGE INHERITANCE 句と一緒に使用できるのは、SYS_AUTH_DBA_ROLE を付与する場合に限られます。WITH NO SYSTEM PRIVILEGE INHERITANCE 句を WITH ADMIN ONLY OPTION と一緒に使用することはできません。

ユーザのシステム権限の継承を無効にすることは、そのユーザをユーザ拡張ロールに変換しようとする場合にのみ有用です。

備考

SYS ロールという例外はありますが、can grant/revoke 追加の権限をシステムロールに追加したり、システムロールから取り消したりできます。ただし、追加や取り消しを実行するロールに対する管理権限がある場合に限ります。

MANAGE ROLES システム権限にロールを付与するとき、SYS_MANAGE_ROLES_ROLE という特別な内部表現を使用する必要があります。例: `GRANT ROLE role-name TO SYS_MANAGE_ROLES_ROLE;`

バージョン 16.0 以前のソフトウェアで使用された権限、パーミッション、グループに関連する GRANT 構文は、引き続きサポートされますが推奨されません。

権限

付与するロールごとに管理権限を持っている必要があります。

SYS_REPLICATION_ADMIN_ROLE システムロールを付与する場合は、MANAGE ROLES システム権限が必要です。

SYS_REPLICATION_RUN_ROLE システムロールを付与する場合は、SYS_REPLICATION_ADMIN_ROLE システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

ユーザへのロールの付与

ユーザ Bob に管理権限なしの SecurityRole というロールを付与するには、次の文を実行します。

```
GRANT ROLE SecurityRole TO Bob;
```

ロール RoleB に RoleA に関連するすべての権限を付与し、RoleA の管理権限は付与しない場合は、次の文を実行します。

```
GRANT ROLE RoleA TO RoleB;
```

RoleB を管理権限とともにユーザ Jane に付与するには、次の文を実行します。

```
GRANT ROLE RoleB TO Jane WITH ADMIN OPTION;
```

ユーザ John に RoleB に対する管理権限を付与するが、RoleB を使用できないようにするには、次の文を実行します。

```
GRANT ROLE RoleB TO John WITH ADMIN ONLY OPTION;
```

SYS_RUN_REPLICATION_ROLE システムロールの付与

次の例は、被付与者 Sam_Singer に SYS_RUN_REPLICATION_ROLE システムロールを付与します。

```
GRANT ROLE SYS_RUN_REPLICATION_ROLE
TO Sam_Singer;
```

SYS_REPLICATION_ADMIN_ROLE システムロールの付与

次の例は、被付与者 Sam_Singer に SYS_REPLICATION_ADMIN_ROLE ロールを付与します。

```
GRANT ROLE SYS_REPLICATION_ADMIN_ROLE
TO Sam_Singer;
```

関連情報

SETUSER 文 [1329 ページ]
REVOKE ROLE 文 [1281 ページ]

1.4.4.180 GRANT CONNECT 文

新しいユーザを作成し、ユーザが自分のパスワードを変更する場合にも使用できます。ただし、ユーザを作成する場合は、GRANT CONNECT 文ではなく CREATE USER 文を使用することをおすすめします。

構文

```
GRANT CONNECT TO userid, ...
[ IDENTIFIED BY password, ... ]
```

パラメータ

CONNECT TO clause

新しいユーザを作成します。GRANT CONNECT を使うと、どのユーザも自分のパスワードを変更できます。パスワードとして空の文字列を持つユーザを作成するには、次のように入力します。

```
GRANT CONNECT TO userid IDENTIFIED BY "";
```

パスワードのないユーザを作成するには、次のように入力します。

```
GRANT CONNECT TO userid;
```

パスワードのないユーザは、データベースに接続できません。これは、グループを作成し、他のユーザはグループユーザ ID を使用してデータベースに接続させないようにする場合に便利です。

verify_password_function オプションは、パスワード規則 (1 桁以上の長さであることなど) の実装に使用できる関数を指定します。パスワード検証機能を使用する場合、GRANT CONNECT 文に複数のユーザ ID とパスワードを指定することはできません。

備考

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワードを指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

自分のパスワードを変更するには、GRANT CONNECT を使用するか、MANAGE ANY USER 権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、新しいデータベースユーザ SQLTester を作成し、パスワードを welcome に設定します。

```
GRANT CONNECT TO SQLTester
IDENTIFIED BY welcome
```

関連情報

CREATE USER 文 [990 ページ]

1.4.4.181 GRANT CONSOLIDATE 文 [SQL Remote]

SQL Remote 階層において、現在のデータベースのすぐ上にあり、現在のデータベースからメッセージを受信するデータベースを識別します。

構文

```
GRANT CONSOLIDATE
TO userid
TYPE message-system, ...
ADDRESS address-string, ...
[ SEND { EVERY | AT } hh:mm:ss ]
```

```
message-system :
FILE | FTP | SMTP
```

```
address : string
```

パラメータ

userid

権限が付与されるユーザのユーザ ID。

message-system

SQL Remote がサポートするメッセージシステムの 1 つ。

address

指定したメッセージシステムのアドレス。

備考

SQL Remote インストール環境では、SQL Remote 階層内の現在のデータベースのすぐ上にあるデータベースに、CONSOLIDATE 権限が付与されます。GRANT CONSOLIDATE は、統合データベースを識別するためにリモートデータベースで発行されます。各データベースが保持できる CONSOLIDATE 権限を持つユーザ ID は 1 つだけです。1 つのデータベースを複数の統合データベースに対して共通のリモートデータベースとして指定できません。

統合ユーザはメッセージシステムにより識別され、統合ユーザに対するメッセージの送受信方法が識別されます。address-name には、メッセージシステムに対して有効なアドレスを、一重引用符で囲んで指定します。統合ユーザは、各リモートデータベースに対して 1 人だけです。

FILE メッセージタイプの場合、アドレスは SQLREMOTE 環境変数で指定されているディレクトリのサブフォルダです。

GRANT CONSOLIDATE 文は、統合データベースがメッセージを受信するためには必要ですが、統合データベースをサブスクライブして何らかのデータを取得するものではありません。データをサブスクライブするには、現在のデータベースにあるパブリケーションの統合ユーザ ID に対するサブスクリプションを作成します。統合データベースでデータベース抽出ユーティリテ

ィを実行すると、それに対して適切な GRANT CONSOLIDATE 文が実行された状態でリモートデータベースが作成されます。

オプションの SEND EVERY 句と SEND AT 句を使用すると、メッセージを送信する間隔を指定できます。文字列には、メッセージ送信間隔の時間 (SEND EVERY) またはメッセージを送信する時刻 (SEND AT) を指定します。SEND AT の場合、メッセージは 1 日に 1 回送信されます。

ユーザに SEND EVERY 句も SEND AT 句もない REMOTE 権限が付与されている場合、Message Agent はメッセージを処理した後、停止します。Message Agent を継続的に実行する場合は、REMOTE 権限を持つすべてのユーザに SEND AT または SEND EVERY の頻度が指定されていることを確認します。

多くのリモートデータベースでは Message Agent を定期的に実行するため、統合データベースに SEND 句が指定されていないことがあります。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザ ID Sam_Singer に CONSOLIDATE 権限を付与します。

```
GRANT CONSOLIDATE TO Sam_Singer
TYPE SMTP
ADDRESS 'Singer, Samuel';
```

関連情報

GRANT PUBLISH 文 [SQL Remote] [1147 ページ]
GRANT REMOTE 文 [SQL Remote] [1149 ページ]
REVOKE CONSOLIDATE 文 [SQL Remote] [1274 ページ]

ALTER REMOTE MESSAGE TYPE 文 [SQL Remote] [674 ページ]
CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote] [903 ページ]
CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]

1.4.4.182 GRANT CREATE 文

指定された DB 領域にデータベースオブジェクトを作成できるユーザ権限を付与します。

構文

```
GRANT CREATE ON dbspace-name
TO userid, ...
```

備考

データベースを初期化すると、1 つのデータベースファイルが格納されます。この最初のデータベースファイルをメインファイルと呼びます。すべてのデータベースオブジェクトとすべてのデータは、デフォルトでこのメインファイルに配置されます。"DB 領域" は、データ用の領域をさらに作成する追加のデータベースファイルです。1 つのデータベースは 13 個までのファイルに保管されます (メインファイル 1 つと 12 の DB 領域)。各テーブルは、そのインデックスとともに、単一のデータベースファイルに含まれている必要があります。CREATE DBSPACE という SQL コマンドで、新しいファイルをデータベースに追加できます。

権限

MANAGE ANY USER 権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.183 GRANT EXECUTE 文

プロシージャまたはユーザ定義ファンクションを実行するユーザ権限を付与します。

構文

```
GRANT EXECUTE ON [ owner.]{ procedure-name | user-defined-function }
TO userid, ...
```

備考

なし。

権限

プロシージャを所有しているか、MANAGE ANY OBJECT PRIVILEGE システム権限を持っている必要があります。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

コア機能。

例

次の例は、ユーザ SQLTester が Calculate_Report プロシージャを実行できるようにします。

```
GRANT EXECUTE ON Calculate_Report
TO SQLTester;
```

次の SQL 文は、my_procedure という名前のプロシージャを実行するために必要な権限を M_Haneef に付与します。

```
GRANT EXECUTE
ON my_procedure
TO M_Haneef;
```

関連情報

システムプロシージャ [1429 ページ]

1.4.4.184 GRANT INTEGRATED LOGIN 文

プロシージャまたはユーザ定義ファンクションを実行するユーザ権限を付与します。

構文

```
GRANT INTEGRATED LOGIN TO user-profile-name, ...
[ AS USER userid ]
```

備考

GRANT INTEGRATED LOGIN 文は、1 つまたは複数の Windows ユーザまたはグループプロファイルと既存のデータベースユーザ ID の間に明示的な統合化ログインマッピングを作成します。これにより、ローカルコンピュータにログインできたユーザは、ユーザ ID またはパスワードを指定せずにデータベースに接続できます。`user-profile-name` は、`domain¥user-name` の形式にできます。統合化ログインがマッピングされているデータベースのユーザ ID にはパスワードを指定する必要があります。

権限

MANAGE ANY USER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.185 GRANT KERBEROS LOGIN 文

1 つ以上の Kerberos プリンシパルから既存のデータベースユーザ ID にマッピングする、Kerberos の認証済みログインを作成します。

構文

```
GRANT KERBEROS LOGIN TO client-Kerberos-principal, ...
AS USER userid
```

備考

GRANT KERBEROS LOGIN 文は、1 つ以上の Kerberos プリンシパルから既存のデータベースユーザ ID にマッピングする、Kerberos の認証済みログインを作成します。このログインマッピングにより、Kerberos に正常にログインしたユーザ (有効な Kerberos TGT (Ticket Granted Ticket) を持ったユーザ) は、ユーザ ID またはパスワードを入力せずに、データベースに接続できます。

GRANT KERBEROS LOGIN 文を使用するには、次の手順に従います。

- あらかじめ Kerberos が SQL Anywhere を使用できるように設定されている必要があります。
- あらかじめ SQL Anywhere データベースが Kerberos を使用できるように設定されている必要があります。
- データベースユーザと Kerberos プリンシパルは、常に存在している必要があります。
- Kerberos ログインがマッピングされているデータベースのユーザ ID にはパスワードを指定する必要があります。
- `client-Kerberos-principal` の形式は、`user/instance@REALM` にします。この `/instance` はオプションです。領域を含む完全なプリンシパルと、インスタンスまたは領域を区別して扱う場合にのみ、異なるプリンシパルを指定する必要があります。
- プリンシパルの大文字と小文字は区別されるため、正しく指定する必要があります。大文字と小文字の違いしかない複数のプリンシパルのマッピングはサポートされていません (たとえば、jjordan@MYREALM.COM と JJordan@MYREALM.COM の両方にマッピングすることはできません)。
- Kerberos プリンシパルに明示的なマッピングがなく、Guest データベースのユーザ ID が存在し、パスワードがある場合、Kerberos プリンシパルは Guest データベースのユーザ ID (統合化ログイン用と同じ Guest データベースのユーザ ID) を使用して接続します。

権限

MANAGE ANY USER 権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の SQL 文は、架空の Kerberos ユーザ pchin に対してデータベースのログイン権限を付与します。

```
GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"
AS USER "kerberos-user";
```

1.4.4.186 GRANT PUBLISH 文 [SQL Remote]

ユーザ ID にパブリッシャ権限を付与します。パブリッシャ権限を付与するには、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

構文

```
GRANT PUBLISH TO userid
```

備考

SQL Remote インストール環境内の各データベースは、出力メッセージ内でユーザ ID により識別され、"パブリッシャ" と呼ばれます。GRANT PUBLISH 文は、これらの出力メッセージと関連付けられているパブリッシャユーザ ID を設定します。パブリッシャを変更するには、現在のパブリッシャから PUBLISH 権限を取り消して (REVOKE PUBLISH)、それらを新しいパブリッシャに付与します。

データベースパブリッシャは、次のように特別値 CURRENT PUBLISHER をクエリすることによって決定できます。

```
SELECT CURRENT PUBLISHER;
```

パブリッシャがない場合、CURRENT PUBLISHER の値は NULL になります。

CURRENT PUBLISHER 特別値を、カラムのデフォルトの設定として使用できます。CURRENT PUBLISHER カラムをテーブルのレプリケーションを行う際のプライマリキーに含めると、複数サイトでのアップデートによりプライマリキーの矛盾が発生するのを回避できるので便利です。

パブリッシャを変更するには、まず REVOKE PUBLISH 文を使って現在のパブリッシャを削除し、それから GRANT PUBLISH 文を使って新しいパブリッシャを作成します。

この文を実行すると、PUBLIC.db_publisher データベース オプションの値が変更されます。

権限

SET ANY SYSTEM OPTION システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、データベースパブリッシャをユーザ ID JohnS に設定します。

```
GRANT PUBLISH TO JohnS;
```

次の文は、JohnS からデータベースパブリッシャを取り消して IrisM に付与します。

```
REVOKE PUBLISH FROM JohnS;
GRANT PUBLISH TO IrisM;
```

関連情報

GRANT ROLE 文 [1136 ページ]
GRANT CONSOLIDATE 文 [SQL Remote] [1141 ページ]
REVOKE PUBLISH 文 [SQL Remote] [1276 ページ]
CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]
CURRENT PUBLISHER 特別値 [87 ページ]
CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote] [903 ページ]
GRANT REMOTE 文 [SQL Remote] [1149 ページ]

1.4.4.187 GRANT REMOTE 文 [SQL Remote]

SQL Remote 階層において、現在のデータベースのすぐ下にあり、現在のデータベースからメッセージを受信するデータベースを識別します。これらはリモートユーザといいます。

構文

```
GRANT REMOTE TO userid, ...
TYPE message-system, ...
ADDRESS address-string, ...
[ SEND { EVERY | AT } send-time ]
```

パラメータ

userid

権限が付与されるユーザのユーザ ID。

message-system

SQL Remote がサポートするメッセージシステムの 1 つ。次のいずれかの値を指定します。

- FILE
- FTP
- SMTP

address-string

指定したメッセージシステムに対して有効なアドレスを含む文字列。

send-time

hh:mm:ss の形式で時刻を指定した文字列。

備考

SQL Remote インストール環境では、現在のデータベースからメッセージを受信する各データベースに、REMOTE 権限が付与されている必要があります。

この唯一の例外として、SQL Remote 階層で現在のデータベースのすぐ上にあるデータベースには、CONSOLIDATE 権限が付与されている必要があります。

リモートユーザはメッセージシステムにより識別され、統合ユーザに対するメッセージの送受信方法が識別されます。address-name には、メッセージシステムに対して有効なアドレスを、一重引用符で囲んで指定します。

FILE メッセージタイプの場合、アドレスは SQLREMOTE 環境変数で指定されているディレクトリのサブフォルダです。

GRANT REMOTE 文は、リモートデータベースがメッセージを受信するのには必要ですが、リモートデータベースをサブスクライブして何らかのデータを取得するものではありません。データをサブスクライブするには、データベース抽出ユーティリティまたは CREATE SUBSCRIPTION 文を使用して、現在のデータベースにあるパブリケーションのユーザ ID に対するサブスクリプションを作成します。

オプションの SEND EVERY 句と SEND AT 句を使用すると、メッセージを送信する間隔を指定できます。文字列には、メッセージ送信間隔の時間 (SEND EVERY) またはメッセージを送信する時刻 (SEND AT) を指定します。SEND AT の場合、メッセージは 1 日に 1 回送信されます。

ユーザに SEND EVERY 句も SEND AT 句もない REMOTE 権限が付与されている場合、Message Agent はメッセージを処理した後、停止します。Message Agent を継続的に実行する場合は、REMOTE 権限を持つすべてのユーザに SEND AT または SEND EVERY の頻度が指定されていることを確認します。

多くの統合データベースでは Message Agent を定期的に実行するため、すべてのリモートデータベースに SEND 句が指定されている必要があります。ラップトップユーザに毎日 (SEND AT)、リモートサーバに 1、2 時間ごとに (SEND EVERY) メッセージを送信するというのが、一般的な設定です。効率的に運用するためには、この設定の時間をできるだけ変えずに使用してください。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、REMOTE 権限をユーザ Sam_Singer に付与し、SMTP 電子メールシステムを使用して、アドレス Singer, Samuel に 2 時間ごとにメッセージを送信するように設定します。

```
GRANT REMOTE TO Sam_Singer
TYPE SMTP
ADDRESS 'Singer, Samuel'
SEND EVERY '02:00';
```

関連情報

REVOKE REMOTE 文 [SQL Remote] [1277 ページ]
GRANT PUBLISH 文 [SQL Remote] [1147 ページ]

GRANT CONSOLIDATE 文 [SQL Remote] [1141 ページ]
CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]
CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote] [903 ページ]

1.4.4.188 GRANT USAGE ON SEQUENCE 文

指定されたシーケンスを使用する権限を付与します。

構文

```
GRANT USAGE ON SEQUENCE sequence-name
TO userid, ...
```

備考

この権限は、ユーザが指定したシーケンスから現在の値または次の値を選択できるようにします。

権限

MANAGE ANY OBJECT PRIVILEGE 権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

ANSI/ISO SQL 言語機能 T176 の一部。

例

チケッティングアプリケーションは、各チケットが一意の番号を持つようシーケンスを作成します。

```
CREATE SEQUENCE TicketNumber
   MINVALUE 1000
   MAXVALUE 100000;
```

チケットの生成にこのアプリケーションを使用しているユーザ TicketAgent は、シーケンスから値を選択できるよう、シーケンスに対する使用権を付与されている必要があります。

```
GRANT USAGE ON SEQUENCE TicketNumber to TicketAgent;
```

TicketAgent は、TicketNumber シーケンスから値を選択し、その値を使用して新しいチケットを作成できるようになりました。例:

```
INSERT INTO NewTicket
   SELECT TicketNumber.nextval, 'Concert Ticket', ...;
```

1.4.4.189 GROUP BY 句

カラム、エイリアス名、関数を SELECT 文の一部としてグループ化します。

構文

```
GROUP BY
| group-by-term, ...
| simple-group-by-term, ...  WITH ROLLUP
| simple-group-by-term, ... WITH CUBE
| GROUPING SETS ( group-by-term, ... )
```

```
group-by-term :
simple-group-by-term
| ( simple-group-by-term, ... )
| ROLLUP ( simple-group-by-term, ... )
| CUBE ( simple-group-by-term, ... )
```

```
simple-group-by-term :
expression
| ( expression )
| ()
```

パラメータ

GROUPING SETS 句

GROUPING SETS 句を使用すると、1 つのクエリ指定から複数のグループ化に対して集約操作を実行できます。GROUPING SET 句で指定した各セットは、GROUP BY 句と同等です。

たとえば、次の 2 つのクエリは同じです。

```
SELECT a, b, SUM( c ) FROM t
GROUP BY GROUPING SETS ( ( a, b ), ( a ), ( b ), ( ) );
```

```
SELECT a, b, SUM( c ) FROM t
  GROUP BY a, b
```

```
UNION ALL
SELECT a, NULL, SUM( c ) FROM t
  GROUP BY a
UNION ALL
SELECT NULL, b, SUM( c ) FROM t
  GROUP BY b
UNION ALL
SELECT NULL, NULL, SUM( c ) FROM t;
```

結果のローが所属するグループに応じて、グループ化の式を NULL 値として結果セットに反映することができます。この場合、NULL が別のグループ化の結果か、NULL が基本となるデータの実際の NULL 値の結果かについて混乱する可能性があります。入力データに存在する NULL 値とグループ化演算子によって挿入された NULL 値を区別するには、GROUPING 関数を使用します。

GROUPING SETS 句に空のかっこ ( ) を指定すると、全体の集約を含む 1 つのローを返します。

ROLLUP 句

1 つのクエリ指定内に複数のグループ化を指定するときに使用できるという点で、ROLLUP 句は GROUPING SETS 句と似ています。`nsimple-group-by-term` の ROLLUP 句は、`n`+1 のグループ化セットを生成します。形式は空のかっこで始まり、左から右に連続する `group-by-term` を付加して構成されます。

たとえば、次の 2 つの文は同等です。

```
SELECT a, b, SUM( c ) FROM t
GROUP BY ROLLUP ( a, b );
```

```
SELECT a, b, SUM( c ) FROM t
GROUP BY GROUPING SETS ( ( a, b ), a, ( ) );
```

GROUPING SETS 句内で ROLLUP 句を使用できます。

CUBE 句

1 つのクエリ指定内に複数のグループ化を指定するときに使用できるという点で、CUBE 句は ROLLUP 句と GROUPING SETS 句に似ています。CUBE 句は、CUBE 句に表示されている式から作成されるすべての可能な組み合わせを表すときに使用します。

たとえば、次の 2 つの文は同じです。

```
SELECT a, b, SUM( c ) FROM t
GROUP BY CUBE ( a, b, c );
```

```
SELECT a, b, SUM( c ) FROM t
GROUP BY GROUPING SETS ( ( a, b, c ), ( a, b ), ( a, c ),
 ( b, c ), a, b, c, () );
```

GROUPING SETS 句内で CUBE 句を使用できます。

WITH ROLLUP 句

これは ROLLUP 句の代わりになる構文であり、Transact-SQL との互換性のために提供されています。

WITH CUBE 句

これは CUBE 句の代わりになる構文であり、Transact-SQL との互換性のために提供されています。

備考

GROUP BY 句を使用している場合、式 (いくつかの制限付き)、カラム、エイリアス名、関数でグループ化できます。クエリの結果には、各グループ化セットの個々の値 (または複数の値) に 1 つのローが含まれます。

空の GROUP BY リスト () は、入力全体の処理を 1 つのグループとして表します。たとえば、次の 2 つの文は同じです。

```
SELECT COUNT(), SUM(Salary) FROM GROUPO.Employees;
```

```
SELECT COUNT(), SUM(Salary) FROM GROUPO.Employees GROUP BY ();
```

権限

なし。

標準

ANSI/ISO SQL 標準

コア機能。GROUPING SETS、GROUP BY ()、ROLLUP、CUBE は、オプションの ANSI/ISO SQL 言語機能 T431 の一部を構成します。ソフトウェアでは、オプションの ANSI/ISO SQL 言語機能 T432、"Nested and concatenated GROUPING SETS" はサポートされていません。

標準にない GROUP BY 句の拡張には、以下が含まれます。

- WITH ROLLUP
- WITH CUBE
- 任意の式を GROUP BY の単語として指定する機能。ANSI/ISO SQL 標準では、GROUP BY のすべての単語が、クエリの FROM 句内の基本となるテーブルからのカラム参照である必要があります。

例

次の例は、注文の総数を示す結果セットを返し、各年 (2000 と 2001) の注文数の小計を提供します。

```
SELECT year ( OrderDate ) Year, Quarter ( OrderDate ) Quarter, count(*) Orders
FROM GROUPO.SalesOrders
GROUP BY ROLLUP ( Year, Quarter )
ORDER BY Year, Quarter;
```

前の ROLLUP 演算の例と同様に、次の CUBE クエリの例は注文の総数を示す結果セットを返し、各年 (2000 と 2001) の注文数の小計を提供します。ROLLUP とは異なり、このクエリは各四半期 (1、2、3、4) の注文数の小計も示します。

```
SELECT year (OrderDate) Year, Quarter ( OrderDate ) Quarter, count(*) Orders
FROM GROUPO.SalesOrders
GROUP BY CUBE ( Year, Quarter )
ORDER BY Year, Quarter;
```

次の例は、2000 年および 2001 年の注文数の小計を示す結果セットを返します。GROUPING SETS 演算は、CUBE 演算のようにすべての小計の組み合わせを返すのではなく、小計するカラムを選択できます。

```
SELECT year (OrderDate) Year, Quarter ( OrderDate ) Quarter, count(*) Orders
FROM GROUPO.SalesOrders
GROUP BY GROUPING SETS ( ( Year, Quarter ), ( Year ) )
ORDER BY Year, Quarter;
```

関連情報

SELECT 文 [1291 ページ]
GROUPING 関数 [集合] [379 ページ]

1.4.4.190 HELP 文 [Interactive SQL]

Interactive SQL 環境のヘルプを提供します。

構文

```
HELP [ 'topic' ]
```

備考

HELP 文を使用して、SQL Anywhere のマニュアルにアクセスします。

ヘルプの `topic` をオプションで指定できます。`topic` は一重引用符で囲む必要があります。一部のヘルプフォーマットでは topic を指定できません。この場合、Interactive SQL の一般的なヘルプページへのリンクが表示されます。

`topic` には次の値を使用できます。

- SQL Anywhere エラーコード
- SQL 文のキーワード (INSERT、UPDATE、SELECT、CREATE DATABASE など)

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.191 IF 文

SQL 文の条件付き実行を制御します。

構文

```
IF search-condition THEN statement-list
[ ELSEIF { search-condition | operation-type } THEN statement-list ] ...
[ ELSE statement-list ]
{ END IF | ENDIF }
```

備考

IF 文は制御文です。これを使うと search-condition が TRUE と評価する SQL 文の最初のリストを条件付きで実行できます。いずれの search-condition も TRUE と評価されず、ELSE 句が存在する場合は、ELSE 句にある statement-list が実行されます。

実行は END IF の後に記述されている最初の文から再開されます。

注記

IF 文の構文と IF 式の構文を混同しないでください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

IF 文は、オプションの ANSI/ISO SQL 言語機能 P002、"Computational completeness" の一部です。ENDIF キーワードは標準にありません。

例

次のプロシージャは、IF 文の使い方を示します。

```
CREATE PROCEDURE TopCustomer2 (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
   DECLARE err_notfound EXCEPTION
   FOR SQLSTATE '02000';
   DECLARE curThisCust CURSOR FOR
   SELECT CompanyName, CAST(    sum(SalesOrderItems.Quantity *
   Products.UnitPrice) AS INTEGER) VALUE
   FROM Customers
   LEFT OUTER JOIN SalesOrders
   LEFT OUTER JOIN SalesOrderItems
   LEFT OUTER JOIN Products
   GROUP BY CompanyName;
   DECLARE ThisValue INT;
   DECLARE ThisCompany CHAR(35);
   SET TopValue = 0;
   OPEN curThisCust;
   CustomerLoop:
   LOOP
      FETCH NEXT curThisCust
      INTO ThisCompany, ThisValue;
      IF SQLSTATE = err_notfound THEN
         LEAVE CustomerLoop;
      END IF;
      IF ThisValue > TopValue THEN
         SET TopValue = ThisValue;
         SET TopCompany = ThisCompany;
      END IF;
   END LOOP CustomerLoop;
   CLOSE curThisCust;
END;
```

関連情報

BEGIN 文 [745 ページ]
IF 式 [35 ページ]
探索条件 [53 ページ]

1.4.4.192 IF 文 [T-SQL]

Watcom SQL IF 文の代わりに、SQL 文の条件付き実行を制御します。

構文

```
 IF expression statement
[ ELSE [ IF expression ] statement ]
```

備考

Transact-SQL IF と ELSE は、それぞれ単一の SQL 文または複合文 (キーワード BEGIN と END の間) の実行を制御します。

Watcom SQL IF 文と比較すると、Transact-SQL IF 文には THEN がありません。Transact-SQL バージョンには、ELSEIF または END IF キーワードもありません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Transact-SQL IF 文の使い方を示します。

```
IF (SELECT max(ID) FROM sysobjects) < 100
   RETURN
ELSE
   BEGIN
      PRINT 'These are the user-created objects'
      SELECT name, type, ID
```

```
      FROM sysobjects
      WHERE ID < 100
   END
```

次の 2 つの文のブロックは、Transact-SQL と Watcom SQL の互換性を示します。

```
/* Transact-SQL IF statement */
IF @v1 = 0
   PRINT '0'
ELSE IF @v1 = 1
   PRINT '1'
ELSE
   PRINT 'other'
/* Watcom SQL IF statement */
IF v1 = 0 THEN
   PRINT '0'
ELSEIF v1 = 1 THEN
   PRINT '1'
ELSE
   PRINT 'other'
END IF
```

1.4.4.193 INCLUDE 文 [ESQL]

SQL プリプロセッサによってスキャンされるソースプログラムにファイルをインクルードします。

構文

```
INCLUDE filename
```

```
filename : SQLDA | SQLCA | string
```

備考

INCLUDE 文は、C プリプロセッサ #include ディレクティブと非常によく似ています。SQL プリプロセッサは Embedded SQL ソースファイルを読み込んで、Embedded SQL 文をすべて C 言語のソースコードに置き換えます。SQL プリプロセッサに必要な情報が入ったファイルがある場合は、Embedded SQL INCLUDE 文でそのファイルをインクルードしてください。

2 つのファイル名 SQLCA と SQLDA が特別に認識されます。Embedded SQL ソースファイルでは、Embedded SQL 文の前に次の文が必要です。

```
EXEC SQL INCLUDE SQLCA;
```

この文を C 言語の静的変数宣言を置くことができる場所に入れます。多くの Embedded SQL 文では、SQLCA インクルード文の位置に SQL プリプロセッサが宣言する変数 (アプリケーション開発者には見えません) が必要となります。SQLDA ファイルを使用している場合は、SQLDA ファイルをインクルードします。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

1.4.4.194 INPUT 文 [Interactive SQL]

外部ファイル、キーボード、ODBC データソース、またはシェイプファイルからデータベーステーブルにデータをインポートします。

構文

構文:

外部ファイルまたはキーボードからのインポート

```
INPUT INTO [ owner.]table-name input-options
```

```
input-options :
[ ( column-name, ... ) ]
[ BYTE ORDER MARK { ON | OFF } ]
[ COLUMN WIDTHS ( integer, ...) ]
[ DELIMITED BY string ]
[ ENCODING encoding ]
[ ESCAPE CHARACTER character ]
[ ESCAPES { ON | OFF } ]
[ FORMAT input-format ]
[ FROM filename | PROMPT ]
[ NOSTRIP ]
[ SKIP integer ]
```

```
input-format : TEXT | FIXED
```

```
encoding : identifier | string
```

構文:

ODBC データソースからのインポート

```
INPUT USING connection-string
FROM source-table-name
INTO destination-table-name
[ CREATE TABLE { ON | OFF } ]
```

```
connection-string :
{ DRIVER=odbc-driver-name
| DSN=odbc-data-source } [ ; { connection-parameter = value } ]
```

構文:

シェイプファイルからのインポート

```
INPUT INTO [ owner.]table-name
FROM filename
FORMAT SHAPEFILE
[ SRID srid-number ]
[ ENCODING encoding ]
```

```
encoding : identifier | string
```

パラメータ

column-name

テーブルのカラムの名前を、入力ファイルの値の順序に対応する順序でリストします。テーブルカラムの順序が入力ファイルの値の順序と異なる場合、またはテーブルのカラム数よりも入力ファイルのカラム数のほうが少ない場合、このパラメータを使用します。

BYTE ORDER MARK 句

この句は、入力ファイルのバイトオーダマーク (BOM) を処理するかどうかを指定するときに使用します。

BYTE ORDER MARK 句は、TEXT フォーマットのファイルから読み込む場合にのみ関係します。TEXT 以外の FORMAT 句とともに BYTE ORDER MARK 句を使用すると、エラーが発生します。

BYTE ORDER MARK 句は、UTF-8 か UTF-16 でエンコードされたファイルの読み込みまたは書き込みをするときにのみ使用されます。その他のエンコードで BYTE ORDER MARK 句を使用しようとすると、エラーが発生します。

ENCODING 句が指定されている場合

- BYTE ORDER MARK オプションが ON のときに、UTF-16BE または UTF-16LE などのエンディアンを持つ UTF-16 エンコードを指定すると、Interactive SQL はデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの検証に使用されます。間違ったエンディアンを指定すると、エラーが返されます。
- BYTE ORDER MARK オプションが ON のときに、明示的なエンディアンのない UTF-16 エンコードを指定すると、Interactive SQL はデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの確認に使用されます。BOM がない場合は、オペレーティングシステムのエンディアンであると想定されます。
- BYTE ORDER MARK オプションが ON のときに、UTF-8 エンコードを指定すると、Interactive SQL はデータの先頭で BOM を検索します。BOM がある場合は無視されます。

ENCODING 句が指定されていない場合

- ENCODING 句を指定しないで BYTE ORDER MARK オプションを ON にすると、Interactive SQL は入力データの先頭で BOM を検索します。BOM が見つかると、BOM のエンコード (UTF-16BE、UTF-16LE、または UTF-8) に基づいてソースエンコードが自動的に選択され、BOM はロードするデータの一部であるとは見なされなくなります。
- ENCODING 句を指定せず、BYTE ORDER MARK オプションが OFF であるか、または BOM が入力データの先頭で見つからない場合は、クライアントコンピュータのエンコードが使用されます。

COLUMN WIDTHS 句

COLUMN WIDTHS 句で入力ファイルのカラムの幅を指定しますが、これは FIXED フォーマットの場合のみ指定できます。COLUMN WIDTHS を指定しない場合、幅はデータベースカラムのタイプによって決定されます。

CREATE TABLE 句

CREATE TABLE 句を使用して、インポート先テーブルが存在しない場合に作成するかどうかを指定します。デフォルトは ON です。CREATE TABLE ON 句を使用できるのは、ODBC データソースからのインポートの場合のみです。

DELIMITED BY 句

DELIMITED BY 句がサポートしているのは、TEXT 入力フォーマットのみです。この句を使用すると、TEXT 入力フォーマットのカラム値間でデリミタとして使用する文字列を指定できます。

デフォルトのデリミタは、少数点にピリオドを使用するロケールではカンマ、少数点にピリオドを使用するロケールではセミコロンです。

ENCODING 句

`encoding` 引数では、ファイルの読み込みに使用されるエンコードを指定します。ENCODING 句は、TEXT および SHAPEFILE フォーマットでのみ使用できます。

Interactive SQL を実行するとき、データのインポートに使用されるエンコードは次の順序で決定されます。

- ENCODING 句で指定されたエンコード (この句が指定されている場合)。
- default_isql_encoding オプションで指定されたエンコード (このオプションが設定されている場合)。
- ファイルにバイトオーダーマークがある場合は、適切な Unicode エンコードが使用されます。
- 実行しているコンピュータのデフォルトのエンコード。英語版 Windows コンピュータでは、デフォルトエンコードは 1252 または 850 です。

OUTPUT 文を使用して入力ファイルを作成したときにエンコードを指定した場合は、INPUT 文でも同じ ENCODING 句を指定してください。

ESCAPE CHARACTER 句

16 進コードに使用するデフォルトのエスケープ文字は円記号 (¥) です。たとえば、¥x0A は改行文字です。

改行文字は ¥n と指定できます。他の文字は、タブ文字は ¥x09 などのように、16 進 ASCII コードを使用して指定できます。2 つの円記号 (¥) は 1 つの円記号として解釈されます。円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字と解釈されます。たとえば、¥q は円記号と q と解釈されます。

エスケープ文字は、ESCAPE CHARACTER 句を使って変更することができます。たとえば、感嘆符 (!) をエスケープ文字として使用するには、次のように指定します。

```
... ESCAPE CHARACTER '!'
```

ESCAPES 句

ESCAPES を ON (デフォルト) にすると、データベースサーバによって円記号に続く文字は特殊文字として解釈されます。ESCAPES を OFF にすると、ソースに記載されているとおりに文字が読み取られます。

FORMAT 句

FORMAT 句によって、入力のファイルフォーマットを指定できます。FORMAT 句を指定していない場合、input_format オプションで指定したフォーマットが使用されます。FORMAT 句を指定した場合、input_format オプションの設定は無視されます。デフォルトの入力フォーマットは TEXT です。input では次のフォーマットを指定できます。

TEXT

入力行は文字であり、1 行あたり 1 つのローで構成され、カラム値はデリミタで区切られているものと見なされます。アルファベットの文字列を一重引用符または二重引用符で囲むことができます。デリミタを含む文字列は、一重引用符または二重引用符のどちらかで囲みます。文字列そのものに一重引用符または二重引用符を含める場合は、引用符文字を文字列の中で 2 つ続けて入力して使用してください。DELIMITED BY 句は、フィールドセパレータを指定するときに使用します。

他にも、3 つの特別なエスケープシーケンスを使用できます。2 文字の ¥n は改行文字を表し、¥¥ は 1 つの (¥) を表し、¥xDD は 16 進コード DD の文字を表します。

省略された値は NULL として扱われます。その位置の値を NULL にできない場合は、数値カラムには 0、文字カラムには空の文字列が挿入されます。

EXCEL

FORMAT EXCEL 句を使用して `.txt` または `.csv` の拡張子を持つファイルをインポートする場合、Microsoft Excel ワークブックファイルのデフォルトのフォーマットが使用されます。

使用されるデフォルトのワークシートは、ファイルの最初のワークシートです。特定のワークシートを指定するには、WORKSHEET 句を使用します。ワークリストの最初のローには、列ヘッダが含まれると見なされます。

ワークブックファイル (`.xl*`) の入力フォームは、Windows でのみサポートされています。

FIXED

入力行は固定フォーマットです。カラムの幅は、COLUMN WIDTHS 句を使って指定できます。カラムの幅を指定しない場合、ファイル内のカラム幅は、対応するデータベースカラムで許容される文字の最大値と同じ値にしてください。

FIXED フォーマットは、埋め込まれた改行とファイルの終わり文字シーケンスを含むバイナリカラムでは使用できません。

SHAPEFILE

入力は、ESRI シェイプファイルの形式です。LOAD 文と違って、INPUT 文を使用してシェイプファイルをロードする場合、シェイプファイルをクライアントコンピュータに置く必要があります。関連付けられている `.shx` ファイルと `.dbf` ファイルは、`.shp` ファイルと同じディレクトリに配置し、ベースファイル名を同じ名前にしてください。

シェイプファイルをロードするときにエンコードが指定されていない場合、デフォルトは ISO-8859-1 になります。

SRID 句を使用して、ジオメトリに関連付ける SRID を指定します。SRID を指定しない場合、デフォルトとして SRID 0 が使用されます。理想としては、シェイプファイルのプロジェクトファイル (`.prj`) に記載されている SRID と同じ SRID にしてください。この SRID を使用できない場合は、これと同等の SRID を使用します。SQL Anywhere には、sa_install_feature システムプロシージャを使用してデータベースに追加できる SRID が多数用意されています。

DBASE II、DBASE III、Microsoft Visual FoxPro、Lotus 123、または Microsoft Excel 97 など、その他のフォーマットを使用する場合は、INPUT USING を使用します。

FROM filename 句

`filename` は引用符で囲むことも囲まないこともできます。文字列を引用符で囲む場合、他の SQL 文字列と同じフォーマット要件に従います。

パスが引用されている場合、ディレクトリパスを示すには、円記号 (¥) を 2 つの円記号で表してください。

相対的な filename のロケーションは、次のように決定されます。

- Interactive SQL で INPUT 文を直接実行した場合、filename へのパスは、Interactive SQL が動作しているディレクトリとの相対パスとして解決されます。たとえば、ディレクトリ c:¥work から Interactive SQL を開き、次の文を実行すると想定します。

  ```
  INPUT INTO GROUPO.Employees
   FROM 'inputs¥¥inputfile.dat';
  ```

  Interactive SQL は c:¥work¥inputs¥inputfile.dat を検索します。
- Interactive SQL では、.sql ファイルに INPUT 文が存在する場合、最初に filename へのパスをこのファイルのロケーションとの相対パスとして解決しようとします。これで解決できなかった場合、Interactive SQL は、Interactive SQL の実行場所のディレクトリを基準に、filename を検索します。
  たとえば、ファイル c:¥homework¥inputs.sql があり、この中に次の文が含まれていると想定します。

  ```
  INPUT INTO GROUPO.Employees
   FROM 'inputs¥¥inputfile.dat';
  ```

  Interactive SQL は、最初に c:¥homework¥inputs から inputfile.dat を検索します。このロケーションで inputfile.dat が見つからない場合は、Interactive SQL が動作しているディレクトリを検索します。

FROM source-table-name 句

source-table-name パラメータは、送信元データベースのテーブル名を含む、引用符で囲まれた文字列です。名前は、database-name.owner.table-name、owner.table-name、または table-name のフォーマットにすることができます。ピリオドが送信元データベースのネイティブのセパレータではない場合も、コンポーネントの区切りにはピリオドを使用します。送信元データベースにデータベース名は必要であるが所有者名は不要な場合、source-table-name の形式は database..table とする必要があります (この場合、所有者名は空)。パラメータ内の名前を引用符で囲まないでください。たとえば、'dba."my-table"' を使用しないで、代わりに 'dba.my-table' を使用します。

INTO 句

データを入力するテーブルの名前です。

PROMPT 句

PROMPT 句では、ユーザがロー内の各カラムの値を入力できます。ウィンドウモードで実行している場合は、ユーザが新規ローの値を入力できるウィンドウが表示されます。コマンドラインから Interactive SQL を実行している場合は、コマンドラインに各カラムの値を入力するように求められます。

NOSTRIP 句

通常、TEXT 入力フォーマットの場合、引用符の付いていない文字列から後続ブランクを削除してから値を挿入します。NOSTRIP を使うと、後続ブランクを削除しないようにできます。後続ブランクは、オプションが使用されているかどうかにかかわらず、引用符付きの文字列からは削除されません。先行ブランクは、NOSTRIP オプション設定にかかわらず引用符の付いていない文字列から削除されます。

SKIP 句

TEXT ファイルから行を挿入する場合、SKIP 句はファイルの先頭から指定の行数を省略します。指定する数は正の整数にします。SKIP 句を使用できるのは、TEXT フォーマットの場合のみです。

ファイル内の行数を超える行数を指定した場合、INPUT 文はデータを一切挿入せず、エラーも返されません。

USING 句

USING 句は ODBC データソースからデータを入力します。DSN オプションを使用して ODBC データソース名を指定するか、または DRIVER オプションを使用して ODBC ドライバ名と接続パラメータを指定できます。`connection-parameter` は、データベース固有の接続パラメータのリストです。

`odbc-data-source` パラメータは、ユーザ名または ODBC データソース名です。たとえば、SQL Anywhere サンプルデータベースの `odbc-data-source` は SQL Anywhere 17 Demo です。

`odbc-driver-name` パラメータは ODBC ドライバ名です。SQL Anywhere データベースでは、`odbc-driver-name` は SQL Anywhere17 です。Ultra Light データベースでは、`odbc-driver-name` は Ultra Light 17 です。

WORKSHEET 句

WORKSHEET 句は、データのインポート元の Microsoft Excel ファイル内のワークシート名を識別します。この句を省略した場合は、Microsoft Excel ファイルの最初のシートが使用されます。

備考

INPUT 文を使用すると、指定したデータベーステーブルの中に効率よく大量に挿入できます。入力行は、入力ウィンドウを使用してユーザが入力するか (PROMPT を指定した場合)、またはファイルから読み込まれます (FROM `filename` を指定した場合)。どちらも指定しない場合、入力は INPUT 文を含む SQL スクリプトファイルから読み取られます。Interactive SQL では、この読み取りを *SQL Statements* ウィンドウ枠から直接行うこともできます。この場合、入力は、文字列 END のみを含む行で終了します。

*SQL 文*ウィンドウ枠から入力を直接読み込む場合、INPUT 文の末尾に挿入するレコード値の前に、セミコロンを指定する必要があります。例:

```
INPUT INTO Owner.TableName;
value1, value2, value3
value1, value2, value3
value1, value2, value3
value1, value2, value3
END
```

END キーワード (セミコロンなし) は、ファイル名が指定されておらず、PROMPT キーワードを含まない INPUT 文のデータを終了します。

カラムリストを指定すると、目的のテーブルの指定したカラムにデータが挿入されます。デフォルトで、INPUT 文では、入力ファイルのカラム値がデータベーステーブル定義のカラム値と同じ順序で出現していると想定されます。カラムの順序が異なる場合、`column-name` パラメータを使用して、テーブルカラムを入力ファイルのカラム値と同じ順序でリストする必要があります。

デフォルトでは、INPUT 文は、エラーの原因となるローを挿入しようとすると停止します。on_error と conversion_error オプションを設定すると、さまざまな方法でエラーを処理できます。

文字列値が INPUT でトランケートされる場合、Interactive SQL は *History* タブに警告を表示します。NOT NULL カラムで値が見つからない場合、数値型カラムは 0 に設定され、数値型でないカラムは空の文字列に設定されます。INPUT で NULL のローを挿入しようとすると、入力ファイルには空のローができます。

INPUT 文は Interactive SQL 文なので、IF 文などの複合文、ストアドプロシージャ、またはデータベースサーバが実行する任意の文では使用できません。

権限

テーブルの所有者であるか、INSERT ANY TABLE システム権限を持っているか、またはテーブルに対する INSERT 権限を持っていることが必要です。さらに、SELECT ANY TABLE システム権限、または、テーブルに対する SELECT 権限も必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、架空の TEXT ファイル `new_emp.inp` から Employees テーブルにデータをインポートします。

```
INPUT INTO GROUPO.Employees
   FROM new_emp.inp
   FORMAT TEXT;
```

この例では、テーブル ulTest をテーブル saTest にコピーします。ulTest は Ultra Light データベースのファイル `C:¥test¥myULDatabase.udb` 内のテーブルです。saTest は *demo.db* に作成されたテーブルです。

```
INPUT USING 'driver=UltraLite 17;dbf=C:¥¥test¥¥myULDatabase.udb'
   FROM "ulTest" INTO "saTest";
```

この例では、シェイプファイル `myshapefile.shp` を myTable テーブルにロードし、SRID 4269 をシェイプファイルのジオメトリに割り当てます。

```
INPUT INTO myTable
   FROM 'myshapefile.shp'
   FORMAT SHAPEFILE SRID 4269
```

この例では、Products テーブルに新しいローを追加し、各カラムに値を入力するようユーザに要求します。

```
INPUT INTO GROUPO.Products PROMPT;
```

この例では、ファイル `c:¥temp¥input.dat` から Employees テーブルにデータをロードします。バックスラッシュを 2 つ使用する方法に注意してください。

```
INPUT INTO GROUPO.Employees
FROM 'c:¥¥temp¥¥input.dat';
```

次の例では、テーブル myInventory を作成して、データを含む `stock.txt` ファイルからデータをインポートしますが、テーブル定義とはカラムの順序が異なります。順序の不一致を修正するには、INPUT 文のテーブル名の後に `column-`

name パラメータを使用して、インポートに必要な正しいカラムの順序を指定します。つまり、**(item, Quantity)** と指定することにより、入力ファイルの最初のカラム値をテーブルの 2 番目のカラムに配置し、入力ファイルの 2 番目のカラム値をテーブルの最初のカラムに配置するよう、Interactive SQL に指示します。

```
CREATE TABLE myInventory (
Quantity INTEGER,
item VARCHAR(60)
);
INPUT INTO myInventory (item, Quantity)
FROM stock.txt;
```

次の例では、ファイル c:¥test¥Book1.xlsx の最初のワークシートから inputTest という名前の既存のテーブルにデータをロードします。

```
INPUT INTO inputTest FROM 'c:¥¥test¥¥Book1.xlsx' FORMAT EXCEL
```

関連情報

OUTPUT 文 [Interactive SQL] [1222 ページ]
INSERT 文 [1167 ページ]
SET OPTION 文 [Interactive SQL] [1314 ページ]
LOAD TABLE 文 [1182 ページ]
sa_install_feature システムプロシージャ [1539 ページ]

1.4.4.195 INSERT 文

データベースの任意の場所から、1 つのローあるいは選択されたローをテーブルへ挿入します。

構文

1 つ以上のローの挿入

```
INSERT [ INTO ] [ owner.]table-name [ ( column-name, ... ) ]
[ ON EXISTING {
    ERROR
    | SKIP
    | UPDATE [ DEFAULTS { ON | OFF } ]
    } ]
{ DEFAULT VALUES
    | VALUES row-value-constructor }
[ OPTION( query-hint [, ... ] ) ]
```

SELECT 文の結果の挿入

```
INSERT [ INTO ] [ owner.]table-name [ ( [ column-name [, ... ] ] ) ]
[ ON EXISTING {
    ERROR
    | SKIP
    | UPDATE [ DEFAULTS { ON | OFF } ]
```

```
    } ]
[ WITH AUTO NAME ]
select-statement
[ OPTION( query-hint[, ... ] ) ]
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| FORCE NO OPTIMIZATION
| option-name = option-value
```

```
option-name :
 identifier
```

```
option-value :
 hostvar （許可されたインジケータ）
| string
| identifier
| number
```

```
insert-expression :
 expression | DEFAULT
```

```
row-value-constructor :
 ( [ insert-expression [, ... ] ] ) [,( [ insert-expression [, ... ] ] ) ... ]
```

パラメータ

VALUES 句

VALUES 句は、挿入する値を指定するときに使用します。対象となるカラムに定義されているデフォルト値を挿入する場合は、DEFAULT VALUES を指定します。VALUES () と指定しても DEFAULT VALUES と同義になります。また、VALUES 句では、1 つの文に複数のローの値を挿入できるように、ロー値コンストラクタがサポートされています。各 `row-value-constructor` 内の `insert-expression` 値の数と順序は、INTO 句で指定したカラムリストに対応させてください。カラムリストを指定していない場合は、テーブルの完全な順序のカラムリストであると見なされます。空のカラムリスト () を指定する場合は、テーブルの各カラムにデフォルト値が含まれている必要があります。複数のロー値コンストラクタを指定する場合は、カンマで区切る必要があります。

ローの挿入時にエラーが発生すると、すべての変更がロールバックされます。

WITH AUTO NAME 句

WITH AUTO NAME は、構文 2 にのみ適用されます。WITH AUTO NAME を指定すると、クエリブロックの項目名によってデータの所属カラムが決まります。クエリブロックの項目には、カラム参照かエイリアスの式を指定してください。クエリブロックで定義されていない送信先カラムには、デフォルト値が割り当てられます。これは送信先テーブルのカラム数が多い場合に役立ちます。

WITH AUTO NAME 句を指定し、ターゲットテーブルのカラムと一致しないカラムがクエリブロックに含まれている場合、INSERT 文はエラーを返します。たとえば、次の文を実行すると、テーブルで SELECT ブロックの操作カラムを見つけることができないことを示すエラーが返されます。

```
CREATE TABLE MyTable5(
      pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
```

```
      TableName CHAR(128),
      TableNameLen INT );
INSERT
INTO MyTable5 WITH AUTO NAME
SELECT length(t.table_name) AS TableNameLen,
      t.table_name AS TableName, 1 as operation
FROM SYS.SYSTAB t
WHERE table_id <= 10;
```

ON EXISTING 句

INSERT 文の ON EXISTING 句は、両方の構文に適用されます。ON EXISTING 句は、プライマリキールックアップに基づいて、新しいカラム値でテーブルの既存のローを更新します。この句は、プライマリキーが設定されたテーブルでのみ使用できます。プライマリキーがないテーブルでこの句を使用すると、構文エラーになります。ON EXISTING を指定してプロキシテーブルに値を挿入することはできません。

> **注記**
>
> ON EXISTING 条件を満たすローが数多く存在する可能性がある場合は、代わりに MERGE 文を使用してください。MERGE 文を使用すると、一致するローに対して実行するアクションをより詳細に制御できます。また、一致を定義するためのより高度な構文も使用できます。

ON EXISTING 句を指定すると、データベースサーバは各入力ローに対してプライマリキールックアップを実行します。対応するローがテーブルに存在しなければ、新しいローを挿入します。ローがテーブルにすでに存在する場合は、入力ローを無視する (SKIP)、重複したキー値のエラーメッセージを生成する (ERROR)、入力ローの値を使用して古い値を更新する (UPDATE) 操作のいずれかを選択できます。ON EXISTING 句を指定しなかった場合は、デフォルトでは、ローがすでに存在するテーブルにローを挿入しようとすると、キー値の重複エラーになります。また、ON EXISTING ERROR 句を指定するときと同じです。スキップされるローは、@@rowcount 変数に含まれます。

ON EXISTING UPDATE を使用している場合、入力したローは格納されているローと比較されます。入力ローに指定されているすべてのカラム値は、格納されているローの対応するカラム値を置換します。同様に、入力ローに明示的にカラム値が指定されていない場合、格納されているローの対応するカラム値は変化しません。ただし、デフォルト値があるカラムの場合は例外です。また、デフォルト値があるカラム (DEFAULT AUTOINCREMENT カラムなど) を含む ON EXISTING UPDATE 句を使用する場合、ON EXISTING UPDATE DEFAULTS ON を指定することでデフォルト値でカラム値を更新するか、ON EXISTING UPDATE DEFAULTS OFF を指定してカラム値をそのままにすることができます。何も指定しない場合、デフォルトの動作は ON EXISTING UPDATE DEFAULTS OFF です。

> **注記**
>
> DEFAULTS ON パラメータと DEFAULTS OFF パラメータは、DEFAULT TIMESTAMP、DEFAULT UTC TIMESTAMP、または DEFAULT LAST USER の値に関係ありません。これらのカラムでは、格納されているローの値は UPDATE 中に常に更新されます。

ON EXISTING SKIP 句と ON EXISTING ERROR 句を使用するときにテーブルにデフォルトのカラムが含まれていると、サーバは、すでに存在するローに対しても、デフォルト値を計算します。結果として、AUTOINCREMENT のようなデフォルト値が、スキップされたローに対しても影響を及ぼします。AUTOINCREMENT の場合は、AUTOINCREMENT のシーケンスで値がスキップされます。次の例で説明します。

```
CREATE TABLE t1( c1 INT PRIMARY KEY, c2 INT DEFAULT AUTOINCREMENT );
INSERT INTO t1( c1 ) ON EXISTING SKIP VALUES( 20 );
INSERT INTO t1( c1 ) ON EXISTING SKIP VALUES( 20 );
INSERT INTO t1( c1 ) ON EXISTING SKIP VALUES( 30 );
```

最初の INSERT 文に定義されたローが挿入され、c2 は 1 に設定されます。2 番目の INSERT 文に定義されたローは、既存のローに一致するため、スキップされます。ただし、オートインクリメントカウンタは 2 に増分されたまま残ります (既存のローには影響ありません)。3 番目の INSERT 文に定義されたローが挿入され、c2 の値は 3 に設定されます。したがって、上記の例で挿入された値は次のようになります。

```
20,1
30,3
```

> ⚠ 警告
>
> SQL Remote を使用している場合、DEFAULT LAST USER カラムをレプリケートしないでください。カラムをレプリケートすると、カラム値はレプリケートされた値ではなく、SQL Remote ユーザに設定されます。

OPTION 句

この句は、文を実行するためのヒントを指定するときに使用します。指定する設定は、現在の文にのみ適用され、ODBC 実行可能アプリケーションによる設定など、パブリックオプションまたはテンポラリオプションの設定よりも優先されます。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- FORCE NO OPTIMIZATION
- `option-name` = `option-value`.
  クエリテキスト内の `OPTION( isolation_level = ... )` の指定を使用すると、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。
  クエリテキスト内の `OPTION( parameterization_level = ... )` の指定を使用すると、クエリのパラメータ化レベルよりも優先されます。

備考

INSERT 文を使って、データベーステーブルに新しいローを追加します。

テキストインデックスとマテリアライズドビューは基本となるテーブルデータの変更によって影響を受けるため、基本となるテーブルにデータをバルクロード (LOAD TABLE、INSERT、MERGE) する前に、従属するテキストインデックスまたはマテリアライズドビューをトランケートすることを検討してください。

1 つ以上のローの挿入

指定された式のカラム値を使用して、1 つのローまたは複数のローを挿入します。複数のローを指定する場合は、追加のカッコで区切ります。キーワード DEFAULT を使うと、カラムのデフォルト値を挿入できます。オプションのカラム名のリストを指定すると、指定したカラムの中に値が 1 つずつ挿入されます。カラム名のリストを指定しないと、作成順 (SELECT * を使って取り出すときと同じ順序) に値がテーブルカラムの中に挿入されます。ローは、テーブル内の任意の位置に挿入されます。リレーショナルデータベースでは、テーブルの順序は指定されません。

SELECT 文の結果の挿入

一般的な SELECT 文の結果を使用して、テーブルに大量に挿入します。SELECT 文に ORDER BY 句が含まれていない場合、任意の順序で挿入が行われます。

カラム名を指定すると、SELECT リストのカラムは、カラムリストに指定されたカラムまたは作成順に並べたカラムと順に一致します。

ビューを定義しているクエリ指定が更新可能な場合は、ビューに対して挿入操作を実行できます。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字の状態のままで格納されます。そのため、テーブルに挿入される文字列 'Value' は、常に V が大文字、残りの文字が小文字でデータベースに格納されます。SELECT 文は、文字列を Value として返します。ただし、データベースで大文字と小文字が区別されない場合は、すべての比較において Value は value、VALUE などと同じと見なされます。さらに、単一カラムのプライマリキーにエントリ Value がある場合は、プライマリキーがユニークでなくなるので、INSERT 文による value の追加は拒否されます。

INSERT 文を使用してデータを大量に挿入する場合も、カラム統計は更新されます。

> **注記**
>
> テーブルに多数のローを挿入するには、個別の INSERT 文を多数実行するよりも、可能であればカーソルを宣言し、そのカーソルを通じてローを挿入する方が効率的です。データを挿入する前に、テーブルページごとに今後の更新のために確保する空き領域の割合を指定できます。

権限

テーブルの所有者であるか、INSERT ANY TABLE 権限を持っているか、またはテーブルに対する INSERT 権限を持っていることが必要です。さらに、ON EXISTING UPDATE 句が指定されている場合は、UPDATE ANY TABLE システム権限を持っているか、またはテーブルに対する UPDATE 権限を持っていることが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。DEFAULT VALUES 句は、オプションの ANSI/ISO SQL 言語機能 F222、"INSERT statement: DEFAULT VALUES clause" です。INSERT 文でのロー値コンストラクタのサポートは、オプションの ANSI/ISO SQL 言語機能 F641、"Row and table constructors" の一部です。ソフトウェアで挿入する値を指定するために必要な VALUES キーワードは、標準の一部ではありません。

以下の拡張子も ANSI/ISO SQL 標準の一部ではありませんが、ソフトウェアではサポートされています。

- INSERT...ON EXISTING 句は標準にありません。多くの場合、ANSI/ISO SQL 準拠の機能では、MERGE 文が同義となります。
- OPTION 句。
- WITH AUTO NAME 句。

例

データベースに Eastern Sales 部を追加します。

```
INSERT INTO GROUPO.Departments ( DepartmentID, DepartmentName )
VALUES ( 230, 'Eastern Sales' );
```

テーブル DepartmentHead を作成し、WITH AUTO NAME 構文を使用して部長とその部の名前を挿入します。

```
CREATE TABLE DepartmentHead(
      pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
      DepartmentName VARCHAR(128),
      ManagerName VARCHAR(128) );
INSERT
INTO DepartmentHead WITH AUTO NAME
SELECT GivenName || ' ' || Surname AS ManagerName,
      DepartmentName
FROM GROUPO.Employees JOIN GROUPO.Departments
ON EmployeeID = DepartmentHeadID;
```

テーブル MyTable6 を作成し、WITH AUTO NAME 構文を使用してデータを移植します。

```
CREATE TABLE MyTable6(
      pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
      TableName CHAR(128),
      TableNameLen INT );
INSERT INTO MyTable6 WITH AUTO NAME
SELECT
      length(t.table_name) AS TableNameLen,
      t.table_name AS TableName
FROM SYS.SYSTAB t
WHERE table_id <= 10;
```

データベースの現在の独立性レベル設定ではなく、独立性レベル 3 で文を実行し、新しい部署を挿入します。

```
INSERT INTO GROUPO.Departments
   (DepartmentID, DepartmentName, DepartmentHeadID)
   VALUES(600, 'Foreign Sales', 129)
   OPTION( isolation_level = 3 );
```

次の例では、架空のテーブル T に 3 つのローを挿入します。

```
INSERT INTO T (c1,c2,c3)
VALUES (1,10,100), (2,20,200), (3,30,300);
```

次の例では、架空のテーブル T に、各カラムにデフォルト値が定義されている 4 つのカラムの 3 つのローを挿入します。

```
INSERT INTO T ()
VALUES (), (), ();
```

関連情報

SQL 変数 [118 ページ]
ALTER TABLE 文 [705 ページ]
MERGE 文 [1204 ページ]

INPUT 文 [Interactive SQL] [1160 ページ]
LOAD TABLE 文 [1182 ページ]
UPDATE 文 [1391 ページ]
SELECT 文 [1291 ページ]
TRUNCATE 文 [1366 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
DELETE 文 [1016 ページ]
PUT 文 [ESQL] [1244 ページ]

1.4.4.196 INSTALL EXTERNAL OBJECT 文

外部環境で実行できるオブジェクトをインストールします。

構文

```
INSTALL EXTERNAL OBJECT object-name
[ update-mode ]
FROM { FILE file-path | VALUE expression }
ENVIRONMENT environment-name
[ AS USER user-name ]
```

```
environment-name :
PERL
| PHP
| JS
```

```
update-mode :
NEW
| UPDATE
```

パラメータ

object-name

データベース内で、インストールしたオブジェクトを識別する名前。

update-mode

オブジェクトの更新モード。更新モードを省略すると、NEW が使用されます。

file-path

オブジェクトのインストール元となる、サーバコンピュータのロケーション。

environment-name

外部オブジェクトが実行される外部環境の名前。

AS USER 句

オブジェクトの所有者を指定します。

備考

なし。

権限

MANAGE ANY EXTERNAL OBJECT システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、ファイルに格納されている Perl スクリプトをデータベースにインストールします。

```
INSTALL EXTERNAL OBJECT 'PerlScript'
NEW
FROM FILE 'perlfile.pl'
ENVIRONMENT PERL;
```

また、次のように、Perl コードを式から構築してインストールできます。

```
INSTALL EXTERNAL OBJECT 'PerlConsoleExample'
NEW
FROM VALUE 'sub WriteToServerConsole { print $sa_output_handle $_[0]; }'
ENVIRONMENT PERL;
```

また、次のように、Perl コードを変数から構築してインストールできます。

```
CREATE VARIABLE PerlVariable LONG VARCHAR;
SET PerlVariable =
  'sub WriteToServerConsole { print $sa_output_handle $_[0]; }';
INSTALL EXTERNAL OBJECT 'PerlConsoleExample'
NEW
FROM VALUE PerlVariable
ENVIRONMENT PERL;
```

関連情報

ALTER EXTERNAL ENVIRONMENT 文 [647 ページ]
REMOVE EXTERNAL OBJECT 文 [1263 ページ]
START EXTERNAL ENVIRONMENT 文 [1338 ページ]
STOP EXTERNAL ENVIRONMENT 文 [1348 ページ]
SYSEXTERNENV システムビュー [1802 ページ]
SYSEXTERNENVOBJECT システムビュー [1804 ページ]

1.4.4.197 INSTALL JAVA 文

Java クラスをデータベース内で使用できるようにします。

構文

```
INSTALL JAVA
[ NEW | UPDATE ]
[ JAR jar-name ]
FROM { FILE filename | expression }
[ AS USER user-name ]
```

パラメータ

NEW キーワードと UPDATE キーワードの句

NEW を指定した場合、参照する Java クラスまたは JAR ファイルには、現在インストールされているクラスを更新したものではなく、新しいクラスが含まれている必要があります。使用するデータベース内または NEW に同じ名前のクラスまたは JAR ファイルがある場合は、エラーが発生します。

UPDATE を指定すると、参照するファイルは指定されたデータベースにすでにインストールされている Java クラスまたは JAR ファイルの代替を含むことがあります。

省略した場合、デフォルトは NEW になります。

JAR 句

この句を指定する場合は、`filename` に JAR ファイルを指定します。JAR ファイルは、通常 `.jar` または `.zip` の拡張子を持ちます。

インストールされた JAR ファイルと ZIP ファイルは、圧縮または圧縮解除できます。圧縮スキームには違いがあるため、テキストリソースを含む JAR ファイルは圧縮をオフにした状態で作成することをお奨めします。

JAR オプションを指定すると、JAR ファイルの持つクラスがインストールされた後で JAR として保持されます。この JAR は、これらの各クラスに関連付けられた JAR です。JAR 句と一緒にデータベースにインストールされている JAR ファイルは、データベースの保持された JAR ファイルと呼ばれます。

jar-name は、255 バイト以内の文字列値です。後続の INSTALL JAVA UPDATE 文と REMOVE JAVA 文で保持された JAR を識別するために、jar-name が使用されます。

FROM FILE 句

インストールする Java クラスまたは JAR ファイルのロケーションを指定します。

filename でサポートされる形式は、'c:¥¥libs¥¥jarname.jar' や '/usr/u/libs/jarname.jar' のような完全修飾ファイル名と、データベースサーバの現在の作業ディレクトリを基準にした相対ファイル名です。データベースサーバの CLASSPATH にはクラスまたは JAR ファイルのパスが含まれており、そのパスはファイル名に含める必要はありません。

filename には、クラスファイル、JAR または ZIP ファイルを指定してください。

FROM 句

式は、値として有効なクラスまたは JAR ファイルを持つバイナリ型に評価されるようにします。

AS USER 句

オブジェクトの所有者を指定します。

備考

各クラスのクラス定義は、最初にそのクラスを使用するときに各接続の VM でロードされます。クラスをインストールすると、接続の VM が暗黙的に再起動されます。したがって、新しいクラスに直ちにアクセスできます。VM が再起動されるため、Java 静的変数に格納されていた値はすべて失われ、Java クラスタイプの SQL 変数はすべて削除されます。

他の接続では、新しいクラスは VM が最初にクラスにアクセスして次のときにロードされます。VM がそのクラスをロード済みの場合、その接続で VM を再起動するまで、その接続から新しい接続は見えません。

すべてのインストールされたクラスは、すべてのユーザがすべての方法で参照できます。

権限

MANAGE ANY EXTERNAL OBJECT システム権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、クラスの名前とロケーションを指定して架空の Java クラス Demo をインストールします。

```
INSTALL JAVA NEW
FROM FILE 'D:¥¥JavaClass¥¥Demo.class';
```

この例では、架空の ZIP ファイルに含まれるすべてのクラスをインストールし、それらを JAR 名 Widget でデータベース内に関連付けます。インストール後、ZIP ファイルのロケーションは保持されません。クラスは完全に修飾されたクラス名 (パッケージ名とクラス名) を参照しなければなりません。

```
INSTALL JAVA
JAR 'Widgets'
FROM FILE 'C:¥¥Jars¥¥Widget.zip';
```

関連情報

REMOVE JAVA 文 [1264 ページ]
SYSJAR システムビュー [1812 ページ]
SYSJARCOMPONENT システムビュー [1813 ページ]
SYSJAVACLASS システムビュー [1814 ページ]

1.4.4.198 INTERSECT 文

2 つ以上のクエリの結果セット間の共通部分を計算します。

構文

```
[ WITH temporary-views ] query-block
  INTERSECT [ ALL | DISTINCT ] query-block
[ ORDER BY [ integer | select-list-expression-name ] [ ASC | DESC ], ... ]
[ FOR XML xml-mode ]
[ OPTION( query-hint, ... ) ]
```

```
query-block : a query block
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

```
option-name : identifier
```

```
option-value :
hostvar （許可されたインジケータ）
| string
| identifier
| number
```

パラメータ

query-block

クエリブロックについては、SQL 構文の共通要素のマニュアルを参照してください。

FOR XML 句

FOR XML 句については、SELECT 文のマニュアルを参照してください。

OPTION 句

この句は、文を実行するためのヒントを指定するときに使用します。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- `option-name` = `option-value`.クエリテキスト内の `OPTION( isolation_level = ... )` の指定は、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。

備考

INTERSECT は、2 つのクエリブロックの結果セット間の集合積を計算します。クエリブロックはネストでき、ネストされた SELECT 文や、集合演算子 UNION、EXCEPT、または INTERSECT を含めることができます。INTERSECT の単独での指定は、INTERSECT DISTINCT の指定と同義です。

INTERSECT ALL では、集合積ではなく、バッグ積を実装します。たとえば、最初の `query-block` 内に特定の値を持つ 5 つの (重複) ローがあり、2 番目の `query-block` 内に同一の値を持つ 3 つの重複ローがある場合、INTERSECT ALL は 3 つのローを返します。

`query-block` 内に重複ローがない場合、INTERSECT の結果は INTERSECT ALL と同じになります。

2 つの `query-block` の結果セットは和両立する必要があります。それぞれの SELECT リストには同数の項目が含まれており、かつ、各項目の型は比較可能である必要があります。2 つの SELECT リスト内の対応する項目が異なるデータ型の場合、データベースサーバは結果の中から対応するカラムのデータ型を選択し、各 `query-block` のカラムを自動的にそれぞれ変換します。

表示されるカラム名は最初の `query-block` に表示されるカラム名と同じであり、これらの名前は ORDER BY 句と一致する式の名前を判断するために使用されます。結果セットのカラム名をカスタマイズする別の方法として、共通のテーブル式 (WITH 句) を使用する方法があります。

権限

SELECT ANY TABLE システム権限を持っているか、`query-block` で指定されたオブジェクトの所有者であるか、各クエリブロック上で SELECT 権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

INTERSECT は、オプションの ANSI/ISO SQL 言語機能 T302 です。INTERSECT とともに DISTINCT キーワードを明示的に指定する方法は、オプションの ANSI/ISO SQL 言語機能 T551 です。INTERSECT とともに ORDER BY 句を指定する方法は、SQL 言語機能 F850 です。ORDER BY 句を含む `query-block` は、ANSI/ISO SQL 機能 F851 の構成要素です。ロー制限句 (SELECT TOP または LIMIT) を含むクエリブロックは、コンテキストに応じて、オプションの ANSI/ISO SQL 言語機能 F857 または F858 を構成します。FOR XML 句と OPTION 句は、標準の一部ではありません。

Transact-SQL

INTERSECT は、Adaptive Server Enterprise ではサポートされていません。ただし、SQL Anywhere でサポートされている Transact-SQL ダイアレクトでは、INTERSECT ALL と INTERSECT DISTINCT の両方を使用できます。

関連情報

EXCEPT 文 [1088 ページ]
UNION 文 [1373 ページ]
SELECT 文 [1291 ページ]

1.4.4.199 LEAVE 文

複合文またはループから出ます。

構文

```
LEAVE [ statement-label ]
```

備考

LEAVE 文は、指定したラベル (`statement-label`) の複合文または指定したラベルのループを離れるための制御文です。LEAVE では、ループまたは BEGIN...END ブロックの先頭にラベルを付けることができます。`statement-label` を指

定しないと、LEAVE は最も内側の BEGIN...END ブロックではなく、最も内側のループで終了します。実行は、複合文またはループの後に記述されている最初の文から再開されます。

ラベルを使用しない LEAVE 文の使用は、Transact-SQL BREAK 文の使用と同等です。

プロシージャまたはトリガの本文である複合文は、プロシージャまたはトリガの名前と同じ暗黙のラベルを持っています。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

オプションの言語機能 P002、"Computational completeness" の一部。

例

次のフラグメントは、LEAVE 文を使ってラベルなしにループを離れる方法を示します。

```
SET i = 1;
lbl:
LOOP
   INSERT
   INTO Counters ( number )
   VALUES ( i );
   IF i >= 10 THEN
      LEAVE;
   END IF;
   SET i = i + 1
END LOOP lbl
```

次のフラグメントは、ネストされたループ内で LEAVE を使用します。

```
outer_loop:
LOOP
   SET i = 1;
   inner_loop:
   LOOP
      ...
      SET i = i + 1;
      IF i >= 10 THEN
         LEAVE outer_loop
      END IF
   END LOOP inner_loop
```

```
END LOOP outer_loop
```

関連情報

LOOP 文 [1203 ページ]
FOR 文 [1106 ページ]
BEGIN 文 [745 ページ]

1.4.4.200 LOAD STATISTICS 文

この文は内部でのみ使用され、dbunload ユーティリティが使用して古いデータベースからカラム統計を ISYSCOLSTAT システムテーブルにアンロードします。

構文

```
LOAD STATISTICS [ [ owner.]table-name.]column-name
 format-id, density, max-steps, actual-steps, step-values, frequencies
```

パラメータ

format-id

ISYSCOLSTAT システムテーブルにある残りのローのフォーマットを決定するための内部フィールド。
density

カラムの単一の値の加重平均による選択性の推定。ローに格納されている大きい単一値の選択性は考慮されません。
max-steps

ヒストグラム内で可能な最大ステップ数。
actual-steps

この時点で実際に使用されているステップ数。
step-values

ヒストグラムのステップの境界値。
frequencies

ヒストグラムのステップの選択性。

権限

MANAGE ANY STATISTICS システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

SYSCOLSTAT システムビュー [1796 ページ]

1.4.4.201 LOAD TABLE 文

外部ファイルからデータベーステーブルにバルクデータをインポートします。

構文

```
LOAD [ INTO ] TABLE [ owner.]table-name |view-name
[ ( column-name, ... ) ]
load-source
[ load-option ... ]
[ statistics-limitation-option ]
```

```
load-source :
{ FROM filename-expression
   | USING FILE filename-expression
   | USING CLIENT FILE client-filename-expression
   | USING VALUE value-expression
   | USING COLUMN column-expression }
```

```
filename-expression :string | variable
```

```
client-filename-expression :string | variable
```

```
value-expression :expression
```

```
column-expression :
column-name
   FROM table-name
   ORDER BY column-list
```

```
load-option :
ALLOW { integer | ALL | NO } ERRORS ]
| BYTE ORDER MARK { ON | OFF }
| CHECK CONSTRAINTS { ON | OFF }
| { COMPRESSED | AUTO COMPRESSED | NOT COMPRESSED }
| COMMENTS INTRODUCED BY comment-prefix
| COMPUTES { ON | OFF }
| DEFAULTS { ON | OFF }
| DELIMITED BY string
| ENCODING encoding
| { ENCRYPTED KEY 'key' | NOT ENCRYPTED }
| ESCAPE CHARACTER character
| ESCAPES { ON | OFF }
| FORMAT {
    TEXT
    | BCP
    | XML row-xpath ( column-xpath,...) [ NAMESPACES namespace ] }
    | SHAPEFILE
| HEXADECIMAL { ON | OFF }
| MESSAGE LOG log-target
| ORDER { ON | OFF }
| PCTFREE percent-free-space
| QUOTE string
| QUOTES { ON | OFF }
| ROW DELIMITED BY string
| ROW LOG log-target
| SKIP integer
| STRIP { OFF | LTRIM | RTRIM | BOTH }
| WITH CHECKPOINT { ON | OFF }
| WITH { FILE NAME | ROW | CONTENT } LOGGING
```

```
statistics-limitation-option :
STATISTICS {
   ON [ ALL COLUMNS ]
   | ON KEY COLUMNS
   | ON ( column-list )
   | OFF
   }
```

```
comment-prefix :string
```

```
encoding :string
```

```
log-target :{
FILE server-filename
| CLIENT FILE client-filename
| VARIABLE variable-name
}
```

パラメータ

view-name

この句は、データのロード先となる通常のビューを指定するときに使用します。ビューの定義はベーステーブルに基づく必要があります。ビュー定義は次のように定義します。

```
CREATE VIEW
[owner.]view-name
AS SELECT * FROM base-table-name
```

column-name

この句を使用して、データのロード先となる 1 つ以上のカラムを指定します。DEFAULTS が OFF に設定されている場合、カラムリストにないカラムは NULL になります。DEFAULTS が ON に設定されていて、カラムにデフォルト値が入っている場合は、その値が使用されます。DEFAULTS が OFF に設定されていて、NULL 入力不可のカラムがカラムリストから省かれている場合は、データベースサーバは、空の文字列をカラムの型に変換しようとします。

カラムリストが指定されていると、ファイルに存在すると思われるカラムと、想定されるファイル内でのカラムの順序がカラムリストにリストされます。カラム名を繰り返すことはできません。リストにないカラム名は、NULL、0、空、または DEFAULT に設定されます。設定値は、カラムに NULL が許可されているかどうか、どのデータ型か、DEFAULTS が ON か OFF かによって異なります。LOAD TABLE によって無視される入力ファイル内のカラムは、カラム名 **filler()** を使用して指定できます。

load-source

この句は、データのロード元となるデータソースを指定するときに使用します。さまざまなデータソースから、データをロードできます。

LOAD TABLE 文が BEGIN PARALLEL WORK 文の内側にある場合、load-source パラメータは USING FILE 文や FROM 文のみをサポートします。

FROM clause

この句は、ファイルを指定するときに使用します。filename-expression は、文字列としてサーバに渡されます。したがって、文字列は他の SQL 文字列と同じデータベースのフォーマット要件に従います。特に、次の点に注意してください。

- フォルダパスを示すには、円記号 (¥) を 2 つの円記号で表してください。
- パス名はデータベースサーバを基準にした相対パスを指定します。クライアントアプリケーションではありません。
- UNC パス名を使用すると、データベースサーバ以外のコンピュータ上のファイルからデータをロードできます。

USING FILE clause

この句は、ファイルからデータをロードするときに使用します。これは、FROM filename 句を指定する場合と同義です。

LOAD TABLE 文を USING FILE 句とともに使用する場合、進行状況メッセージを要求できます。

また、Progress 接続プロパティを使用して、文がどの程度実行されたかを確認することもできます。

USING CLIENT FILE clause

この句は、クライアントコンピュータ上のファイルからデータをロードするときに使用します。データベースサーバが client-filename-expression からデータを取り出す場合、データはサーバのメモリに実体化されないため、ファイルにはデータベースサーバでの BLOB 式サイズの制限が適用されません。したがって、クライアントファイルには任意のサイズを指定できます。

クライアント側データの転送は、SQL Anywhere データベースドライバを使用するクライアントに対してサポートされます。SAP Adaptive Server Enterprise、jConnect JDBC ドライバ、または SAP Open Client アプリケーションによって使用される **Tabular Data Stream** (**TDS**) プロトコルではサポートされません。

テーブルがクライアントファイルから読み込まれている場合、ファイル名のロギングはできません。ロギングタイプが指定されていない場合、WITH CONTENT LOGGING が使用されます。

LOAD TABLE 文を USING CLIENT FILE 句とともに使用する場合、進行状況メッセージを要求できます。

また、Progress 接続プロパティを使用して、文がどの程度実行されたかを確認することもできます。

データベースサーバによるローカルファイルの読み込や書き込みが許可されている場合、ClientFileValidator 接続パラメータを使用して制御します。

USING VALUE clause

この句は、CHAR 型、NCHAR 型、BINARY 型、または LONG BINARY 型の式や、BLOB 文字列からデータをロードするときに使用します。次は、この句を使用する方法の例です。

- 次の構文は、xp_read_file システムプロシージャを使用して、対象ファイルからロードする値を取得します。

```
... USING VALUE xp_read_file( 'filename' )...
```

- 次の構文は、値を直接指定して 2 つのローを挿入します。値はそれぞれ 4 と 5 です。

```
... USING VALUE '4¥n5'...
```

- 次の構文は、READ_CLIENT_FILE 関数の結果を値として使用します。

```
... USING VALUE READ_CLIENT_FILE( client-filename-expression )
```

この場合、セマンティック上同じであるため、`USING CLIENT FILE client-filename-expression` も指定できます。

LOAD TABLE 文で ENCODING 句を指定しない場合、`value-expression` が CHAR 型または BINARY 型のときはデータベース文字セット (db_charset) が、`value-expression` が NCHAR 型のときは NCHAR データベース文字セット (nchar_charset) が `value-expression` のエンコードと見なされます。

USING COLUMN clause

この句は、別のテーブルの単一カラムからデータをロードするときに使用します。データベースサーバは、リカバリ時に LOAD TABLE...WITH CONTENT LOGGING 文をリプレイしてトランザクションログをリプレイする場合に、この句を使用します。LOAD TABLE...WITH CONTENT LOGGING 文のトランザクションログレコードは、元の入力ファイルのまとまりで構成されます。リカバリ時にデータベースサーバがトランザクションログ内でこうしたまとまりを見つけた場合は、このまとまりがテンポラリテーブルにロードされてから、元のロード操作のデータがすべてロードされます。

USING COLUMN 句では次の句が必要です。

table-name

データのロード元となるカラムを含むベーステーブルまたはテンポラリテーブルの名前。リカバリ時にデータベースサーバがトランザクションログから使用する場合、解析されてロードされるローのまとまりを保持するのはこのテーブルです。

column-name

ロードするローのまとまりを保持する `table-name` 内のカラム名。

column-list

送信先テーブルの 1 つまたは複数のカラムであり、データのロード前にローをソートするために使用。`column-list` は、プライマリキーやカラムリスト内に含まれる NULL 入力不可のカラム上に一意のインデックスなどの検証可能な形で一意の値のセットにする必要があります。

load-option clause

さまざまなロードオプションを使用して、データのロード方法を制御できます。次のリストは、サポートされているロードオプションを示します。

ALLOW (integer | ALL | NO) ERRORS clause

この句は各文に 1 回しか指定できません。この句のデフォルト値は 0 で、違反が発生するとエラーを生成し、文はロールバックされます。整数 n を指定した場合、エラー n+1 でデータベースサーバが文をロールバックします。値 ALLOW NO ERRORS と ALLOW 0 ERRORS は同義です。この句を使用すると、データベースサーバは問題のあるデータを排除して、ロード操作を進めます。

データベースサーバは、ユーザが遭遇した最後のエラーをレポートします。このエラーは MESSAGE LOG にも記録されます。ROW LOG に書き込まれるローを変更したり、後続の LOAD TABLE 文の入力として使用したりできます。

ROW LOG がデータベースサーバまたはクライアントファイルに書き込まれる場合、その内容は元の入力ファイルと同じ文字セットで書き込まれます。MESSAGE LOG がサーバまたはクライアントファイルに書き込まれる場合、その内容はクライアントの言語で、クライアント接続時に CHAR で指定した文字セットで書き込まれます。ROW または MESSAGE LOG が CHAR 変数または NCHAR 変数に書き込まれる場合、それぞれ CHAR または NCHAR 文字セットで書き込まれます。

BYTE ORDER MARK clause

この句は、データの先頭でサーバがバイトオーダーマーク (BOM) を検索して解釈するかどうかを指定するときに使用します。デフォルトでは、このオプションは ON になります。BYTE ORDER MARK が OFF の場合、サーバは BOM を検索しません。

ENCODING 句が指定されている場合

- BYTE ORDER MARK オプションが ON のときに、UTF-16BE または UTF-16LE などのエンディアンを持つ UTF-16 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの検証に使用されます。間違ったエンディアンを指定すると、エラーが返されます。
- BYTE ORDER MARK オプションが ON のときに、明示的なエンディアンのない UTF-16 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの判別に使用されます。BOM がない場合は、オペレーティングシステムのエンディアンであると想定されます。
- BYTE ORDER MARK オプションが ON のときに UTF-8 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は無視されます。

ENCODING 句が指定されていない場合

- ENCODING 句を指定しないで BYTE ORDER MARK オプションを ON にすると、サーバは入力データの先頭で BOM を検索します。BOM が見つかると、BOM のエンコード (UTF-16BE、UTF-16LE、または UTF-8) に基づいてソースエンコードが自動的に選択され、BOM はロードするデータの一部であるとは見なされなくなります。BOM が見つからなかった場合、データベースの CHAR エンコードが使用されます。
- ENCODING 句を指定せず、BYTE ORDER MARK オプションが OFF である場合、データベースの CHAR エンコーディングが使用され、データベースサーバはデータの先頭で BOM を検索したり、解釈したりしません。

CHECK CONSTRAINTS clause

この句は、ロード時に制約をチェックするかどうかを制御するときに使用します。CHECK CONSTRAINTS はデフォルトで ON ですが、アンロードユーティリティ (dbunload) は CHECK CONSTRAINTS を OFF に設定して LOAD TABLE 文を書き出します。CHECK CONSTRAINTS を OFF に設定すると、データベースの再構築などの場合に役立つ検査制約が無効になります。まだ作成されていないユーザ定義関数を呼び出す検査制約がテーブルにあると、CHECK CONSTRAINTS が OFF に設定されていない場合、再構築は失敗します。

COMMENTS INTRODUCED BY clause

この句は、データファイルに使用する文字列を指定して、コメントを導入するときに使用します。使用すると、LOAD TABLE は文字列 comment-prefix で始まる行を無視します。

コメントは新しい行の先頭にのみ指定できます。

COMMENTS INTRODUCED BY を省略する場合、コメントがデータとして解釈されるため、データファイルにはコメントを含めないでください。

COMPRESSED clause

ロードするデータが入力ファイル内で圧縮されている場合に COMPRESSED を指定します。データベースサーバはデータを解凍してからロードします。データが圧縮されていないときに COMPRESSED を指定した場合は、LOAD が失敗してエラーが返されます。

データベースサーバで入力ファイル内のデータが圧縮されているかどうかを判断できるようにするには、AUTO COMPRESSED を指定します。圧縮されていた場合、データベースサーバはデータを解凍してからロードします。

入力ファイル内のデータが圧縮されていないことを指定するには、NOT COMPRESSED を指定します。データが圧縮されているときにデータベースサーバでそのデータを解凍しない場合にも、NOT COMPRESSED を指定できます。この場合、データは圧縮された状態でデータベースに残ります。ただし、ファイルが暗号化されて圧縮されている場合、NOT COMPRESSED なしで NOT ENCRYPTED を使用することはできません。

COMPUTES clause

デフォルトでは、このオプションは ON です。この場合、計算カラムの再計算が有効になります。COMPUTES を OFF に設定すると、計算カラムの再計算が無効になります。COMPUTES OFF は、たとえば、データベースを再構築するとき、まだ作成されていないユーザ定義関数を呼び出す計算カラムがテーブルの中にある場合に役立ちます。この場合は、COMPUTES を OFF に設定しないと再構築に失敗します。

アンロードユーティリティ (dbunload) は、COMPUTES を OFF に設定して LOAD TABLE 文を書き出します。

DEFAULTS clause

デフォルトでは、DEFAULTS は OFF に設定されています。DEFAULTS が OFF の場合は、カラムリストにないカラムすべてに NULL が割り当てられます。DEFAULTS が OFF に設定されており、NULL 入力不可のカラムがカラムリストから省かれている場合は、データベースサーバは、空の文字列をカラムの型に変換しようとします。DEFAULTS が ON に設定されており、カラムにデフォルト値が入っている場合は、その値が使用されます。

DELIMITED BY clause

この句は、カラムデリミタ文字を指定するときに使用します。デフォルトのカラムデリミタ文字列はカンマです。ただし、最長で 255 バイトの文字列を指定できます。たとえば、`... DELIMITED BY '###' ...` などです。指定するデリミタは文字列を用い、引用符で囲む必要があります。タブで区切った値を指定する場合、タブ文字を表す 16 進のエスケープシーケンス (9) を使用して、`... DELIMITED BY '¥x09' ...` のように指定します。

ENCODING clause

この句は、データベースにロードするデータに使用する文字エンコードを指定するときに使用します。ENCODING 句は、BCP フォーマットでは使用できません。

変換エラーがロード操作時に発生した場合、on_charset_conversion_failure オプションの設定に基づいてレポートされます。

データ内のバイトオーダーマークを解釈するには、BYTE ORDER 句を指定します。

ENCODING 句が指定されている場合

- BYTE ORDER MARK オプションが ON のときに、UTF-16BE または UTF-16LE などのエンディアンを持つ UTF-16 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの検証に使用されます。間違ったエンディアンを指定すると、エラーが返されます。
- BYTE ORDER MARK オプションが ON のときに、明示的なエンディアンのない UTF-16 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は、データのエンディアンの判別に使用されます。BOM がない場合は、オペレーティングシステムのエンディアンであると想定されます。
- BYTE ORDER MARK オプションが ON のときに UTF-8 エンコードを指定すると、データベースサーバはデータの先頭で BOM を検索します。BOM がある場合は無視されます。

ENCODING 句が指定されていない場合

- ENCODING 句を指定しないで BYTE ORDER MARK オプションを ON にすると、サーバは入力データの先頭で BOM を検索します。BOM が見つかると、BOM のエンコード (UTF-16BE、UTF-16LE、または UTF-8) に基づいてソースエンコードが自動的に選択され、BOM はロードするデータの一部であるとは見なされなくなります。
- ENCODING 句を指定せず、BYTE ORDER MARK オプションが OFF である場合、データベースの CHAR エンコーディングが使用され、データベースサーバはデータの先頭で BOM を検索したり、解釈したりしません。

ENCRYPTED clause

この句は、暗号化設定を指定するときに使用します。暗号化されたデータをロードする場合は、ENCRYPTED KEY に続けて、入力ファイル内のデータ暗号化に使用するキーを指定します。このキーには文字列または変数名を使用できます。

入力ファイル内のデータが暗号化されていないことを指定するには、NOT ENCRYPTED を指定します。データが暗号化されているときにデータベースサーバでそのデータを復号しない場合にも、NOT ENCRYPTED を指定できます。この場合、データは暗号化された状態でデータベースに残ります。ただし、ファイルが暗号化されて圧縮されている場合、NOT COMPRESSED なしで NOT ENCRYPTED を使用することはできません。

ESCAPE CHARACTER clause

この句は、データで使用するエスケープ文字を指定するときに使用します。16 進のコードおよび記号として格納されている文字に使用するデフォルトのエスケープ文字は、円記号 (¥) です。たとえば、¥x0A は改行文字です。エスケープ文字は、ESCAPE CHARACTER 句を使って変更することができます。たとえば、感嘆符 (!) をエスケープ文字として使用するには、次のように入力します。

```
ESCAPE CHARACTER '!'
```

エスケープ文字に指定する文字列は 1 マルチバイト文字を超えないようにすることをおすすめします。

ESCAPES clause

この句は、エスケープ文字を認識するかどうかを制御するときに使用します。ESCAPES を ON (デフォルト) にすると、データベースサーバによってエスケープ文字 (デフォルトでは ¥) に続く文字が認識され、特殊文字として解釈されます。改行文字は ¥n という組み合わせとしてインクルードされ、他の文字はタブ文字の ¥x09 のような 16 進の ASCII のコードとしてデータにインクルードされます。2 つの円記号 (¥) は 1 つの円記号として解釈されます。円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字と解釈されます。たとえば、¥q であれば、円記号と q が挿入されます。エスケープ文字に指定する文字列は 1 マルチバイト文字を超えないようにすることをおすすめします。

FORMAT TEXT

FORMAT TEXT (デフォルト) を選択すると、(ENCODING オプションで定義したとおりに) 入力行は文字と見なされ、1 行あたり 1 つのローで構成され、カラムデリミタ文字列によって値が区切られます。

FORMAT BCP

FORMAT BCP を指定すると、Adaptive Server Enterprise 生成の BCP アウトファイルを読み込むことができます。

FORMAT SHAPEFILE

FORMAT SHAPEFILE を指定して、ESRI シェイプファイルをロードします。シェイプファイルは、データベースサーバコンピュータ上にある必要があり、FROM `filename-expression` または USING FILE `filename-expression` を使用してロードする必要があります。ここで、`filename-expression` は、ファイル拡張子が `.shp` の ESRI シェイプファイルを示します。関連付けられている `.shx` ファイルと `.dbf` ファイルは、`.shp` ファイルと同じディレクトリに配置し、ベースファイル名を同じ名前にしてください。

FORMAT SHAPEFILE については、ENCODING 句を指定しないと、エンコードはデフォルトで ISO-8859-1 になります。

FORMAT SHAPEFILE を指定する場合は、次のロードオプションを使用できます。

- CHECK CONSTRAINTS
- COMPUTES
- DEFAULTS
- ENCODING
- ORDER
- PCTFREE
- WITH CHECKPOINT
- WITH ....LOGGING

LOAD TABLE 文はロードしている 2 番目のカラムタイプから SRID を取得します。たとえば、2 番目のカラムをタイプ ST_Geometry(SRID=4326) で作成した場合、ジオメトリは SRID 0 を使用してロードされます。2 番目のカラムのタイプが ST_Geometry (明示的な SRID でない) の場合、ジオメトリは SRID 0 を使用してロードされます。

FORMAT XML

FORMAT XML を指定する場合は、次のロードオプションを使用できます。

- BYTE ORDER MARK
- CHECK CONSTRAINTS
- COMPRESSED
- COMPUTES
- DEFAULTS
- ENCODING
- ENCRYPTED
- ORDER
- PCTFREE
- WITH CHECKPOINT
- WITH...LOGGING

FORMAT XML を使用する場合、入力ファイルは OPENXML 演算子を使用するクエリと同じように解析されます。SQL 文の引数は、システムプロシージャのパラメータと次のように対応します。

LOAD TABLE 文の句	OPENXML 演算子引数	詳細
`row-xpath`	`xpath`	

LOAD TABLE 文の句	OPENXML 演算子引数	詳細
-	flags	OPENXML の flags 引数に対応する FORMAT XML で値を指定する方法はありません。
NAMESPACES	namespaces	

FORMAT XML 句では、次のパラメータが使用されます。

row-xpath

XPath クエリを含む文字列または変数。XPath を使用すると、問い合わせる XML ドキュメントの構造を記述するパターンを指定できます。この引数にある XPath パターンによって、XML ドキュメントからノードが選択されます。row-xpath 引数の XPath クエリに一致する各ノードが、テーブルにローを 1 つずつ生成します。

FORMAT XML 句の row-xpath 引数に指定できるのは、メタプロパティのみです。メタプロパティは、属性のように XPath クエリ内でアクセスされます。namespaces が指定されていない場合、デフォルトでプレフィクス mp が Uniform Resource Identifier (URI) urn:sap-com:sa-xpath-metaprop にバインドされます。namespace が指定された場合、この URI は、クエリのメタプロパティにアクセスするために mp または他のプレフィクスにバインドされます。メタプロパティ名の大文字と小文字は区別されます。次のメタプロパティがサポートされています。

@mp:id

XML ドキュメント内のユニークなノードの ID を返します。データベースサーバが再起動されると、ドキュメント内の指定されたノードの ID が変更される場合もあります。このメタプロパティの値は、ドキュメントの順序で増えていきます。

@mp:localname

ノードの名前のローカル部分を返します。ノードに名前がない場合は NULL を返します。

@mp:prefix

ノードの名前のプレフィクス部分を返します。ノードに名前がない場合または名前にプレフィクスが付いていない場合は NULL を返します。

@mp:namespaceuri

ノードが含まれているネームスペースの URI を返します。ノードがネームスペースに含まれていない場合は NULL を返します。

@mp:xmltext

XML ドキュメントのサブツリーを XML 形式で返します。たとえば、内部ノードを照合する場合、このメタプロパティを使用して、下位のテキストノードの連結値ではなく、XML 文字列を返します。

column-xpath

結果セットのスキーマと、結果セット内で各カラムに値がどのように格納されるかを指定する文字列または変数。FORMAT XML 句の式が複数のノードと一致した場合、ドキュメントの順序における最初のノードのみが使用されます。ノードがテキストノードではない場合、テキストノードの下位ノードをすべて追加することにより結果を検索します。FORMAT XML 句の式がどのノードにも一致しない場合は、そのローのカラムは NULL になります。

namespace

XML ドキュメントを含む文字列または変数。クエリのスコープ内のネームスペースは、ドキュメントのルート要素から取得されます。

HEXADECIMAL clause

この句は、バイナリ値を 16 進数値として読み込むかどうかを指定するときに使用します。デフォルトでは、HEXADECIMAL は ON です。HEXADECIMAL ON の場合、バイナリカラム値は **0x**nnnnnn... として読み込まれます。ここで、0x は 1 つのゼロの後に 1 つの x が続きます。n はそれぞれ 16 進数の数字です。マルチバイト文字セットを扱う場合は、HEXADECIMAL ON を使用することが重要です。

HEXADECIMAL 句は、FORMAT TEXT 句を指定した場合にのみ使用できます。

MESSAGE LOG clause

この句は各文に 1 回しか指定できません。ローの挿入または解析中にエラーが発生すると、データベースサーバは、指定された場所にエラーを書き込みます。

ORDER clause

この句は、ロード時にデータをソートするかどうかを指定するときに使用します。ORDER のデフォルトは ON です。ORDER が ON に設定され、クラスタドインデックスが宣言されている場合、LOAD TABLE はクラスタドインデックスに従って入力データをソートし、同じ順序でローを挿入します。ロードするデータがすでにソート済みの場合は、ORDER を OFF に設定してください。

PCTFREE clause

この句は、各テーブルページに確保する空き領域の割合を指定するときに使用します。この設定は、テーブルに対する永続的な設定を上書きしますが、それはロード中のみ実行されます。また、ロードされているデータだけが対象となります。値 percent-free-space は 0 ～ 100 の整数です。値 0 を指定すると、各ページに空き領域を残しません。各ページは完全にパックされます。大きい値を指定すると、各ローが単独でページに挿入されます。

QUOTE clause

QUOTE 句は TEXT データ専用です。string は文字列値を囲みます。デフォルトは一重引用符 (アポストロフィ) です。

QUOTES clause

この句は、文字列を引用符で囲むかどうかを指定するときに使用します。QUOTES を ON (デフォルト) に設定すると、LOAD TABLE 文は引用符文字で囲まれた文字列を探します。QUOTES 句が省略されている場合、引用符はアポストロフィ (一重引用符) または引用符 (二重引用符) のいずれかになり、文字列に最初に登場するこれらの文字がその文字列の引用符として扱われます。文字列の終わりには先頭にあるものと同じ引用符が必要です。

QUOTES を ON に設定すると、カラムデリミタ文字列をカラム値の中に入れることができます。また、引用符文字は値の一部とは見なされません。したがって、次の行はアドレスにカンマがあるかどうかに関係なく、3 つではなく 2 つの値として扱われます。また、アドレスを囲む引用符は、データベースへは挿入されません。

```
'123 High Street, Anytown',(715)398-2354
```

値の中に引用符文字を含めるには、QUOTES を ON に設定して、2 つの引用符を使用します。次の行は、第 3 カラムの中へ一重引用符文字である値を入れます。

```
'123 High Street, Anytown','(715)398-2354',''''
```

ROW DELIMITED BY clause

この句は、入力レコードの末尾を示す文字列を指定するときに使用します。デフォルトのデリミタ文字列は改行 (¥n) です。ただし、最長で 255 バイトの文字列を指定できます。たとえば、`ROW DELIMITED BY '###'` などです。タブで区切った値を指定する場合、タブ文字を表す 16 進のエスケープシーケンス (9) を使用して、`ROW DELIMITED BY '¥x09'` のように指定します。デリミタ文字列に ¥n が含まれる場合、¥r¥n または ¥n のどちらかに一致します。

ROW LOG clause

この句は各文に 1 回しか指定できません。ローの挿入または解析中にエラーが発生すると、データベースサーバはローをユーザにレポートするだけでなく、指定された場所に入力ローのイメージを書き込みます。

SKIP clause

この句は、ファイルの最初の数行を無視するかどうかを指定するときに使用します。`integer` 引数では、スキップする行数を指定します。たとえば、この句を使用して、カラム見出しを含む行をスキップできます。ローデリミタがデフォルト (改行) でない場合、引用符付きの文字列に埋め込まれたローデリミタがデータに含まれると、スキップが正しく動作しないことがあります。

STRIP clause

この句は、引用符で囲まれていない値を挿入する前に、この値の先行ブランクまたは後続ブランクを削除するかどうかを指定するときに使用します。STRIP オプションは次のオプションを受け入れます。

STRIP OFF

先行ブランクも後続ブランクも削除しません。

STRIP LTRIM

先行ブランクを削除します。

STRIP RTRIM

後続ブランクを削除します。

STRIP BOTH

先行ブランクと後続ブランクの両方を削除します。

STRIP 動作は QUOTES 句と結び付いています。QUOTES OFF を指定すると、STRIP OFF、STRIP LTRIM、STRIP RTRIM、STRIP BOTH は正確に文字どおりに機能します。QUOTES 句を指定しなかったり、QUOTES ON を指定したりすると、引用符の付いていない文字列は常に左トリムおよび右トリムされます (ただし、文字列が右トリムされないようにする場合は、STRIP OFF または STRIP LTRIM を指定します)。

WITH CHECKPOINT clause

この句は、チェックポイントを実行するかどうかを指定するときに使用します。デフォルト設定は OFF です。この句を ON に設定すると、文が正常に完了し、ロギングされた後にチェックポイントが発行されます。この句が ON に設定されているとき、チェックポイントを発行する前にデータベースの自動リカバリが必要な場合で、FILE NAME LOGGING を使用している場合には、リカバリを正しく完了するために、テーブルのロードに使用されるデータファイルが存在している必要があります。WITH CHECKPOINT ON を指定した後にリカバリが必要な場合には、リカバリはチェックポイントの後で始まるため、データファイルは不要です。

データベースが破損したため、バックアップを使用して現在のログファイルを適用する必要がある場合は、FILE NAME LOGGING を使用していれば、この句に何が指定されているかに関係なく、データファイルが必要となります。

LOAD TABLE 文が BEGIN PARALLEL WORK 文の内側にある場合、WITH CHECKPOINT ON 文はサポートされません。しかしながら、チェックポイントは BEGIN WORK PARALLEL 文の開始時に実行されます。

WITH { FILE NAME | ROW | CONTENT } LOGGING

この句は、ロード操作時にトランザクションログに記録される詳細レベルを制御するときに使用します。

ロギングのレベルを次に示します。

WITH FILE NAME LOGGING clause

WITH FILE NAME LOGGING 句を指定すると、LOAD TABLE 文のみがトランザクションログに記録されます。リカバリ時にトランザクションログを使用する場合に結果が矛盾しないようにするには、元のロード操作に使用したファイルが元のロケーションにあり、そのファイルに元のデータが含まれている必要があります。このレベルのロギングが、最もパフォーマンスが高いです。ただし、データベースがミラーリングを行っている場合や、同期によって参照されているテーブル上にある場合は、このレベルを使用しないでください。また、式またはクライアントファイルからロードする場合にも、このレベルは使用できません。

LOAD TABLE 文でロギングレベルを指定しない場合、次を指定したときは WITH ROW LOGGING がデフォルトのレベルになります。

- FROM `filename-expression`
- USING FILE `filename-expression`

WITH ROW LOGGING clause

WITH ROW LOGGING 句を指定すると、各ローのロードが INSERT 文としてトランザクションログに記録されます。FROM `filename-expression` または USING FILE `filename-expression.` を使用している場合、同期を行っているデータベースやデータベースミラーリングのデフォルトになっているデータベースには、このレベルのロギングをおすすめします。ただし、大量のデータをロードする場合は、このロギングタイプがパフォーマンスに影響したり、トランザクションログがかなり長くなったりする可能性があります。

非決定的な値がない場合は、WITH CONTENT LOGGING が最もパフォーマンスが高くなる可能性があります。

また、このレベルは、計算カラムや CURRENT TIMESTAMP のデフォルトなどの非決定的な値がロード先のテーブルに含まれている場合に最適です。

LOAD TABLE 文が BEGIN PARALLEL WORK 文の内側にある場合、この文はサポートされません。

WITH CONTENT LOGGING clause

WITH CONTENT LOGGING 句を指定すると、データベースサーバは入力ファイルをまとまりとしてトランザクションログにコピーします。これらのまとまりは、リカバリ時にトランザクションログからなど、後から入力ファイルのコピーに再構成できます。大量のデータをロードする場合、このロギングタイプはパフォーマンスへの影響が非常に少なく、強力にデータを保護できますが、トランザクションログが長くなります。このロギングレベルは、ミラーリングを行っているデータベース、または非決定的な値がないときに後でリカバリするために元のデータファイルを維持しない方が望ましい場合におすすめします。

データベースが同期を行っている場合は、WITH CONTENT LOGGING 句を使用できません。WITH CONTENT LOGGING 句はテーブルがクライアントファイルからロードされるときに必要です。

LOAD TABLE 文でロギングレベルを指定しない場合、次を指定したときは WITH CONTENT LOGGING がデフォルトのレベルになります。

- USING CLIENT FILE `client-filename-expression`
- USING VALUE `value-expression`
- USING COLUMN `column-expression`

同様に、プライマリサーバまたはルートサーバ上で LOAD TABLE 文を使用する場合、WITH CONTENT LOGGING がデフォルトとなります。

LOAD TABLE 文が BEGIN PARALLEL WORK 文の内側にある場合、この文はサポートされません。

statistics-limitation-option

LOAD TABLE の実行中に統計が生成されるカラムを制限できます。制限しないと、すべてのカラムに対して統計が生成されます。統計が一部のカラムで使用されないことが確かな場合にのみ、この句を使用してください。ON ALL COLUMNS (デフォルト)、OFF、ON KEY COLUMNS、または統計を生成するカラムのリストを指定できます。

備考

LOAD TABLE を使うと、ファイルからデータベーステーブルの中へ効率よく大量の挿入を行えます。LOAD TABLE は、Interactive SQL 文の INPUT より効率的です。

-im データベースサーバオプションを使用して非書き込みモードで実行すると、複数の LOAD TABLE 文を同時に実行できます。

BEGIN PARALLEL WORK 文を使用すると、LOAD TABLE 文のリストを同時に実行できます。LOAD TABLE 文が BEGIN PARALLEL WORK 文の内側にある場合、次のとおりとなります。

- コンテンツは、データベースへのすべての接続によって共有可能なベーステーブルまたはグローバルテンポラリテーブルにのみロードできます。
- 各 LOAD TABLE 文は、別々のターゲットテーブルを指定する必要があります。
- 接続に対して wait_for_commit データベースオプションが有効化されている場合、それを無効化してから BEGIN PARALLEL WORK 文を実行する必要があります。

LOAD TABLE は、排他スキーマロックを使用します。ベーステーブル、グローバルテンポラリテーブル、ローカルテンポラリテーブルの場合、コミットが実行されます。

即時テキストインデックスが構築されているテーブル、または即時ビューから参照されているテーブルで LOAD TABLE を使用しようとすると、ロードは失敗します。即時テキストインデックス以外のインデックスまたはマテリアライズドビューでこのようなことは発生しません。ただし、従属するインデックスやマテリアライズドビューのデータをトランケートしてから LOAD TABLE 文を実行し、その後、インデックスとビューをリフレッシュすることを強くおすすめします。

作成時に ON COMMIT DELETE ROWS が明示的またはデフォルトで指定されている場合、テンポラリテーブルには LOAD TABLE 文を使用しないでください。ただし、ON COMMIT PRESERVE ROWS または NOT TRANSACTIONAL が指定されている場合は、LOAD TABLE を使用できます。

FORMAT TEXT の場合、値を指定しないことが、NULL 値を指定することになります。たとえば、`1,,'Fred',` という値を含むファイルに 3 つの値が予想される場合、挿入される値は 1、NULL、Fred です。ファイルに `1,2,` が含まれる場合、値 1、2、NULL が挿入されます。空白のみで構成される値も、NULL 値と見なされます。ファイルに `1, ,'Fred',` が含まれる場合、値 1、NULL、Fred が挿入されます。他の値はすべて NULL 以外と見なされます。たとえば、'' (一重引用符が 2 つ続く) は空の文字列です。'NULL' は 4 文字の文字列です。

LOAD TABLE でロードされるカラムが NULL 値を許容しておらず、ファイル値が NULL の場合、数値カラムには値 0 (ゼロ)、文字カラムには空の文字列 ('') が指定されます。LOAD TABLE でロードされたカラムが NULL 値を許容し、ファイル値が NULL の場合、カラム値は (すべての型で) NULL になります。

テーブルに a、b、c の各列が含まれ、入力データに a、b、c が含まれているときに、LOAD 文がデータのロード先として a と b のみを指定した場合は、カラム c に次の値が挿入されます。

- DEFAULTS ON が指定されると、カラム c にデフォルト値が入っている場合は、デフォルト値が使用されます。
- カラム c が NULL 値を許容するときにデフォルトが定義されていない場合は、NULL が使用されます。
- カラム c が NULL を許容しないときにデフォルトが定義されていない場合は、カラムのデータ型に応じて、ゼロ (0) または空の文字列 ('') が使用されるか、またはエラーが返されます。

LOAD TABLE は、テーブルのカラムに関するヒストグラムを作成するために、データのロード時にカラム統計を取得します。ヒストグラムは、オプティマイザによって使用されます。次に、ロードとカラム統計に関する追加のヒントを示します。

- LOAD TABLE は、今後の使用のためにベーステーブルに関する統計情報を保存します。グローバルテンポラリテーブルに関する統計情報は保存しません。
- 以前にデータが含まれていた可能性のある空のテーブルにデータをロードする場合は、LOAD TABLE 文を実行する前にカラムの統計を削除します。
- カラムで LOAD TABLE が実行されたときにカラムの統計が存在すると、このカラムの統計は再計算されません。代わりに、既存の統計に新しいデータの統計が挿入されます。既存のカラム統計が古い情報である場合、カラムに新しいデータをロードした後も依然として古いデータのまま残ります。カラム統計が古いことが考えられる場合は、LOAD TABLE 文の実行の前または後に更新することを検討します。
- LOAD TABLE は、テーブルに 5 つ以上のローがある場合にのみ統計情報を追加します。テーブルのローの数が 5 つ以上の場合、ヒストグラムは次のように変更されます。

テーブル内の既存データの有無	ヒストグラムの有無	実行されるアクション
はい	はい	既存のヒストグラムに変更を統合する
はい	いいえ	ヒストグラムを構築しない
いいえ	はい	既存のヒストグラムに変更を統合する
いいえ	いいえ	新しいヒストグラムを作成する

- LOAD TABLE は、ロードされたローの 90% を超える値が NULL 値の場合、カラムの統計情報を生成しません。

変数にファイル名を割り当て、その変数名を LOAD TABLE 文で使用することによって、動的に構成されたファイル名を使用して、LOAD TABLE 文を実行できます。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

権限

必要な権限は、-gl サーバオプションによって決まります。

-gl オプションが ALL に設定されている場合、次のいずれかの条件に該当する必要があります。

- そのテーブルの所有者である
- そのテーブルに対する LOAD 権限を持っている
- LOAD ANY TABLE システム権限を持っている
- ALTER ANY TABLE システム権限を持っている

-gl オプションが DBA に設定されている場合は、LOAD ANY TABLE または ALTER ANY TABLE システム権限が必要です。

-gl オプションが NONE に設定されている場合、LOAD TABLE は使用できません。

ビューにロードする際には、次のいずれかの条件に該当する必要があります。

- LOAD ANY TABLE システム権限を持っている
- ビューの LOAD 権限と、ビューが依存しているベーステーブルの LOAD 権限を持っている

クライアントコンピュータのファイルからロードする際には、

- READ CLIENT FILE 権限も必要です。
- 読み込み元のディレクトリに対する読み込み権限が必要です。
- allow_read_client_file データベースオプションが有効になっている必要があります。
- READ_CLIENT_FILE 機能が有効になっている必要があります。

関連する動作

オートコミット。

挿入は、WITH ROW LOGGING 句が指定された場合を除き、ログファイルに記録されません。そのため、ロギングタイプによっては、エラーが発生した場合に、挿入されたローを回復できないことがあります。ローを回復する必要があり、WITH FILE NAME LOGGING を使用している場合は、元のファイルが必要です。また、Mobile Link クライアントとして使用されるデータベースまたは SQL Remote レプリケーションに関連するデータベースでは、WITH ROW LOGGING 句のない LOAD TABLE 文は使用しないでください。これは、これらのテクノロジでは、トランザクションログファイルを解析して変更がレプリケートされるためです。

LOAD TABLE 文は、テーブルに関連したトリガは起動しません。

チェックポイントは操作開始時に実行されます。WITH CHECKPOINT ON を指定すると、2 番目のチェックポイントが終了時に実行されます。

データを大量にロードすると、カラム統計が更新されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

テーブル myTable を作成する手順は、次のようになります。

```
CREATE TABLE myTable( a CHAR(100), let_me_default INT DEFAULT 1, c CHAR(100) );
```

次に、入力ファイル `c:¥temp¥input.txt` を作成して、以下のデータを含めます。

```
ignore_me, this_is_for_column_c, this_is_for_column_a
```

最後に、作成した `c:¥temp¥input.txt` から myTable にデータをロードします。

```
LOAD TABLE myTable ( filler(), c, a ) FROM 'c:¥¥temp¥¥input.txt' FORMAT TEXT
DEFAULTS ON;
```

コマンド `SELECT * FROM myTable` は、次の結果セットを返します。

a	let_me_default	c
this_is_for_column_a	1	this_is_for_column_c

次の例は、EXECUTE IMMEDIATE 文を使用して、動的に構成されるファイル名で LOAD TABLE 文を実行します。

```
CREATE PROCEDURE LoadData( IN from_file LONG VARCHAR )
BEGIN
    DECLARE path LONG VARCHAR;
    SET path = 'd:¥¥data¥¥' || from_file;
    LOAD TABLE MyTable FROM path;
END;
```

次の例は、UTF-8 エンコードのテーブルデータを架空のファイルから mytable にロードします。

```
LOAD TABLE mytable FROM 'c:¥¥temp¥¥mytable_data_in_utf8.dat' ENCODING 'UTF-8';
```

この例では、架空のファイル `c:¥temp¥input2.dat` の中の `//` で始まる行は無視されます。

```
LOAD TABLE GROUPO.Employees FROM 'c:¥¥temp¥¥input2.dat' COMMENTS INTRODUCED BY
'//'
```

関連情報

UNLOAD 文 [1376 ページ]

1.4.4.202 LOCK FEATURE 文

他の同時接続でデータベースサーバの機能が使用されないようにします。

構文

```
LOCK FEATURE feature-name { ON | OFF }
```

```
feature-name :
'synchronization schema'
| 'all'
```

パラメータ

feature-name

ロックまたはロック解除する機能の名前。接続でロックされているすべての機能をロック解除するには、all を指定します。

ON | OFF

他の接続で機能が使用されないようにするには、ON を指定します。接続で機能を使用できるようにするには、OFF を指定します。

備考

同じ接続に対して 1 つの機能を複数回ロックすることはできません。現在の接続によってロックされていない機能をロック解除する場合に、機能名として all を指定しないと、エラーが返されます。1 つの機能が 2 つ以上の接続によってロックされている場合に、その機能を他の接続で使用できるようにするには、事前にすべての接続によってその機能がロック解除される必要があります。接続によって作成された機能ロックは、その接続が切断されると、削除されます。機能ロックは、データベースサーバが停止すると、削除されます。

同期スキーマ機能がロックされている場合、他の接続では次の文を実行できません。

- START SYNCHRONIZATION SCHEMA CHANGE
- CREATE SYNCHRONIZATION SUBSCRIPTION
- DROP SYNCHRONIZATION SUBSCRIPTION
- ALTER SYNCHRONIZATION SUBSCRIPTION
- ALTER PUBLICATION

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、他の接続で同期スキーマ機能が使用されないようにします。

```
LOCK FEATURE 'synchronization schema' ON;
```

関連情報

LOCK TABLE 文 [1201 ページ]

1.4.4.203 LOCK MUTEX 文

事前定義されたミューテックスを使用して、ファイルやシステムプロシージャなどのリソースをロックします。

構文

```
LOCK MUTEX [ owner.]mutex-name
[ IN { SHARE | EXCLUSIVE } MODE ]
[ TIMEOUT num-milliseconds ]
```

パラメータ

owner

ミューテックスの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

mutex-name

ミューテックスの名前。mutex-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

IN { SHARE | EXCLUSIVE } MODE 句

この句は、ロックでリソースへの排他アクセスを提供するかどうか (EXCLUSIVE)、または他の接続でもリソースを使用できるかどうか (SHARE) を指定するときに使用します。IN...MODE 句を指定しない場合、EXCLUSIVE がデフォルト動作になります。

TIMEOUT 句

ロックを取得するまでに待機するミリ秒単位の時間 (0 より大きい)。TIMEOUT 句を指定しない場合、接続はロックを取得できるまで無制限に待機します。

number-milliseconds の指定には、変数 (`TIMEOUT @timeout-value` など) を使用できます。number-milliseconds に変数が設定され、その変数が NULL の場合、句を指定しないのと同じ動作になります。

備考

再帰 LOCK MUTEX 文が許可されています。ただし、接続スコープミューテックスのミューテックスを解放するには、同じ数のリリース (RELEASE MUTEX) が必要です。

接続が SHARE MODE で LOCK MUTEX 文を実行し、再度 EXCLUSIVE MODE で実行する場合、他の接続が SHARE MODE でミューテックスをロックしていれば、実行がブロックされることがあります。そうでない場合、ロックモードは排他的ロックに変更され、ロックが接続によって完全に解除されるまで維持されます。

トランザクションスコープミューテックス (作成時に SCOPE TRANSACTION 句が指定された場合) では、ミューテックスはトランザクションが終了するまで保持されます。接続スコープミューテックス (作成時に SCOPE CONNECTION 句が指定された場合) では、ミューテックスは RELEASE MUTEX 文が実行されるかトランザクションが終了するまで保持されます。

LOCK MUTEX 文は、テーブルとローのロックで使用されるデッドロック検出と同じ恩恵を受けることができます。

権限

UPDATE ANY MUTEX SEMAPHORE システム権限を持っているか、ミューテックスの所有者である必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文では、protect_my_cr_section ミューテックスを排他モードでロックします。

```
LOCK MUTEX protect_my_cr_section IN EXCLUSIVE MODE;
```

関連情報

CREATE MUTEX 文 [863 ページ]
DROP MUTEX 文 [1048 ページ]
RELEASE MUTEX 文 [1259 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.204 LOCK TABLE 文

同時に実行されている他のトランザクションがテーブルにアクセスしたり、テーブルを修正したりすることを防止します。

構文

```
LOCK TABLE table-name
[ WITH HOLD ]
IN { SHARE | EXCLUSIVE } MODE
```

パラメータ

table-name

テーブル名。テーブルとして、ビューではなくベーステーブルを指定してください。テンポラリテーブルのデータは現在の接続に固有のローカルなものであるため、グローバルテンポラリテーブルまたはローカルテンポラリテーブルでは、ロックは効力を持ちません。

WITH HOLD 句

この句は、接続が終了するまでテーブルをロックするときに指定します。この句を指定しない場合は、現在のトランザクションがコミットまたはロールバックされた時点でロックは解放されます。

IN SHARE MODE 句

この句は、テーブルの共有テーブルロックを取得して、他のトランザクションはテーブルを修正できないようにするが、読み込みアクセスは許可するときに指定します。トランザクションがテーブルに共有ロックをかけると、他のトランザクションが修正中のローにロックをかけていない場合に、テーブルのデータを変更できます。IN SHARE MODE 句が選択されている場合、個々のローの読み込みロックは取得されません。

IN EXCLUSIVE MODE 句

この句は、テーブルの排他テーブルロックを取得して、他のトランザクションがそのテーブルにアクセスできないようにするときに指定します。他のトランザクションは、クエリ、更新を含め、テーブルに対するどのようなアクションも実行できません。`LOCK TABLE...IN EXCLUSIVE MODE` などの文を使用してテーブルを排他的にロックすると、デフォルトの動作ではテーブルのローロックを取得しません。subsume_row_locks オプションを Off に設定すると、この動作を無効にできます。

備考

LOCK TABLE 文によって、現在の独立性レベルに関係なく、同時実行性をテーブルレベルで直接制御できます。

トランザクションの独立性レベルは、現在のトランザクションが要求を実行するときに設定されるロックを制御しますが、LOCK TABLE 文では、もっと明示的にテーブル内のローのロックを制御できます。

ビューには LOCK TABLE 文を実行できません。ただし、ベーステーブルに LOCK TABLE 文を実行すると、共有スキーマロックが作成され、このロックによって従属ビューがロックされます。共有スキーマロックは、トランザクションがコミットされるか、またはロールバックされるまで継続します。

権限

テーブルを SHARE モードにロックするには、テーブルの所有者であるか、テーブルに対する SELECT 権限を持っているか、または SELECT ANY TABLE システム権限を持っていることが必要です。

テーブルを EXCLUSIVE モードにロックするには、次のいずれかの条件に該当する必要があります。

- テーブル所有者である。
- そのテーブルに対する ALTER オブジェクトレベル権限を持っている。
- ALTER ANY TABLE システム権限を持っている。
- ALTER ANY OBJECT システム権限を持っている。

関連する動作

ロックされたテーブルへのアクセスが必要な他のトランザクションは、遅延またはブロックされます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在のトランザクションの実行中は、他のトランザクションに対して Customers テーブルの修正を禁止します。

```
LOCK TABLE GROUPO.Customers
IN SHARE MODE;
```

関連情報

SELECT 文 [1291 ページ]
sa_locks システムプロシージャ [1546 ページ]

1.4.4.205 LOOP 文

文リストの実行を繰り返します。

構文

```
 [ statement-label : ]
[ WHILE search-condition ] LOOP
     statement-list
END LOOP [ statement-label ]
```

備考

WHILE と LOOP 文は制御文です。これを使って、search-condition が TRUE と評価するかぎり、SQL 文のリストを繰り返し実行できます。LEAVE 文を使って、END LOOP の後に記述されている最初の文から実行を再開できます。

終了の statement-label を指定する場合は、開始の statement-label と一致させる必要があります。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

LOOP/END LOOP 文は、オプションの ANSI/ISO SQL 言語機能 P002、"Computational completeness" の一部です。ANSI/ISO SQL 標準では、WHILE DO/END WHILE 文は、言語機能 P002 の一部でもある別個の文です。ソフトウェアでサポートされている構文の組み合わせ WHILE search-condition LOOP は標準にありません。

Transact-SQL

LOOP は Transact-SQL ダイアレクトではサポートされていません。Transact-SQL ストアドプロシージャ内でのループは Transact-SQL WHILE 文で実行されます。

例

次のフラグメントの例は、プロシージャ内の WHILE ループを示しています。

```
...
SET i = 1;
WHILE i <= 10 LOOP
   INSERT INTO Counters( number ) VALUES ( i );
   SET i = i + 1;
END LOOP;
...
```

次のフラグメントの例は、プロシージャ内のラベル付き LOOP を示しています。

```
SET i = 1;
lbl:
LOOP
   INSERT
   INTO Counters( number )
   VALUES ( i );
   IF i >= 10 THEN
      LEAVE lbl;
   END IF;
   SET i = i + 1;
END LOOP lbl
```

関連情報

FOR 文 [1106 ページ]
CONTINUE 文 [778 ページ]
WHILE 文 [T-SQL] [1408 ページ]

1.4.4.206 MERGE 文

テーブル、ビュー、プロシージャの結果をテーブルまたはビューにマージします。

構文

```
MERGE
INTO target-object [ into-column-list ]
USING [ WITH AUTO NAME ] source-object
   ON merge-search-condition
merge-operation [...]
 [ OPTION( query-hint, ... ) ]
```

```
target-object :
[ userid.]target-table-name [ [ AS ] target-correlation-name ]
| [ userid.]target-view-name [ [ AS ] target-correlation-name ]
| ( select-statement ) [ AS ] target-correlation-name
```

```
source-object :
```

```
[ userid.]source-table-name [ [ AS ] source-correlation-name ] [ WITH ( table-
hints ) ]
| [ userid.]source-view-name [ [ AS ] source-correlation-name ]
| [ userid.]source-mat-view-name [ [ AS ] source-correlation-name ]
| ( select-statement ) [ AS ] source-correlation-name [  using-column-list ]
| procedure
```

```
procedure :
[ owner.]procedure-name ( procedure-syntax )
   [ WITH ( column-name data-type, ... ) ]
   [ [ AS ] source-correlation-name ]
```

```
merge-search-condition :
search-condition
| PRIMARY KEY
```

```
merge-operation :
WHEN MATCHED [ AND search-condition ] THEN match-action
| WHEN NOT MATCHED [ AND search-condition ] THEN not-match-action
```

```
match-action :
DELETE
| RAISERROR [ error-number ]
| SKIP
| UPDATE SET set-item, ...
| UPDATE [ DEFAULTS { ON | OFF } ]
```

```
not-match-action :
INSERT
| INSERT [ insert-column-list ] VALUES ( value, ... )
| RAISERROR [ error-number ]
| SKIP
```

```
set-item :
[target-correlation-name.]column-name = { expression | DEFAULT }
| [ owner-name.]target-table-name.column-name = { expression | DEFAULT }
```

```
insert-column-list :
( column-name, ... )
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

```
into-column-list :
( column-name, ... )
```

```
using-column-list :
( column-name, ... )
```

```
error-number: 17000 よりも大きい正の整数または変数
```

```
option-name :identifier
```

```
option-value :
```

```
hostvar （許可されたインジケータ）
| string
| identifier
| number
```

```
table-hints: FROM 句のマニュアルを参照
```

```
search-condition: 検索条件のマニュアルを参照
```

```
set-clause-list: SET 文のマニュアルを参照
```

パラメータ

INTO 句

この句は、MERGE 文のターゲットオブジェクトを定義するために使用します。target-object は、ベーステーブル、通常のビュー、または派生テーブルの名前にすることができますが、マテリアライズドビューの名前にすることはできません。派生テーブルまたはビューは、更新可能なクエリブロックである必要があります。たとえば、ビュー定義または派生テーブル定義に UNION、INTERSECT、EXCEPT、または GROUP BY が含まれている場合、MERGE 文のターゲットとなるオブジェクトとしては使用できません。

target-object が派生テーブルの場合、オプションの into-column-list を使用すると、派生テーブルのカラムに代替名を指定できます。このように使用する場合は、into-column-list のサイズが派生テーブルのカラムリストと一致し、2 つのリストの順序も同じである必要があります。

target-object がベーステーブルまたはビューの場合、into-column-list を使用すると、MERGE 文が終わるまでテーブルのカラムまたはビューのカラムのサブセットを関連として指定できます。

データベースサーバは into-column-list を使用して、次の項目を解析します。

- WHEN MATCHED 句の中に SET 句のない UPDATE
- WHEN NOT MATCHED 句の中に VALUES 句のない INSERT
- ON 句の中の PRIMARY KEY 探索条件
- USING 句の中の WITH AUTO NAME 句

into-column-list を指定しない場合は、into-column-list には target-object のすべてのカラムが含まれると見なされます。

USING 句

この句は、マージ元のデータソースを定義するために使用します。source-object には、ベーステーブル (テーブルヒントを含む)、ビュー、マテリアライズドビュー、派生テーブル、またはプロシージャを指定できます。source-object が派生テーブルの場合は、using-column-list を指定できます。using-column-list を指定しない場合は、source-object のすべてのカラムが使用されます。

WITH AUTO NAME 句

この句は、target-object の into-column-list カラム内のカラムに一致するカラム名を、サーバがマージ操作で自動的に使用するように指定するときに使用します。次の例は同等であり、WITH AUTO NAME を指定すると、into-column-list 内のカラムの順序が、source-object 内のカラム名に一致するように変更されることを示します。

```
... INTO T ( Name, ID, Description )
   USING WITH AUTO NAME ( SELECT Description, Name, ID FROM Products WHERE
Description LIKE '%cap%')
... INTO T ( Description, Name, ID )
   USING ( SELECT Description, Name, ID FROM Products WHERE Description LIKE
'%cap%' )
```

ON 句

この句は、source-object 内のローを target-object 内のローに一致させるための条件を指定するときに使用します。

target-object のプライマリキー定義に基づいて source-object のローを一致させるには、ON PRIMARY KEY を指定します。source-object にプライマリキーは必要ありません。ただし、target-object にはプライマリキーが必要です。ON PRIMARY KEY を指定すると、次のようになります。

- target-object がベーステーブルでない場合、またはプライマリキーがない場合はエラーが返されます。
- into-column-list 内に 1 つ以上のプライマリキーカラムが含まれていない場合はエラーが返されます。
- into-column-list 内のすべてのプライマリキーカラムが using-column-list 内の対応するカラムと一致している場合、into-column-list と using-column-list のカラム数が異なる可能性があります。たとえば、into-column-list が (I1, I2, I3)、using-column-list が (U1, U2)、プライマリキーカラムが (I2, I3) の場合、target-object のプライマリキーカラム (I3) が using-column-list で一致しないため、エラーが返されます。
- プライマリキーの定義に関係なく、into-column-list のプライマリキーカラムと using-column-list の式の一致は、into-column-list 内のプライマリキーカラムの位置に左右されます。たとえば、target-object のプライマリキーが (B, C) と定義されており、into-column-list が (E, C, F, A, D, B) であると想定します。ON PRIMARY KEY を指定すると、target-object のカラム B は into-column-list の 6 番目の位置にあるため、using-column-list の 6 番目の要素と比較されます。同様に、target-object のカラム C は using-column-list の 2 番目の要素と比較されます。
  ON PRIMARY KEY は、対応する ON 条件の構文上の省略形です。たとえば、into-column-list が (I1, I2, .. In)、対応する一致した using-column-list が (U1, U2, .. Um) であると想定します。また、target-object のプライマリキーカラムが I1, I2, I3 であり、プライマリキーカラムがすべて into-column-list に含まれているとします。この場合、merge-search-condition は、連結された "I1=U1 AND I2=U2 AND I3=U3" と定義されます。

WHEN MATCHED 句と WHEN NOT MATCHED 句

WHEN MATCHED 句と WHEN NOT MATCHED 句は、source-object のローが target-object のローに一致する場合、または一致しない場合に実行するアクションを定義するときに使用します。アクションは、THEN キーワードの後に指定します。追加の AND 句を指定すると、一致するローのサブセットまたは一致しないローのサブセットに実行するアクションを制御できます。

ON 句を使用すると、source-object のローを、一致するローと一致しないローに分類する方法が決まります。target-object 内の少なくとも 1 つのローについて ON 句が TRUE の場合、source-object 内のローは一致するローと見なされます。target-object 内のどのローも ON 句が TRUE でない場合、source-object のローは一致しないローと見なされます。複数の WHEN MATCHED 句と WHEN NOT MATCHED 句を使用すると、一致するローと一致しないローのセットが切断のサブセットに分割されます。各サブセットは WHEN 句によって処理されます。WHEN MATCHED 句と WHEN NOT MATCHED 句は、MERGE 文に記載されている順序で処理されます。

WHEN MATCHED 句または WHEN NOT MATCHED 句の AND 句で指定した探索条件によって、候補となるローを特定の句で処理するかどうかが決まります。AND 句なしで WHEN MATCHED 句または WHEN NOT MATCHED 句を指定すると、AND 句の探索条件は TRUE であると見なされます。あるローが複数句の AND 条件を満たす場合、そのローは MERGE 文で最初に記載されている句によって処理されます。

同じ target-object のローを WHEN MATCHED 句のいずれかが複数回処理した場合は、エラーが返されます。1 つの target-object のローは、source-object の 2 つの異なる入力ローに一致する場合、同じ WHEN MATCHED 句の同じサブセットに複数回割り当てられます。

次の例では、target-object の Products で ID 300 のローが source-object の SalesOrderItems の 111 のローに一致するため、エラーが返されます。すべての一致は、WHEN MATCHED THEN UPDATE 句に対応する同じサブセットに属しています。

```
MERGE INTO GROUPO.Products
   USING GROUPO.SalesOrderItems S
   ON S.ProductID = Products.ID
   WHEN MATCHED THEN UPDATE SET Products.Quantity = 20;
```

WHEN MATCHED: 一致するローの場合、match-action には次のいずれかのアクションを指定できます。

DELETE

DELETE は、target-object のローを削除するときに指定します。

RAISERROR

RAISERROR は、マージ操作を終了し、変更をロールバックしてエラーを返すときに指定します。RAISERROR を指定すると、データベースサーバはデフォルトで SQLSTATE 23510 および SQLCODE -1254 を返します。

必要に応じて、RAISERROR キーワードの後に error-number パラメータを指定し、返される SQLCODE をカスタマイズできます。カスタム SQLCODE には 17000 よりも大きい正の整数を指定してください。数または変数のいずれとしても指定できます。カスタム SQLCODE を指定すると、負の数が返されます。

たとえば、`WHEN MATCHED AND search-condition THEN RAISERROR 17001` を指定すると、WHEN 句の条件を満たすローが検出された場合、マージ操作は失敗し、変更がロールバックされ、SQLSTATE 23510 と SQLCODE -17001 を持つエラーが返されます。

SKIP

SKIP は、ローをスキップするときに指定します。何のアクションも行われません。

UPDATE

UPDATE SET は、set-item 値を使用してローを更新するときに指定します。set-item は単純な割り当て式であり、カラムは expression の値に設定されます。expression には制限がありません。また、DEFAULT を指定すると、カラムをカラム用に定義されているデフォルトに設定できます。

たとえば、`UPDATE SET target-column1=DEFAULT, target-column2=source-column2` によって、target-column1 はデフォルト値に設定され、target-column2 は source-object の source-column2 の修正ローと同じ値に設定されます。

SET 句を指定しない場合、SET 句は into-column-list と using-column-list によって定義されます。たとえば、into-column-list が (I1, I2, .. In)、using-column-list が (U1, U2, .. Un) である場合、SET 句には `"SET I1=U1 , I2=U2 , .. In=Un"` が使用されます。

WHEN NOT MATCHED: 一致しないローの場合、non-match-action には次のいずれかのアクションを指定できます。

INSERT

INSERT...VALUES は、指定した値を使用してローを挿入するときに指定します。VALUES 句なしで INSERT 句を指定すると、VALUES 句は `into-column-list` と `using-column-list` によって定義されます。たとえば、`into-column-list` が (I1, I2, .. In)、`using-column-list` が (U1, U2, .. Un) である場合、、VALUES 句のない INSERT は `INSERT (I1, I2, .. In) VALUES (U1, U2, .. Un)` と同義になります。

RAISERROR

RAISERROR は、マージ操作を終了し、変更をロールバックしてエラーを返すときに指定します。RAISERROR を指定すると、データベースサーバはデフォルトで SQLSTATE 23510 および SQLCODE -1254 を返します。必要に応じて、RAISERROR キーワードの後に `error-number` パラメータを指定し、返される SQLCODE をカスタマイズできます。カスタム SQLCODE には 17000 よりも大きい正の整数を指定してください。数または変数のいずれとしても指定できます。カスタム SQLCODE を指定すると、負の数が返されます。

たとえば、`WHEN NOT MATCHED AND search-condition THEN RAISERROR 17001` を指定すると、WHEN 句の条件を満たすローが検出された場合、マージ操作は失敗し、変更がロールバックされ、SQLSTATE 23510 と SQLCODE -17001 を持つエラーが返されます。

SKIP

SKIP は、ローをスキップするときに指定します。何のアクションも行われません。

OPTION 句

この句は、文を実行するためのヒントを指定するときに使用します。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- `option-name` = `option-value`.クエリテキスト内の `OPTION( isolation_level = ... )` の指定は、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。

備考

`source-object` 内のローは `target-object` 内のローと比較され、ON 句の条件を満たすかどうかに応じて一致または不一致となります。`merge-search-condition` が true になるローが少なくとも 1 つ `target-table` に存在する場合、`source-object` のローは一致と見なされます。AND 句で指定された探索条件に従い、一致するローと一致しないローは WHEN MATCHED 句と WHEN NOT MATCHED 句で定義されたアクションによってグループ分けされます。一致するアクションと一致しないアクションごとにローをグループ化することを分岐化と呼び、各グループを分岐と呼びます。

分岐化が完了すると、データベースは分岐のローに定義されたアクションの実行を開始します。分岐は、出現した順序で処理されます。これは、文の中に WHEN 句が出現する順序と一致します。分岐化中、`source-object` の複数のローに、`target-object` の同じローに定義されたアクションがある場合、マージ操作は失敗してエラーが返されます。このことによって、`target-object` の指定されたローでマージ操作が複数のアクションを実行することがなくなります。

分岐が処理されると、挿入、更新、および削除の各アクションがそれぞれ INSERT 文、UPDATE 文、および DELETE 文としてトランザクションログに記録されます。

権限

必要な権限は、指定されたオブジェクトとマージの結果としての次の操作に基づいて実行時に決定されます。

source-object または **target-object** の選択

オブジェクトの所有者であるか、SELECT ANY TABLE システム権限を持っているか、またはターゲットオブジェクトに対する SELECT 権限を持っていることが必要です。

target-object へのローの挿入

`target-object` の所有者であるか、INSERT ANY TABLE システム権限を持っているか、またはターゲットオブジェクトに対する INSERT 権限を持っていることが必要です。

target-object でのローの更新

`target-object` の所有者であるか、UPDATE ANY TABLE システム権限を持っているか、またはターゲットオブジェクトに対する UPDATE 権限を持っていることが必要です。

target-object からのローの削除

`target-object` の所有者であるか、DELETE ANY TABLE システム権限を持っているか、またはターゲットオブジェクトに対する DELETE 権限を持っていることが必要です。

MERGE 文で参照されるすべてのプロシージャに対する EXECUTE 権限。

関連する動作

`target-object` にトリガが定義されていると、そのトリガが起動されます。

標準

ANSI/ISO SQL 標準

MERGE 文は、ANSI/ISO SQL 標準の機能 F312 および F313 を構成します。ソフトウェアの MERGE 文は、ANSI/ISO SQL 標準の MERGE 文仕様に準拠しており、追加の拡張が行われています。MERGE 文のソフトウェア固有の拡張を次に示します。

- WHEN MATCHED 句の中の DELETE
- WHEN [NOT] MATCHED 句の中の RAISERROR
- WHEN [NOT] MATCHED 句の中の SKIP
- OPTION 句
- PRIMARY KEY 句
- DEFAULTS 句
- VALUES 句を指定しない INSERT 句
- WITH AUTO NAME 句
- SET 句を指定しない UPDATE 句

例

次の例は、派生テーブルのローを Products テーブルにマージします。この例は、既存の T シャツと同じ属性で、色、数量、および製品識別子の異なる新しい T シャツを追加します。ここでは、ID 番号 304 の製品が Products テーブルにすでに存在すると、ローは挿入されません。

```
MERGE INTO Products ( ID, Name, Description, Size, Color, Quantity, UnitPrice,
Photo )
   USING WITH AUTO NAME (
      SELECT 304 AS ID,
             'Purple' AS Color,
             100 AS Quantity,
             Name,
             Description,
             Size,
             UnitPrice,
             Photo
          FROM Products WHERE Products.ID = 300 ) AS DT
   ON PRIMARY KEY
   WHEN NOT MATCHED THEN INSERT;
```

関連情報

FROM 句 [1112 ページ]
探索条件 [53 ページ]
SET 文 [1325 ページ]
UPDATE 文 [1391 ページ]
INSERT 文 [1167 ページ]
DELETE 文 [1016 ページ]
SELECT 文 [1291 ページ]
探索条件 [53 ページ]

1.4.4.207 MESSAGE 文

メッセージを表示します。

構文

```
MESSAGE expression
[ TYPE { INFO | ACTION | WARNING | STATUS } ]
[ TO { CONSOLE
   | CLIENT [ FOR { CONNECTION conn-id-number [ IMMEDIATE ] | ALL } ]
   | [ EVENT | SYSTEM ] LOG }
   [ DEBUG ONLY ] ]
```

```
conn-id :integer
```

パラメータ

TYPE 句

この句は、メッセージタイプを指定します。クライアントアプリケーションでは、メッセージの処理方法を指定します。たとえば、TO CLIENT を指定すると、Interactive SQL は、次のロケーションにメッセージを表示します。

INFO

履歴タブをクリックします。INFO がデフォルトタイプです。

ACTION

OK ボタンのあるウィンドウ。

WARNING

OK ボタンのあるウィンドウ。

STATUS

履歴タブをクリックします。

TO 句

この句は、メッセージの送信先を指定します。

CONSOLE

データベースサーバメッセージウィンドウにメッセージを送信します。データベースサーバメッセージログファイルが指定されている場合は、このファイルにも送信します。CONSOLE がデフォルトです。

CLIENT

メッセージをクライアントアプリケーションに送信します。アプリケーションではメッセージの処理方法を決めます。また、この決定のベースとなる情報として TYPE を使用できます。

LOG

メッセージを -o オプションで指定されたデータベースサーバメッセージログファイルに送信します。EVENT または SYSTEM を指定すると、メッセージはデータベースサーバメッセージウィンドウとイベントログにも書き込まれます。データベースサーバメッセージログのメッセージは以下のように指定します。

i

タイプ INFO または STATUS のメッセージ。

w

タイプ WARNING のメッセージ。

e

タイプ ACTION のメッセージ。

FOR 句

TO CLIENT が指定されたメッセージの場合、この句はメッセージに関する通知を受信する接続を指定します。デフォルトでは、接続は次回 SQL 文または WAITFOR DELAY 文が実行されたときにメッセージを受信します。

CONNECTION conn-id-number

受信者の接続 ID 番号を指定します。IMMEDIATE を指定すると、SQL 文がいつ実行されるかに関係なく、接続はメッセージを数秒以内に受信します。

ALL

開いているすべての接続がメッセージを受信することを指定します。

DEBUG ONLY

この句を使用すると、デバッグメッセージがストアドプロシージャに追加されるかどうか、また debug_messages オプションの設定を変更することによってトリガを有効にするか無効にするかを制御できます。DEBUG ONLY が指定されている場合、MESSAGE 文は、debug_messages オプションが On に設定されている場合にのみ実行されます。

> **注記**
>
> debug_messages オプションを Off に設定している場合、DEBUG ONLY メッセージはパフォーマンスへの影響が小さいため、通常は運用システム上のストアドプロシージャに残すことができます。ただし、頻繁に実行する位置では控え目に使用してください。そうしないと、パフォーマンスがわずかに低下することがあります。

備考

MESSAG 文は、メッセージを表示します。メッセージには、任意の式を指定できます。句は、メッセージタイプおよびメッセージが表示される場所を指定します。

`expression` のサイズがデータベースページサイズを超えた場合、`expression` はデータベースページサイズに収まるようにトランケートされます。データベースに有効なページサイズをチェックするには、PageSize データベースプロパティに対するクエリを実行します (`SELECT DB_PROPERTY( 'PageSize' );`)。

MESSAGE...TO CLIENT 文を実行するプロシージャは、接続に関連している必要があります。

たとえば、次の例ではイベントが接続以外で発生しているため、ウィンドウは表示されません。

```
CREATE EVENT CheckIdleTime
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) > 100
HANDLER
BEGIN
   MESSAGE 'Idle server' TYPE WARNING TO CLIENT;
END;
```

ただし、次の例では、データベースサーバメッセージウィンドウにメッセージが書き込まれます。

```
CREATE EVENT CheckIdleTime2
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) > 100
HANDLER
BEGIN
   MESSAGE 'Idle server' TYPE WARNING TO CONSOLE;
END;
```

有効な式には、引用文字列または他の定数、変数、関数を含めることができます。

FOR 句を使用すると、アプリケーションでイベントを明示的にチェックしなくても、データベースサーバ上で検出されたイベントを別のアプリケーションに通知できます。FOR 句が使用されている場合、受信者は、SQL 文を次に実行するときにメッセージを受信します。受信者が現在 SQL 文を実行している場合は、文の完了時にメッセージが受信されます。実行されている文がストアドプロシージャ呼び出しである場合、メッセージは呼び出しが完了する前に受信されます。

アプリケーションがメッセージの送信直後に通知を必要とし、接続が SQL 文を実行していない場合は、複数の WAITFOR DELAY 文を同時に使用するのではなく、IMMEDIATE 句を使用してクライアント通知を実装します。

通常、宛先の接続がデータベースサーバ要求を実行していない場合でも、IMMEDIATE 句を使用して送信されたメッセージは 5 秒未満で配信されます。クライアント接続が 1 秒間に複数の要求を実行した場合、クライアント接続が非常に大きな BLOB データを受信した場合、またはクライアントのメッセージコールバックの実行に 1 秒以上かかった場合、メッセージの配信が遅れる可能性があります。また、複数の IMMEDIATE メッセージを 1 つの接続に 2 秒ごとに送信すると、メッセージの配信が遅れたり、エラーメッセージが生成されたりすることがあります。クライアント接続が切断されると、MESSAGE...IMMEDIATE 文が正常に配信されないことがあります。

IMMEDIATE 句のないメッセージが送信されるのは、クライアントが特定の要求または WAITFOR DELAY 文を実行した場合のみです。そのため、メッセージの配信時間は無制限になります。

IMMEDIATE 句には、SQL Anywhere 11 以降の DBLib、ODBC、または SQL Anywhere JDBC ドライバが必要です。IMMEDIATE 句は、非スレッド型の UNIX クライアントライブラリではサポートされません。IMMEDIATE 句をサポートしない宛先接続にメッセージが送信されると、エラーメッセージが生成されます。MESSAGE 文の実行元と同じ接続に IMMEDIATE メッセージが送信されると、エラーメッセージが生成されます。

```
MESSAGE 'Please disconnect' TYPE WARNING TO CLIENT
    FOR CONNECTION 16 IMMEDIATE;
```

MESSAGE...TO CLIENT 式は、2048 バイトにトランケートされる場合があります。IMMEDIATE 句が指定されて送信されたメッセージの場合、メッセージ式は、接続のパケットサイズと 2048 バイトのうちの小さい方にトランケートされる場合があります。

Embedded SQL と ODBC クライアントは、メッセージコールバック関数を介してメッセージを受信します。それぞれの場合に、これらの関数を登録してください。Embedded SQL では、メッセージのコールバックが DB_CALLBACK_MESSAGE パラメータを使用して db_register_a_callback に登録されます。ODBC では、メッセージのコールバックは、SA_REGISTER_MESSAGE_CALLBACK パラメータを使用して SQLSetConnectAttr に登録されます。

権限

FOR 句、TO EVENT LOG 句、TO SYSTEM LOG 句を含む MESSAGE 文を実行するには、SERVER OPERATOR システム権限を持っている必要があります。それ以外の場合、この文について権限は必要ありません。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次のプロシージャは、データベースサーバメッセージウィンドウにメッセージを表示します。

```
CREATE PROCEDURE message_text( )
BEGIN
MESSAGE 'The current date and time: ', Now( );
END;
```

次の文は、データベースサーバメッセージウィンドウに文字列 `The current date and time` を表示し、続けて現在の日付と時刻を表示します。

```
CALL message_text( );
```

関連情報

sa_conn_info システムプロシージャ [1464 ページ]
CREATE PROCEDURE 文 [891 ページ]
WAITFOR 文 [1403 ページ]

1.4.4.208 NOTIFY SEMAPHORE 文

セフォマに関連付けられたカウンタを増分します。

構文

```
NOTIFY SEMAPHORE [ owner.]semaphore-name
[ INCREMENT BY number ]
```

パラメータ

owner

セマフォの所有者。`owner` は、間接識別子 (`` `[@variable-name]` `` など) を使用することでも指定できます。

semaphore-name

セマフォの名前。`semaphore-name` は、間接識別子 (`` `[@variable-name]` `` など) を使用することでも指定できます。

INCREMENT BY 句

正の整数は、セフォマに関連付けられたカウンタに増分する数を示します。この句を指定しない場合、カウンタは 1 ずつ増えます。

number の指定には、変数 (INCREMENT BY @inc-number など) を使用できます。

number に NULL を設定するか、変数を設定し、その変数が NULL の場合、句を指定しないのと同じ動作になります。

備考

カウンタが 0 で、このセフォマで WAITFOR SEMAPHORE 文により接続がブロックされた場合、NOTIFY SEMAPHORE 文によって接続が通知されます。

セフォマを通知した接続が削除またはキャンセルされてもカウンタの増分は存続するため、ご使用のアプリケーションでこの状況を対処する必要があります。

権限

UPDATE ANY MUTEX SEMAPHORE システム権限を持っているか、セマフォの所有者である必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文では、license_counter セフォマのカウンタを 1 増分します。

```
NOTIFY SEMAPHORE license_counter INCREMENT BY 1;
```

関連情報

CREATE SEMAPHORE 文 [909 ページ]
DROP SEMAPHORE 文 [1059 ページ]
WAITFOR SEMAPHORE 文 [1405 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.209 NOTIFY TRACE EVENT 文

ユーザ定義トレースイベントをトレースセッションに記録します。

構文

```
NOTIFY TRACE EVENT trace-event-name ( [ param1 [ ,... ] ] )
```

パラメータ

trace-event-name

トレースイベント名は、ユーザ定義のトレースイベントの名前である必要があります。システム定義のトレースイベントの名前は使用できません。

param1

トレースイベントフィールドの値。

備考

この文は、指定されたトレースイベントを含むセッションを通知するのに使用します。トレースイベントがどのセッションでもトレースされていない場合、この文は機能せず、パラメータは評価されません (たとえば、ユーザ定義の関数に対する呼び出しによって)。

システム権限

NOTIFY ANY EVENT システム権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、現在の (架空の) イベントトレースセッション my_event に対するイベントのロギングを行います。

```
NOTIFY TRACE EVENT my_event( 1, 'Hello world' ); -- trigger user-defined trace
events
NOTIFY TRACE EVENT my_event( 2, 'Hello world 2' );
NOTIFY TRACE EVENT my_event( 3, 'Hello world 3' );
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.4.4.210 OPEN 文 [ESQL] [SP]

事前に宣言したカーソルを開き、データベースの情報にアクセスします。

構文

Embedded SQL

```
OPEN cursor-name
[ USING { DESCRIPTOR sqlda-name | hostvar, ... } ]
[ WITH HOLD ]
[ ISOLATION LEVEL isolation-level ]
[ BLOCK n ]
```

ストアドプロシージャ

```
OPEN cursor-name
[ WITH HOLD ]
[ ISOLATION LEVEL isolation-level ]
```

```
cursor-name : identifier | hostvar
```

```
sqlda-name : identifier
```

```
isolation-level : 0 | 1 | 2 | 3 | SNAPSHOT | STATEMENT SNAPSHOT | READONLY
STATEMENT SNAPSHOT
```

パラメータ

USING DESCRIPTOR 句

USING DESCRIPTOR 句を使用できるのは、Embedded SQL のみです。この句は、カーソルが宣言されている SELECT 文のプレースホルダバインド変数にバインドされるホスト変数を指定します。

OPEN...USING はストアドプロシージャでは使用できません。

WITH HOLD 句

デフォルトで、すべてのカーソルは現在のトランザクションの最後 (COMMIT または ROLLBACK) で自動的に閉じられます。オプションの WITH HOLD 句は、次に実行されるトランザクションのためにカーソルを開いたままにします。カーソルは現在の接続が終了するまで、または明示的な CLOSE 文が実行されるまで開いたままです。接続が終了すると、カーソルは自動的に閉じます。

COMMIT または ROLLBACK を実行すると、WITH HOLD カーソルの結果セットを構成するローを含め、接続で保持されている長期間のローロックがすべて解放されます。ただし、独立性レベル 1、2、3 で取得されたカーソル安定性ロックは、カーソルが存在するかぎり保持され、カーソルが閉じられるか、または接続が終了したときにのみ解放されます。

ROLLBACK 文が完了すると、WITH HOLD カーソルの内容とその範囲内での位置付けは予測できなくなり、保証されなくなります。ansi_close_cursors_on_rollback オプションを使用して、ROLLBACK 文によって WITH HOLD カーソルを自動的に閉じるかどうかを制御できます。

ISOLATION LEVEL 句

ISOLATION LEVEL 句を使用すると、isolation_level オプションの現在の設定とは異なる独立性レベルでカーソルを開くことができます。このカーソルのすべてのオペレーションを、オプション設定とは関係なく、指定した独立性レベルで実行できます。この句を指定しない場合、カーソルが開いている間のカーソルの独立性レベルは、カーソルを開いたときの isolation_level オプションの値です。

次の値がサポートされます。

- 0
- 1
- 2
- 3
- SNAPSHOT
- STATEMENT SNAPSHOT
- READONLY STATEMENT SNAPSHOT

カーソルは、最初のローの前に置かれます。

BLOCK 句

この句は、Embedded SQL でのみ使用されます。クライアントアプリケーションは一度に複数のローをフェッチできます。これはブロックフェッチ、プリフェッチ、またはマルチローフェッチと呼ばれます。BLOCK 句を指定すると、プリフェッチされるローの数を減らすことができます。OPEN で BLOCK 句を指定することは、各 FETCH で BLOCK 句を指定することと同じです。

備考

OPEN 文は、指定したカーソルを開きます。カーソルを事前に宣言しておきます。

カーソルタイプが、カーソルの基本となる文の特性と一致していない場合、OPEN 文によって SQL 警告が返されることがあります。

カーソルが CALL 文にあるとき、OPEN は最初の結果セット (INTO 句がない SELECT 文) が見つかるまでプロシージャを実行します。プロシージャが完了しても結果セットが見つからない場合は、SQLSTATE_PROCEDURE_COMPLETE 警告が設定されます。

ESQL の使用方法

OPEN 文が正常に実行された後で、SQLCA (SQLIOESTIMATE) の `sqlerrd`[3] フィールドに、クエリのすべてのローをフェッチするのに必要な入出力操作の数の推定値が格納されます。同様に、SQLCA (SQLCOUNT) の `sqlerrd`[2] フィールドに、カーソル内にある実際のロー数 (0 以上の値)、またはその推定値 (絶対値が推定値である負の数) が入ります。データベースサーバが、ローをカウントしなくてもこの値を計算できる場合、これは実際のローの数になります。ローの実際の数を常に返すようにデータベースを設定することもできますが、これはお奨めしません。

`cursor-name` を識別子または文字列によって指定する場合、C プログラムでは OPEN の前に対応する DECLARE CURSOR 文を置きます。`cursor-name` をホスト変数によって指定する場合、DECLARE CURSOR 文を OPEN 文の前で実行します。

権限

カーソルが SELECT 文上にある場合、カーソルで参照されるオブジェクトの所有者であるか、オブジェクトに対する SELECT 権限を持っているか、適切な SELECT システム権限 (たとえば SELECT ANY TABLE) を持っている必要があります。

カーソルが CALL 文上にある場合、プロシージャの所有者であるか、プロシージャに対する EXECUTE 権限を持っているか、EXECUTE ANY PROCEDURE システム権限を持っている必要があります。

関連する動作

OPEN 文を実行すると、INSENSITIVE カーソルの結果セットが完全に実体化されます。

アクセスプランのキャッシュが有効になっている場合、OPEN 時にアプリケーションに返される SQL 警告が抑制されることがあります。抑制される警告には、カーソルタイプが変更されたこと、基本となるクエリが決定的でないこと、または文に埋め込まれている複数のリテラル定数で文字列のトランケーションが発生していることを示す警告があります。

標準

ANSI/ISO SQL 標準

Embedded SQL 内での OPEN 文の使用は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部です。ストアドプロシージャ内での OPEN 文の使用は、のコア機能です。ISOLATION LEVEL 句と BLOCK 句は、

CALL 文でのカーソルの OPEN 機能と同様に、標準にありません。ANSI/ISO SQL 標準では、WITH HOLD は、DECLARE CURSOR 文の一部として指定し、OPEN では指定しません。

SQLCA の特定の値の設定は、標準にありません。

Transact-SQL

OPEN 文は Adaptive Server Enterprise でサポートされています。Adaptive Server Enterprise では、ISOLATION LEVEL 句、BLOCK 句、WITH HOLD 句はサポートされていません。

例

次の例は、Embedded SQL での OPEN の使用を示します。

```
EXEC SQL OPEN employee_cursor;
```

```
EXEC SQL PREPARE emp_stat FROM
'SELECT empnum, empname FROM GROUPO.Employees WHERE name like ?';
EXEC SQL DECLARE employee_cursor CURSOR FOR emp_stat;
EXEC SQL OPEN employee_cursor USING :pattern;
```

この例のフラグメントは、プロシージャまたはトリガ内の OPEN 文を示します。

```
BEGIN
   DECLARE cur_employee CURSOR FOR
   SELECT Surname
   FROM GROUPO.Employees;
   DECLARE name CHAR(40);
   OPEN cur_employee;
   LP: LOOP
         FETCH NEXT cur_employee INTO name;
         IF SQLCODE <> 0 THEN LEAVE LP END IF;
    ...
   END LOOP
   CLOSE cur_employee;
END
```

関連情報

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
RESUME 文 [1271 ページ]
PREPARE 文 [ESQL] [1238 ページ]
FETCH 文 [ESQL] [SP] [1101 ページ]
DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
RESUME 文 [1271 ページ]
CLOSE 文 [ESQL] [SP] [767 ページ]
FOR 文 [1106 ページ]

1.4.4.211 OUTPUT 文 [Interactive SQL]

現在のクエリ結果をファイルに出力します。

構文

ファイルへの出力

```
OUTPUT TO filename
[ APPEND ]
[ BYTE ORDER MARK { ON | OFF } ]
[ COLUMN WIDTHS ( integer, ...) ]
[ DELIMITED BY string ]
[ ENCODING encoding ]
[ ESCAPE CHARACTER character ]
[ ESCAPES { ON | OFF } ]
[ FORMAT output-format ]
[ HEXADECIMAL { ON | OFF | ASIS } ]
[ QUOTE string [ ALL ] ]
[ VERBOSE ]
[ WITH COLUMN NAMES ]
```

```
output-format :
TEXT
| FIXED
| EXCEL
| HTML
| SQL
| XML
```

```
encoding : string | identifier
```

ODBC データソースへの出力

```
OUTPUT
USING connection-string
INTO destination-table-name
[ CREATE TABLE { ON | OFF } ]
```

```
connection-string :
{ DSN = odbc-data-source
| DRIVER = odbc-driver-name [; connection-parameter = value [; ... ] ] }
```

パラメータ

APPEND 句

これはオプションのキーワードであり、既存の出力ファイルの前の内容を上書きしないで、末尾にクエリの結果を追加するために使用します。APPEND 句を使用しない場合、OUTPUT 文はデフォルトで出力ファイルの内容を上書きします。出力フォーマットが TEXT、FIXED、または SQL の場合に、APPEND キーワードが有効です。

BCP フォーマット句

BCP フォーマット句は、SQL Anywhere と Adaptive Server Enterprise 間でファイルのインポートとエクスポートを実行するために使用します。

BYTE ORDER MARK 句

この句は、Unicode ファイルの最初にバイトオーダーマーク (BOM) を追加するかどうかを指定するときに使用します。デフォルトでは、このオプションは ON です。この場合、ファイルの最初でバイトオーダーマーク (BOM) を書き込むように Interactive SQL に指示します。BYTE ORDER MARK が OFF の場合、DBISQL は BOM を書き込みません。

BYTE ORDER MARK 句は、TEXT フォーマットされたファイルに書き込む場合にのみ関係します。TEXT 以外の FORMAT 句とともに BYTE ORDER MARK 句を使用すると、エラーが返されます。

BYTE ORDER MARK 句は、UTF-8 か UTF-16 でエンコードされたファイルの読み込みまたは書き込みをするときにのみ使用されます。その他のエンコードで BYTE ORDER MARK 句を使用しようとすると、エラーが返されます。

COLUMN WIDTHS 句

COLUMN WIDTHS 句を使用して、FIXED フォーマット出力のカラム幅を指定します。

CREATE TABLE 句

CREATE TABLE 句を使用して、インポート先テーブルが存在しない場合に作成するかどうかを指定します。デフォルトは ON です。

DELIMITED BY 句

DELIMITED BY 句を使用できるのは、TEXT 出力フォーマットの場合のみです。カラムはデリミタ文字列で区切られます。デリミタ文字列は、isql_field_separator_option によって決まります。

デフォルトのデリミタは、少数点にピリオドを使用するロケールではカンマ、少数点にピリオドを使用するロケールではセミコロンです。

ENCODING 句

ENCODING 句では、ファイルの書き込みに使用されるエンコードを指定できます。ENCODING 句は、TEXT フォーマットでのみ使用できます。

ENCODING 句は、オペレーティングシステムの文字セットで表すことができないデータがある場合に役立ちます。この場合、ENCODING 句を使用しないと、デフォルトエンコードで表すことができない文字は出力から失われます (つまり、損失を伴う変換が発生します)。

OUTPUT 文を使用して入力ファイルを作成したときにエンコードを指定した場合は、INPUT 文でも同じ ENCODING 句を指定してください。

Interactive SQL を実行するとき、データのエクスポートに使用されるエンコードは次の順序で決定されます。

- ENCODING 句で指定されたエンコード (この句が指定されている場合)。
- default_isql_encoding オプションで指定されたエンコード (このオプションが設定されている場合)。
- 実行しているプラットフォームのデフォルトエンコード。英語版 Windows コンピュータでは、デフォルトエンコードは 1252 です。

ESCAPE CHARACTER 句

16 進コードと記号として格納されている文字に使用するデフォルトのエスケープ文字は円記号 (¥) です。たとえば、¥x0A は改行文字です。

この設定は、ESCAPE CHARACTER 句を使用して変更することができます。たとえば、感嘆符 (!) をエスケープ文字として使用するには、次のように指定します。

```
... ESCAPE CHARACTER '!'
```

改行文字は '¥n' と指定できます。他の文字は、タブ文字は ¥x09 などのように、16 進 ASCII コードを使用して指定できます。2 つの円記号 (¥) は 1 つの円記号として解釈されます。円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字と解釈されます。たとえば、¥q は円記号と q と解釈されます。

ESCAPES 句

ESCAPES を ON (デフォルト) にすると、データベースサーバによって円記号に続く文字が認識され、特殊文字として解釈されます。ESCAPES を OFF にすると、ソースデータに記載されているとおりに文字が書き込まれます。

FORMAT 句

FORMAT 句によって、出力のファイルフォーマットを指定できます。FORMAT 句を指定していない場合、output_format オプションで指定したフォーマットが使用されます。FORMAT 句を指定した場合、output_format オプションの設定は無視されます。デフォルトの出力フォーマットは TEXT です。使用できる出力フォーマットは、次のとおりです。

TEXT

出力は、ファイルの 1 行に 1 つのローが格納されたテキストフォーマットファイルです。すべての値を項目デリミタで区切り、文字列をアポストロフィ (一重引用符) で囲みます。項目デリミタは、通常コンマ (,) またはセミコロン (;) です。デリミタと引用符文字列は、DELIMITED BY 句と QUOTE 句を使って変更できます。

他の 3 つの特別なシーケンスも使用されます。2 文字の ¥n は改行文字を表し、¥¥ は ¥ を表し、¥xDD は 16 進コード DD の文字を表します。

出力で TEXT に引用符または改行が含まれないようにするには、次のように引用符とエスケープをオフにします。`QUOTE '' ESCAPES OFF`。

EXCEL

FORMAT EXCEL 句を使用して `.csv` または `.txt` の拡張子を持つファイルをエクスポートする場合、Microsoft Excel ファイルのデフォルトのフォーマットが使用されます。ファイル名に `.csv` または `.txt` のファイル拡張子がない場合は、`.xlsb` 拡張子を持つ Microsoft Excel ワークシートが出力されます。列ヘッダはシートの最初のローに書き込まれます。

Microsoft Excel ファイルにエクスポートするには、Interactive SQL とビット設定が一致する Microsoft Excel ODBC ドライバがインストールされている必要があります。

Microsoft Excel ワークブック (`.xl*`) ファイルへのエクスポートは、Windows でのみサポートされています。

FIXED

この出力は、それぞれのカラムが固定幅を持つ固定フォーマットです。各カラムの幅は COLUMN WIDTHS 句を使用して指定できます。カラムの見出しはこのフォーマットでは出力されません。

COLUMN WIDTHS 句を省略すると、各カラムの幅はデータ型から計算され、そのデータ型のどのような値でも十分に保持できる大きさになります。例外は、32 KB がデフォルトの LONG VARCHAR と LONG BINARY データです。

HTML

この出力は HTML (Hyper Text Markup Language) フォーマットです。

SQL

出力は、(テーブル内の情報を再作成するのに必要な) Interactive SQL の INPUT 文で、`.sql` ファイルに格納されます。

XML

この出力は、UTF-8 でエンコードされ、DTD が埋め込まれた XML ファイルです。バイナリ値は、2 桁の 16 進数文字列として表されるバイナリデータで CDATA ブロック内にエンコードされます。

HEXADECIMAL 句

HEXADECIMAL 句は、バイナリ値を TEXT フォーマットに出力する方法を指定します。使用できる値は、次のとおりです。

ON

ON に設定すると、0x プレフィクスとその後に続く一連の 16 進ペアでバイナリ値が書き込まれます。たとえば、0xabcd のようになります。

OFF

OFF に設定すると、印刷不能な文字値の先頭にエスケープ文字 (円記号など) が追加され、その後に x、バイトの 16 進ペアと続きます。印刷可能な文字はそのまま出力されます。

ASIS

ASIS に設定すると、値はそのまま書き込まれます。値が制御文字を含む場合も、エスケープはされません。ASIS は、タブや改行などのフォーマット記号を含むテキストに適しています。

QUOTE 句

QUOTE 句を使用できるのは、TEXT 出力フォーマットの場合のみです。文字列値は引用符で囲みます。デフォルトは一重引用符 (') です。ALL を QUOTE 句の中に指定する場合、引用符文字列を文字列だけではなくすべての値の周囲に配置します。引用符で囲まないようにするには、空の一重引用符を指定します。たとえば、`QUOTE ''` のように記述します。

USING 句

USING 句は ODBC データソースにデータをエクスポートします。DSN オプションを使用して ODBC データソース名を指定するか、または DRIVER オプションを使用して ODBC ドライバ名と接続パラメータを指定できます。`Connection-parameter` は、データベース特定の接続パラメータのオプションのリストです。

`odbc-data-source` は、ユーザ名または ODBC データソース名です。たとえば、SQL Anywhere サンプルデータベースの `odbc-data-source` は SQL Anywhere 17 Demo です。

`odbc-driver-name` は、ODBC ドライバ名です。SQL Anywhere データベースの場合、`odbc-driver-name` は SQL Anywhere です。Ultra Light データベースの場合、`odbc-driver-name` は Ultra Light 17 です。

VERBOSE 句

オプションの VERBOSE キーワードを指定すると、クエリに関するエラーメッセージ、データの選択に使用された SQL 文、データ自体が出力ファイルに書き込まれます。データを含まない行には 2 つのハイフン (--) のプレフィクスが付きます。VERBOSE を省略すると (デフォルト)、ファイルにはデータのみが書き込まれます。出力フォーマットが TEXT、FIXED、または SQL の場合に、VERBOSE キーワードが有効です。

WITH COLUMN NAMES 句

WITH COLUMN NAMES 句は、テキストファイルの先頭行にカラム名を挿入します。WITH COLUMN NAMES 句を使用できるのは、TEXT フォーマットの場合のみです。Microsoft Excel ファイルでは、列ヘッダはファイルの最初のローに書き込まれます。

WORKSHEET

FORMAT 句に EXCEL を設定すると、WORKSHEET 句によってデータをエクスポートする Microsoft Excel ファイル内のワークシートが指定されます。この句を省略すると、データは *Results* というデフォルトのワークシートにエクスポートされます。

備考

出力するデータを取り出す文の直後に OUTPUT 文を使用します。

OUTPUT 文の APPEND 句と VERBOSE 句は、Interactive SQL の以前のバージョンの演算子 >#、>>#、>&、>>& と同じです。これらの演算子はデータをリダイレクトしますが、Interactive SQL 文を使用すると、出力がより正確になり、コードが読み取りやすくなります。

実行された文から複数の結果セットが返された場合、Interactive SQL によって結果セットごとに 1 つのファイルが作成されます。ファイルは、`filename-x` という名前になります。ここで、`x` は 1 から始まるカウンタです。

この構文は、複数の結果セットをエクスポートするための ODBC データソースへの出力には使用できません。複数の結果セットは Microsoft Excel ワークブックにエクスポートできません。

出力フォーマットは、オプションの FORMAT 句を使って指定できます。デフォルトのフォーマットは TEXT です。FORMAT 句を指定しない場合、Interactive SQL output_format オプションの設定が使用されます。

INPUT 文は Interactive SQL 文なので、IF 文などの複合文、ストアドプロシージャ、サーバから実行される任意の文では使用できません。

Microsoft Excel では BINARY または LONG BINARY データはサポートしていないため、データをエクスポートする前に文字列または数値に変換する必要があります。

権限

なし。

関連する動作

Interactive SQL の場合、結果タブには現在のクエリの結果が表示されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

Employees テーブルの内容をテキストファイルに出力します。

```
SELECT * FROM GROUPO.Employees;
OUTPUT TO 'c:¥¥temp¥¥Employees.txt'
FORMAT TEXT;
```

Employees テーブルの内容を既存のテキストファイルの最後に追加し、クエリに関するメッセージも同じファイルに追加します。

```
SELECT * FROM GROUPO.Employees;
OUTPUT TO 'c:¥¥temp¥¥Employees.txt'
APPEND VERBOSE;
```

改行文字が埋め込まれた値をエクスポートするとします。改行文字の数値は 10 です。SQL 文ではこれを文字列 '¥x0a' として表すことができます。たとえば、HEXADECIMAL を ON に設定して次の文を実行します。

```
SELECT CAST ('line1¥x0aline2' AS VARBINARY);
OUTPUT TO 'c:¥¥temp¥¥file.txt' HEXADECIMAL ON;
```

次のテキストを含む 1 行のファイルができます。`0x6c696e65310a6c696e6532`。

同じ文を、HEXADECIMAL を OFF にして実行すると、次のようになります。`'line1¥x0Aline2'`。

最後に、HEXADECIMAL を ASIS に設定すると、2 行のファイルができます。

```
'line1
line2'
```

ASIS を使用した場合に 2 行になるのは、埋め込まれた改行文字が 2 桁の 16 進表現に変換されず、プレフィクスも付かないままエクスポートされたためです。

次の例は、Customers テーブルから新しいテーブル Customers2 にデータを出力します。

```
SELECT * FROM Customers;
OUTPUT USING 'DSN=SQL Anywhere 17 Demo;PWD=sql'
INTO "Customers2";
```

次の例は、DRIVER オプションを使用して、サンプルデータベースから架空のデータベース `mydatabase.db` に Customers テーブルをコピーします。

```
SELECT * FROM Customers;
OUTPUT USING 'DRIVER=SQL Anywhere 17;UID=DBA;PWD=passwd;DBF=c:¥¥test¥
¥mydatabase.db'
INTO "Customers";
```

次の例は、DRIVER オプションを使用して、SQL Anywhere サンプルデータベースから架空の Ultra Light データベース `myULDatabase.db` の Customers テーブルに Customers テーブルをコピーします。

```
SELECT * FROM Customers;
OUTPUT USING 'DRIVER=UltraLite 17;DBF=c:¥¥test¥¥myULDatabase.udb'
INTO "Customers";
```

次の例は、DRIVER オプションを使用して、架空の MySQL データベース `mydatabase` に Customers テーブルをコピーします。

```
SELECT * FROM GROUPO.Customers;
OUTPUT USING 'DRIVER=MySQL ODBC 5.1
Driver;DATABASE=mydatabase;SERVER=mySQLHost;UID=me;PWD=secret'
INTO "Customers";
```

次のコマンドは、`'one¥x0Atwo¥x0Athree'` を含むファイルを出力します。

```
SELECT 'one¥ntwo¥nthree';
```

```
OUTPUT TO 'c:¥¥temp¥¥test.txt' HEXADECIMAL OFF;
```

次の例は、Customers テーブルを customers.xlsb という Microsoft Excel ワークブックにエクスポートします。

```
SELECT * FROM Customers;
OUTPUT TO 'Customers.xlsb' FORMAT EXCEL
```

関連情報

SELECT 文 [1291 ページ]
INPUT 文 [Interactive SQL] [1160 ページ]
UNLOAD 文 [1376 ページ]

1.4.4.212 PARAMETERS 文 [Interactive SQL]

Interactive SQL スクリプトファイルにパラメータを指定します。

構文

```
PARAMETERS parameter1, parameter2, ...
```

備考

PARAMETERS 文はスクリプトファイルのパラメータに名前を付けて、スクリプトファイルの中で後から参照できるようにします。

パラメータを参照するには、指定したパラメータを置き換えるファイルの中に {parameter1} を配置します。波括弧とパラメータ名の間には、スペースを入れないでください。

Interactive SQL は、文を実行するとき、その中で使われている不明なパラメータの値を入力するよう要求します。PARAMETERS 文が存在するからといって、必ずしも不明なパラメータの値の入力が要求されるというわけではありません。

.SQL ファイルに含まれるリテラル文字列に波括弧が含まれているが、パラメータ名を囲んでいない場合、Interactive SQL は値の入力を要求しません。

.SQL ファイルで使用されていないパラメータが PARAMETERS 文で指定されている場合、Interactive SQL は値の入力を要求せず、エラーとして扱われることもありません。

権限

なし

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の Interactive SQL スクリプトファイルは、2 つのパラメータを取ります。

```
PARAMETERS department_id, file;
SELECT Surname
FROM GROUPO.Employees
WHERE DepartmentID = {department_id}
>#{file}.dat;
```

このスクリプトを `c:¥temp¥test.sql` という架空のファイルに保存すると、次のコマンドを使用して Interactive SQL から実行できます。

```
READ 'c:¥¥temp¥¥test.sql' [100] [data]
```

パラメータはリテラル文字列の中で使用することもできます。次の例では、結果として "Hello, World" が返されます。

```
-- Hello.sql
PARAMETERS yourName;
MESSAGE 'Hello, {yourName}' TO CLIENT;
READ Hello.sql [World]
```

次の文を実行すると、結果として `"Hello, World{end}"` が返されます。なぜなら Interactive SQL は "end" というパラメータの入力を要求しないからです。

```
-- Hello2.sql
PARAMETERS yourName;
MESSAGE 'Hello, {yourName}{end}' TO CLIENT;
```

デモデータベースには、とある架空の企業の部門に関する ID や名前などの情報を格納した Departments テーブルが含まれています。ID を指定して、特定の部門の名前を変更する `.SQL` ファイルを記述してみましょう。以下のようなスクリプトになるでしょう。

```
-- UpdateDepartmentName.sql
PARAMETERS id, name, headID;
UPDATE Departments D SET D.DepartmentName='{name}' where D.DepartmentID={id};
```

このスクリプトは、READ 文に 2 つのパラメータを指定して実行します。1 つは部門 ID、もう 1 つは新しい部門名です。R&D 部門 (ID は 100) の名前を Research 部門に変更する場合、以下の文を実行します。

```
READ UpdateDepartmentName.sql  [100] [Research]
```

READ 文に含まれるパラメータの数が、PARAMETERS 文で宣言されているパラメータの数より少ない場合、Interactive SQL は、それらが最初に .SQL ファイルの中で使用されたときに値の入力を要求します。たとえば、以下の文では新しい部門名が省略されています。これを実行すると、Interactive SQL は UPDATE 文を実行するときにその値の入力を要求します。

```
READ UpdateDepartmentName.sql [100];
```

関連情報

READ 文 [Interactive SQL] [1248 ページ]

1.4.4.213 PASSTHROUGH 文 [SQL Remote]

SQL Remote 管理のパススルーモードを起動または停止します。

構文

ユーザ ID へのパススルーの開始

```
PASSTHROUGH [ ONLY ] FOR userid, ...
```

パブリケーションのすべてのサブスクライバへのパススルーの開始

```
PASSTHROUGH [ ONLY ] FOR SUBSCRIPTION
TO [ owner. ]publication-name [ ( constant ) ]
```

パススルーの停止

```
PASSTHROUGH STOP
```

備考

パススルーモードでは、SQL 文はすべてデータベースサーバで実行されます。また、トランザクションログに保存され、メッセージに含めてサブスクライバに送信されます。パススルーモードの起動に ONLY キーワードを使用した場合、この文はサブスクライバに送られるだけで、サーバでは実行されません。パススルーセクションにストアドプロシージャへの呼び出しが含まれている場合、そのストアドプロシージャは、パススルーコマンドを発行するサーバ上に存在していなければなりません (これらのストアドプロシージャの中には、サーバ上では実行されないものもあります)。パススルー SQL 文の受信者は、ユーザ ID リ

スト、または指定されたパブリケーションのすべてのサブスクライバです。パススルーモードは、統合データベースからリモートデータベースへの変更を適用したり、リモートデータベースから統合データベースへ文を送信するのにも使用できます。

パブリケーションのサブスクライバへのパススルーは、サブスクリプションを起動したリモートデータベースに文を送信します。サブスクリプションを作成しても、起動していないリモートデータベースには送信しません。

PASSTHROUGH STOP は、現在の接続のパススルーモードを停止します。PASSTHROUGH STOP 文は、パススルーモードを開始した接続で実行してください。接続のパススルーモードを起動し、その接続が PASSTHROUGH STOP 文の実行前に切断されると、切断によって PASSTHROUGH STOP 文が暗黙的に実行されます。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

```
PASSTHROUGH FOR Sam_Singer ;
...
( SQL statements to be executed at the remote database )
...
PASSTHROUGH STOP ;
```

関連情報

FORWARD TO 文 [1110 ページ]
UNLOAD 文 [1376 ページ]

1.4.4.214 PIVOT 句

SELECT 文の FROM 句 (FROM pivoted-derived-table) のテーブル式を、行列変換された派生テーブルに変換します。行列変換された派生テーブルでは、テーブル式のカラムから取得したローの値を複数のカラムに回転し、必要に応じて結果セットに含まれるカラムを集計するための容易な方法を提供します。

構文

```
FROM pivoted-derived-table
```

```
pivoted-derived-table :
pivot-source-table PIVOT [ XML ] ( pivot-clause ) [ AS ] pivoted-correlation-name
```

```
pivot-source-table : table-expression
```

```
pivot-clause :
   aggregate-clause pivot-for-clause pivot-in-clause
```

```
aggregate-clause :
aggregate-function( [ aggregate-expression ] ) [ [ AS ] aggregate-alias ] [,...]
```

```
pivot-for-clause :
FOR pivot-column
| FOR ( pivot-column [,...] )
```

```
pivot-in-clause :
IN( constant-expression [[ AS ] constant-expression-alias ] [,...] )
| IN( ( constant-expression [,...] ) [[ AS ] constant-expression-alias ] [,...] )
| IN variable-name
| IN ( subquery-expression )
| IN( ALL )
| IN( ANY )
```

パラメータ

XML 句

新しい単一カラムの集計およびピボット値を XML 形式で出力するには、XML を指定します。この句は、定数ではない数値リストを使用して pivot-in-clause の値を指定するときに使用します。XML 句は、IN (ANY)、IN (ALL)、IN (subquery-expression) 句とともに使用されます。

aggregate-function([aggregate-expression]) [[AS] aggregate-alias]

一連のローについて単一の値を返す任意の集合関数を指定します。

- aggregate-function: 一連のローについて単一の値を返す任意の集合関数を指定します (スカラ関数とも呼ばれる)。
- aggregate-expression: 集合関数のパラメータを指定します。集計式では、pivot-source-table 内のカラムのみを参照する必要があります。

集合関数で許可されている場合は、ORDER BY 句および DISTINCT 句を指定できます。例:

```
LIST( DISTINCT DepartmentID ORDER BY Salary )
```

- aggregate-alias: 集合関数のエイリアスを指定します。集合関数のリストには、エイリアスのない集合関数を 1 つまで含めることができます。集計のエイリアスは、IN 句のエイリアスとともに、行列変換された派生テーブルの新しいカラム名を生成するときに使用します。ベストプラクティスとして、常に集計のエイリアスを指定してください。

FOR 句

PIVOT を実行するデータとして、複数のカラムを指定します。pivot-column には、pivot-source-table 内のカラムを指定してください。複数の pivot-column を指定する場合は、括弧で囲んで指定してください。

IN (constant-expression [[AS] constant-expression-alias] [,...])

PIVOT を実行するデータとして、一連の定数式を指定します。この構文は、FOR 句に 1 つのカラム (FOR pivot-column) のみをリストするときに使用します。

エイリアスを指定しない場合、定数式を表す文字列が暗黙的エイリアスとなります。たとえば、定数 10 の暗黙的エイリアスは '10' です。ベストプラクティスとして、常に constant-expression のエイリアスを指定してください。IN 句の定数式における暗黙的および明示的エイリアスは、集計のエイリアスとともに、行列変換された派生テーブルの新しいカラム名を生成するときに使用します。

行列変換された派生テーブルの新しいカラムは、それぞれ集合関数と IN 項目のペアに対応しており、このペアを表す名前が付きます。名前の最初の部分は IN 項目のエイリアスで、次の部分 (アンダースコアの後ろ) は集合関数のエイリアスです。生成されたカラム名が無効な識別子の場合、エラーが生成されて文が失敗します。

IN ((constant-expression [,...]) [[AS] constant-expression-alias] [,...])

PIVOT を実行するデータとして、定数式を含む一連のリストを指定します。IN 句のこの形式は、FOR 句で複数のカラムを指定するときに使用します。FOR 句で指定するカラムの数は、IN 句の定数のリストにある項目数と一致させる必要があります。

エイリアスを指定しない場合、定数式のリストを表す文字列が暗黙的エイリアスとなります。たとえば、定数リスト (10, 20) の暗黙的エイリアスは "(10, 20)" です。IN 句の定数式における暗黙的および明示的エイリアスは、集計のエイリアスとともに、行列変換された派生テーブルの新しいカラム名を生成するときに使用します。

IN variable-name

PIVOT を実行するデータとして、一連の定数式を含む変数を指定します。IN 句のこの形式は、定数を使用する IN 句と類似しています。定数を持つ IN 句に関する上記のすべての条件は、変数を使用する IN 句でも保持する必要があります。

DESCRIBE 文、OPEN 文、または RUN 文に行列変換された派生テーブルが含まれる場合は、変数を設定する必要があります。変数は、ARRAY 型を使用して宣言する必要があります。IN 句のエイリアスは、変数の現在値を使用して暗黙的に定義されます。たとえば、変数が配列 (10, 20) の場合、最初の定数のエイリアスは "10"、2 番目の定数のエイリアスは "20" です。

FOR 句で指定するピボット列が 1 つだけの場合、変数は、その要素が pivot-column のデータに準拠したドメインを持つ配列である必要があります。たとえば、FOR 句が FOR DepartmentID で、DepartmentID カラムが INT 型の場合、IN 句では INT 型の配列として定義した変数 @var を使用できます。

FOR 句に一連のピボット列を指定する場合、変数は、その要素が FOR 句のカラムに準拠したドメインを持つローの配列である必要があります。たとえば、FOR 句が FOR (DepartmentID, State) で、DepartmentID カラムが INT 型で State カラムが CHAR(16) 型の場合、IN 句では ROW(X INT, Y CHAR(16)) 型の配列として定義した変数 @var を使用できます。

IN (subquery-expression)

サブクエリで見つかったすべての値に対して PIVOT を実行する場合は、IN (subquery-expression) を指定します。

派生テーブルの XML カラムは、FOR 句のカラム名で定義された名前になります。これは、一連の項目を含む XML 要素で、1 つの項目が、ピボットカラムと集計のエイリアスの値の各ペアを示します。NULL 値は、空の文字列として XML 要素にエンコードされます。

FOR 句が FOR pivot-column (たとえば、COUNT(*) AS "COUNT" FOR DepartmentID) の場合、XML カラムの各項目のフォーマットは、以下のようになります。

```
<item>
<column name="DepartmentID">100</column>
<column name="COUNT">1</column>
</item>
```

IN 句のこの形式を使用する場合は、XML キーワードを指定します。

IN (ALL)

pivot-source-table のすべての値に対して PIVOT を実行する場合は、IN (ALL) を指定します。

IN (ALL) は、IN (subquery-expression) と同義になります。ここで、subquery-expression は pivot-source-table です。

IN 句のこの形式を使用する場合は、XML キーワードを指定します。

IN (ANY)

pivot-source-table の値に対して PIVOT を実行する場合は、IN (ANY) を指定します。ANY を指定した場合、XML カラムは IN (ALL) で生成された XML カラムと類似していますが、集計の NULL 値は XML 文字列から除外されます。IN 句のこの形式により、より簡潔な XML カラムが生成されます。

IN (ANY) 句による行列変換された派生テーブルは、IN(ALL) 句を指定した派生テーブルと類似していますが、NULL 値は追加カラムの XML 文字列に含まれません。

IN 句のこの形式を使用する場合は、XML キーワードを指定します。

備考

行列変換された派生テーブルの定義には、入力テーブル式 pivot-source-table が含まれます。PIVOT を実行するカラムおよび値は、FOR 句および IN 句で定義されます。行列変換された派生テーブルのグループ化カラムは、pivot-source-table のカラムのサブセットです。行列変換された派生テーブルは、グループ化カラムの pivot-source-table をグループ化し、集合句で指定された集合関数を計算することで求められます。行列変換された派生テーブルは、グループ化カラムの値ごとに 1 つのカラムを持ちます。行列変換された派生テーブルに追加カラムが追加されます。この追加カラムは、集合句内の項目と IN 句内の項目のペアごとに 1 つ追加されます。新しいカラムの値は、集合句で指定された集合関数です。これらの新しいカラムの名前は、集合関数の集合句で指定されたエイリアスと、IN 句で指定されたエイリアスおよび値から生成されます。集合関数が A 個指定され、IN 句が I 個の要素を持つ場合、追加カラムは合計 A x I 個です。ただし、XML 句が指定された場合は、追加カラムが 1 つだけ追加されます。このカラムには、IN 句および集合句のすべての値の組み合わせの集合が、XML 形式で 1 つの文字列として含まれます。

権限

pivot-source-table および任意のサブクエリ (IN subquery-expression など) で参照されるオブジェクトに対する SELECT 権限が必要です。DESCRIBE 文、OPEN 文、または RUN 文の IN variable-name 構文では、文の現在の範囲内に variable-name を設定する必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Employees テーブルからのデータを選択し、DepartmentID カラムの Department ID が 100、200、300、400、または 500 のデータに対して PIVOT を実行します。

```
SELECT *
FROM ( SELECT DepartmentID, State, Salary
       FROM   Employees
       WHERE State IN ( 'OR', 'CA', 'AZ', 'UT' )
     ) MyPivotSourceData
   PIVOT (
      SUM( Salary ) TotalSalary
      FOR DepartmentID IN ( 100, 200, 300, 400, 500 )
   ) MyPivotedData
ORDER BY State;
```

STATE	100_TotalSalary	200_TotalSalary	300_TotalSalary	400_TotalSalary	500_TotalSalary
AZ	(NULL)	(NULL)	93,732.000	(NULL)	85,300.800
CA	(NULL)	156,600.000	(NULL)	(NULL)	(NULL)
OR	(NULL)	47,653.000	(NULL)	80,339.000	54,790.000
UT	306,318.690	37,900.000	31,200.000	107,129.000	59,479.000

結果では、PIVOT が実行された値を明示するため、集計のエイリアスおよび DepartmentID の値が結果セットのカラム名に含まれます (100_TotalSalary など)。この例のカラム名は、"部門 X の合計給与" を意味します。各 State/DepartmentID のタプルにおける従業員の給与が集計されます (この例では、合計)。

次に示すのは、ピボットカラムを 1 つだけ持つときの IN variable-name 構文の例です。変数は、pivot-column に準拠したドメインの要素の配列である必要があります。

```
CREATE VARIABLE @var INT ARRAY;
```

```
SET @var = ( SELECT ARRAY_AGG( DISTINCT  DepartmentID ORDER BY Salary ) FROM
Employees )
SELECT *
FROM ( SELECT DepartmentID,State, Salary FROM Employees ) p
PIVOT  (
        SUM(Salary) S, COUNT(*) C
        FOR DepartmentID in @var
) PivotTable
```

次に示すのは、一連のピボットカラムを持つ場合の IN `variable-name` 構文の例です。変数は、その要素が FOR 句のカラムに準拠したドメインを持つローの配列である必要があります。

```
CREATE VARIABLE @var ROW( DeptID INT, St CHAR(16)) ARRAY;
SET @var = ( SELECT ARRAY_AGG( DISTINCT ROW( DepartmentID, State) ORDER BY
Salary ) FROM Employees )
SELECT *
FROM ( SELECT DepartmentID,State, Salary FROM Employees ) p
PIVOT (
    SUM ( Salary ) S, COUNT(*) C
    FOR ( DepartmentID, State ) IN @var
) PivotTable
```

次の文は、サポートされている異なる PIVOT 構成を使用して、同一の行列変換された派生テーブルを計算します。異なる PIVOT 構成で、必要とする行列変換された派生テーブルを生成する代替の方法を提供します。IN 句は、現在のデータに応じて動的に生成されます。

PIVOT (...IN (constant-list))

IN 句で指定する定数が既知の場合に、この形式を使用できます。

```
SELECT *
   FROM ( SELECT DepartmentID, City FROM Employees) E
   PIVOT  (
     COUNT(*) AS C
     FOR DepartmentID IN ( 100, 200, 300 ) ) AS PivotedTable
 ORDER BY City;
```

City	100_C	200_C	300_C
Charlottetown	0	2	0
Cornwall	2	1	1
Elora	0	1	0
...	...	...	...

PIVOT (...IN (generated-constant-list))

動的に生成される IN 句を使用すると、Employees テーブルの現在のデータに応じて、必要な IN 句を生成できます。

```
BEGIN
DECLARE stmt LONG VARCHAR;
 SET stmt =    ( SELECT 'SELECT * INTO dba.ConstantsTable
        FROM ( SELECT DepartmentID, City FROM Employees ) p
        PIVOT  (
            COUNT(*) AS C
            FOR DepartmentID IN ( ' + ( SELECT LIST( DepartmentID )
     FROM ( SELECT DISTINCT DepartmentID FROM Employees WHERE DepartmentID <
400 ) T )
   +  ' ) )  AS PivotedTable
 ORDER BY City' );
```

```
EXECUTE IMMEDIATE stmt;
END;
```

文が終了すると、作成した行列変換された派生テーブルである XMLTable のコンテンツを表示する、次の文が実行されます。

```
SELECT * FROM XMLTable;
```

City	100_C	200_C	300_C
Charlottetown	0	2	0
Cornwall	2	1	1
Elora	0	1	0
...	...	...	...

PIVOT (...IN variable-name)

IN 句は、ARRAY 型の変数を使用して指定できます。この型には、必要な一連の値を設定できます。

```
BEGIN
DROP TABLE IF EXISTS VarTable;
DECLARE @var INT ARRAY;
SET @var = ( SELECT  ARRAY_AGG( DepartmentID )  FROM  ( SELECT DISTINCT
DepartmentID FROM Employees WHERE DepartmentID < 400 ) T ) ;
SELECT * INTO dba.VarTable
        FROM (SELECT DepartmentID, City FROM Employees) E
        PIVOT  (
            COUNT(*) AS C
            FOR DepartmentID IN @var ) AS PivotedTable
ORDER BY City;
END
SELECT * from VarTable;
```

City	100_C	200_C	300_C
Charlottetown	0	2	0
Cornwall	2	1	1
Elora	0	1	0
...	...	...	...

PIVOT XML(...IN (subquery-expression))

次の例は、行列変換された派生テーブルを XML 形式で生成し、そのデータから行列変換された派生テーブルを抽出します。IN 句の定数が既知ではない場合に、この形式を使用できます。

```
BEGIN
DECLARE qry LONG VARCHAR;
DECLARE city_c INT;
DECLARE total_c INT;
CREATE OR REPLACE VARIABLE globalvals ARRAY OF ROW( City CHAR(26),
DepartmentID INT, C INT );
SELECT COUNT(*) total_c, COUNT( DISTINCT city ) city_c,
ARRAY_AGG( ROW ( City, DepartmentID, C ) ORDER BY city, DepartmentID )  INTO
total_c, city_c, globalvals
FROM (
SELECT PivotedTable.City, DepartmentID, C
```

```
FROM   ( SELECT *
         FROM (SELECT DepartmentID, City FROM Employees) E
         PIVOT  XML (
             COUNT(*) AS C
             FOR DepartmentID IN ( SELECT DepartmentID FROM Employees WHERE
DepartmentID < 400 ) )  AS EEE  ) AS PivotedTable,
LATERAL (SELECT * FROM  openxml( PivotedTable.[DepartmentID_xml], '/PivotSet/
item')
          WITH ( DepartmentID INT 'column[@name="DepartmentID"]',  C  INT
'column[@name="C"]' ) ) XXX
) AS  dt;
SELECT
 'SELECT ( globalvals[[ row_num ]]).city AS City,
 ' || LIST('(globalvals[[row_num+ ' || row_num || ']]).C AS ' || STRING( '[',
globalvals[[row_num+1]].DepartmentID, '_C]') ) ||  '
 INTO dba.XMLTable FROM sa_rowgenerator(1, ' || total_c || ' , ' || (total_c/
city_c) || ' )' INTO qry
FROM sa_rowgenerator( 0, (total_c/city_c) -1);
DROP TABLE IF EXISTS XMLTable;
EXECUTE IMMEDIATE qry;
DROP VARIABLE IF EXISTS globalvals;
END;
```

例が終了すると、作成した行列変換された派生テーブルである XMLTable のコンテンツを表示する、次の文が実行されます。

```
SELECT * FROM XMLTable;
```

City	100_C	200_C	300_C
Charlottetown	0	2	0
Cornwall	2	1	1
Elora	0	1	0
...	...	...	...

関連情報

FROM 句 [1112 ページ]
UNPIVOT 句 [1382 ページ]

1.4.4.215 PREPARE 文 [ESQL]

後で実行する文を準備するか、カーソルを定義します。

構文

```
PREPARE statement-name
   FROM statement [ FOR { UPDATE [ cursor-concurrency ] | READ ONLY } ]
[ DESCRIBE describe-type INTO [ [ SQL ] DESCRIPTOR ] descriptor ]
[ WITH EXECUTE ]
```

```
statement-name :  identifier | hostvar
```

```
statement :   string | hostvar
```

```
describe-type :
  [ ALL | BIND VARIABLES | INPUT | OUTPUT | SELECT LIST ]
  [ LONG NAMES [ [ [ OWNER. ]TABLE. ]COLUMN ]
    | WITH VARIABLE RESULT ]
```

```
cursor-concurrency :
BY { VALUES | TIMESTAMP | LOCK }
```

パラメータ

statement-name

文の名前は、識別子またはホスト変数とすることができます。ただし、複数の SQLCA を使用する場合には識別子を使用しないでください。そのような場合に識別子を使用すると、2 つの準備文の文番号が同じになり、誤った文が実行されたり開かれたりすることがあります。また、マルチスレッドのアプリケーションの場合、文の名前に識別子を使用することはお奨めしません。文の名前は複数のスレッドから同時に参照される場合があるためです。

DESCRIBE 句

DESCRIBE INTO DESCRIPTOR を使用すると、準備文は指定した記述子に記述されます。記述タイプとして、DESCRIBE 文で許容されるいずれかのものを使用できます。

FOR UPDATE | FOR READ ONLY

カーソルで文が使用される場合のカーソル更新可能性を定義します。FOR READ ONLY カーソルは UPDATE (位置付け) 操作や DELETE (位置付け) 操作には使用できません。FOR READ ONLY がデフォルトです。

FOR UPDATE を指定したカーソル要求への応答では、データベースサーバは value-sensitive カーソルまたは sensitive カーソルを提供します。insensitive カーソルと asensitive カーソルは更新できません。

BY VALUES | BY TIMESTAMP

データベースサーバは、キーセット駆動型カーソルを使用して、結果セットをスクロールしているときにローが変更または削除された場合にアプリケーションが通知されるようにします。

BY LOCK 句

データベースサーバは、結果セットのフェッチされたローに対する意図的ローロックを取得します。このロックは長期間のロックであり、トランザクションがコミットまたはロールバックされるまで保持されます。

WITH EXECUTE 句

WITH EXECUTE 句を使用すると、CALL または SELECT 文ではなく、ホスト変数が含まれていない場合にのみ、文が実行されます。正常に実行された後、文はその場で削除されます。文の準備と記述が正常に終了し、文が実行できない場合、警告 SQLCODE 111、SQLSTATE 01W08 が設定され、文は削除されます。

DESCRIBE INTO DESCRIPTOR 句と WITH EXECUTE 句を使用すると、必要なクライアント/サーバ通信を最小限に抑えて、パフォーマンスを改善できる場合があります。

WITH VARIABLE RESULT 句

WITH VARIABLE RESULT 句は、複数の結果セットと異なる数または種類のカラムを持つプロシージャの記述に使用します。

WITH VARIABLE RESULT を使用する場合、記述後にデータベースサーバによって SQLCOUNT 値に次のいずれかの値が設定されます。

0

結果セットは変更される場合があります。各 OPEN 文の後でプロシージャコールを記述し直してください。

1

結果セットは固定です。再度記述する必要はありません。

> **注記**
>
> 互換性を保つために、COMMIT、PREPARE TO COMMIT、ROLLBACK 文の準備は引き続きサポートされています。ただし、特定のアプリケーション環境で必要とされるため、トランザクション管理操作はすべて、静的 Embedded SQL を使用して実行する必要があります。また、他の Embedded SQL システムは、動的トランザクション管理操作をサポートしません。

備考

PREPARE 文は、`statement` から SQL 文を作成し、作成した文を `statement-name` と関連付けます。この文名を参照し、文を実行します。または、文が SELECT 文または CALL 文の場合、カーソルを開きます。`statement-name` は、`sqlca.h` ヘッダファイルの中で定義される a_sql_statement_number 型のホスト変数であり、自動的に含まれます。識別子を `statement-name` で使用する場合、この `statement-name` を使用して各モジュールに準備できるのは、それぞれ 1 文だけです。

`statement-name` にホスト変数を使用する場合は、short int 型にします。`sqlca.h` 内には、a_sql_statement_number というこの型の型定義があります。この型を SQL プリプロセッサが認識し、DECLARE セクションの中で使用できます。ホスト変数は、PREPARE 文の実行中にデータベースによって定義されるため、初期化する必要はありません。

権限

なし。

関連する動作

以前に同じ名前で準備した文が失われます。

WITH EXECUTE を指定した文は、実行が成功した場合のみ削除されます。それ以外の場合は、DROP を使って文を使用後に削除してください。DROP しないと、文が使ったメモリを再使用できません。

標準

ANSI/ISO SQL 標準

PREPARE は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部を構成します。オプションの FOR UPDATE、FOR READ ONLY、DESCRIBE、WITH EXECUTE 句は標準にありません。

例

次の文は、単純なクエリを準備します。

```
EXEC SQL PREPARE employee_statement FROM
'SELECT Surname FROM Employees';
```

関連情報

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
DESCRIBE 文 [ESQL] [1019 ページ]
OPEN 文 [ESQL] [SP] [1218 ページ]
EXECUTE 文 [ESQL] [1093 ページ]
DROP STATEMENT 文 [ESQL] [1066 ページ]

1.4.4.216 PREPARE TO COMMIT 文

COMMIT を正常に実行できるかどうかをチェックします。

構文

PREPARE TO COMMIT

備考

PREPARE TO COMMIT 文は、COMMIT がうまく実行できるかどうかをテストします。COMMIT を実行するときデータベースの整合性違反があると、文はエラーを発生します。

PREPARE TO COMMIT 文は、ストアドプロシージャ、トリガ、イベント、またはバッチに使用できません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文のシーケンスは、外部キーが Employees テーブルをチェックするため、エラーになります。

```
EXECUTE IMMEDIATE
   "SET OPTION wait_for_commit = 'On'";
EXECUTE IMMEDIATE "DELETE FROM Employees
   WHERE EmployeeID = 160";
EXECUTE IMMEDIATE "PREPARE TO COMMIT";
```

次の一連の文では、削除文の実行時に整合性違反があっても、エラーは発生しません。PREPARE TO COMMIT 文はエラーを返します。

```
SET OPTION wait_for_commit= 'On';
DELETE
FROM GROUPO.Departments
WHERE DepartmentID = 100;
PREPARE TO COMMIT;
```

関連情報

COMMIT 文 [771 ページ]
ROLLBACK 文 [1284 ページ]

1.4.4.217 PRINT 文 [T-SQL]

クライアントにメッセージを返すか、データベースサーバメッセージウィンドウにメッセージを表示します。

構文

```
PRINT format-string [, arg-list ]
```

備考

PRINT 文は、Open Client アプリケーションまたは jConnect アプリケーションから接続している場合、クライアントのウィンドウにメッセージを返します。Embedded SQL または ODBC アプリケーションから接続している場合、このメッセージはデータベースサーバメッセージウィンドウに表示されます。

フォーマット文字列は、オプション引数リスト (arg-list) にある引数のプレースホルダを含むことができます。プレースホルダのフォームは `%nn!` で、`nn` は 1 ～ 20 の間の整数です。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文はメッセージを表示します。

```
PRINT 'Display this message';
```

次の文は、PRINT 文内でのプレースホルダの使い方を示します。

```
DECLARE @var1 INT, @var2 INT
```

```
SELECT @var1 = 3, @var2 = 5
PRINT 'Variable 1 = %1!, Variable 2 = %2!', @var1, @var2
```

関連情報

MESSAGE 文 [1211 ページ]

1.4.4.218 PUT 文 [ESQL]

指定したカーソルにローを挿入します。

構文

```
PUT cursor-name
{ USING DESCRIPTOR sqlda-name | FROM hostvar-list }
[ INTO { DESCRIPTOR sqlda-name | hostvar-list } ]
[ ARRAY :row-count ]
```

```
cursor-name : identifier | hostvar
```

```
sqlda-name : identifier
```

```
hostvar-list: インジケータ変数を含めることができる
```

```
row-count : integer | hostvar
```

備考

指定したカーソルにローを挿入します。カラムの値は、最初の SQLDA から取得されます。または、INSERT 文に指定したカラム (INSERT カーソル用) または SELECT リストのカラム (SELECT カーソル用) と 1 対 1 で対応するホスト変数リストから取得されます。

PUT 文は、INSERT 文または SELECT 文のカーソルでのみ使用でき、これらの文では FROM 句に指定した 1 つのテーブルを参照するか、1 つのベーステーブルで構成される更新可能ビューを参照します。

SQLDA 内の sqldata ポインタが NULL ポインタである場合、そのカラムには値を指定しません。sqldata ポインタが DEFAULT VALUE を持つ場合、カラムはこの値を使います。そうでない場合は、NULL 値を使います。

2 番目の SQLDA またはホスト変数のリストには、PUT 文の結果が格納されています。

オプションの ARRAY 句を使って、ワイドプットを実行します。ワイドプットでは一度に複数のローが挿入され、パフォーマンスが改善されます。整数値は、挿入するローの数です。SQLDA には各エントリの変数 (ロー数 * カラム数) を入れます。最初のローは SQLDA の変数 0 から (ロー当たりのカラム数) -1 に入り、以後のローも同様です。

注記

スクロール (value-sensitive) カーソルの場合は、新しいローが WHERE 句と一致し、キーセットカーソルが移植を完了していない場合に、挿入されたローが表示されます。動的カーソルの場合は、挿入されたローが WHERE 句と一致すると、そのローが表示される場合があります。Insensitive カーソルは更新できません。

権限

カーソルで参照されるテーブルの所有者であるか、テーブルに対する INSERT 権限を持っているか、INSERT ANY TABLE システム権限を持っている必要があります。

関連する動作

ローを value-sensitive (キーセット駆動型) カーソルに挿入するとき、挿入されたローは結果セットの最後に表示されます。ローがクエリの WHERE 句に一致しない場合も、ORDER BY 句が正常にローを結果セットの別の場所に置く場合も同様です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Embedded SQL での PUT の使い方を示します。

```
EXEC SQL PUT cur_employee FROM :employeeID, :surname;
```

関連情報

UPDATE 文 [1391 ページ]
UPDATE (位置付け) 文 [ESQL] [SP] [1386 ページ]
DELETE 文 [1016 ページ]
DELETE 文 (位置付け) [ESQL] [SP] [1014 ページ]
INSERT 文 [1167 ページ]
SET 文 [1325 ページ]

1.4.4.219 RAISERROR 文

エラー信号を送り、クライアントにメッセージを送信します。

構文

```
RAISERROR error-number [ format-string ] [, arg-list ]
```

パラメータ

error-number

error-number は、17000 を超える 5 桁の整数です。このエラー番号はグローバル変数 @@error に格納されます。

format-string

format-string は、最大 255 バイト長の文字列です。オプション引数リスト (arg-list) にある引数のプレースホルダを含むことができます。プレースホルダの形式は %nn! で、nn は 1 ～ 20 の整数です。

備考

RAISERROR 文で、ユーザ定義エラーの信号を送り、クライアント上にメッセージを送信できます。

新しいエラーメッセージを作成するには、CREATE ERROR 文を使用します。ISYSUSERMESSAGE システムテーブルのメッセージを表示するには、SYSUSERMESSAGE システムビューに問い合わせます。

Adaptive Server Enterprise RAISERROR 文でサポートされる拡張値は、SQL Anywhere ではサポートされていません。

format-string を指定しないか、空にすると、エラー番号を使用してシステムテーブルのエラーメッセージが検索されます。Adaptive Server Enterprise は、17000 から 19999 までのメッセージを SYSMESSAGES テーブルから取得します。SQL Anywhere では、このテーブルは空のビューであるため、この範囲のエラーに対してはフォーマット文字列を指定してください。エラー番号が 20000 以上のメッセージは、ISYSUSERMESSAGE システムテーブルから取得します。

中間レベルの RAISERROR のステータスとコード情報は、プロシージャが終了すると失われます。結果が返されるときに RAISERROR と一緒にエラーが発生した場合は、エラー情報が返され、RAISERROR 情報は失われます。アプリケーションでは、別の実行ポイントでグローバル変数 @@error を検査して、中間の RAISERROR ステータスを問い合わせることができます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

RAISERROR 文は Transact-SQL 複合文およびプロシージャでは使用できません。

例

この例では、RAISERROR を使用して接続を禁止します。

```
CREATE PROCEDURE DBA.login_check()
BEGIN
    // Allow a maximum of 3 concurrent connections
    IF( DB_PROPERTY('ConnCount') > 3 ) THEN
         RAISERROR 28000
      'User %1! is not allowed to connect -- there are ' ||
                    'already %2! users logged on',
      Current User,
      CAST( DB_PROPERTY( 'ConnCount' ) AS INT )-1;
    ELSE
        CALL sp_login_environment;
    END IF;
END
go
GRANT EXECUTE ON DBA.login_check TO PUBLIC
go
SET OPTION PUBLIC.login_procedure='DBA.login_check'
go
```

関連情報

BEGIN 文 [745 ページ]
CREATE MESSAGE 文 [T-SQL] [856 ページ]
SYSUSERMESSAGE システムビュー [1858 ページ]
CREATE TRIGGER 文 [T-SQL] [989 ページ]
CREATE TRIGGER 文 [982 ページ]
SIGNAL 文 [SP] [1332 ページ]

1.4.4.220 READ 文 [Interactive SQL]

ファイルから Interactive SQL 文を読み込む。

構文

```
READ [ ENCODING encoding ] filename [ parameter ] ...
```

```
encoding : identifier | string
```

パラメータ

ENCODING

ENCODING 句では、ファイルの読み込みに使用されるエンコードを指定できます。READ 文は、ファイルを読み込むときにエスケープ文字を処理しません。ファイル全体が指定されたコードであると想定します。

Interactive SQL を実行するとき、データの読み込みに使用されるエンコードは次の順序で決定されます。

1. ENCODING 句で指定されたエンコード (この句が指定されている場合)。
2. ファイル内でバイトオーダーマーク (BOM) で指定されたエンコード (BOM が指定されている場合)。
3. default_isql_encoding オプションで指定されたエンコード (このオプションが設定されている場合)。
4. 実行しているプラットフォームのデフォルトエンコード。英語版 Windows コンピュータでは、デフォルトエンコードは 1252 です。

備考

READ 文は、指定したファイルから Interactive SQL 文のシーケンスを読み込みます。このファイルには、別の READ 文を含む有効な Interactive SQL 文が含まれています。READ 文はどのような深さにまでもネストできます。

`filename` にファイル拡張子がない場合、Interactive SQL は同じファイル名で拡張子 `.sql` を持つものを検索します。

`filename` に絶対パスが含まれていない場合でも、Interactive SQL はファイルを検索します。`filename` の場所は、READ 文の場所に基づいて次にように決定されます。

- READ 文が Interactive SQL で直接実行された場合、Interactive SQL は、まず、Interactive SQL の実行場所のディレクトリを基準に `filename` のパスを解決しようとします。これで解決できなかった場合、Interactive SQL は、環境変数 SQLPATH で指定されたディレクトリで `filename` を検索し、次に環境変数 PATH で指定されたディレクトリで検索を行います。
- READ 文が外部ファイル (`.sql` ファイルなど) に含まれている場合、Interactive SQL は、まず、その外部ファイルのロケーションを基準に、`filename` のパスを解決しようとします。これで解決できなかった場合、Interactive SQL は、Interactive SQL の実行場所のディレクトリを基準に、`filename` を検索します。引き続き失敗した場合、Interactive SQL は、環境変数 SQLPATH の中で指定されたディレクトリ内で検索し、さらに環境変数 PATH の中で指定されたディレクトリ内で検索します。

パラメータは、SQL スクリプトファイル名の後にリストできます。これらのパラメータは、文ファイルの先頭の PARAMETERS 文で指定したパラメータに対応します。

パラメータ名は角カッコで囲む必要があります。Interactive SQL は、ソースファイルに含まれている `{ parameter-name }` (`parameter-name` は該当するパラメータ名) を検出するたびに、対応するパラメータを代入します。

スクリプトファイルに渡すパラメータには、識別子、数値、引用符付きの識別子、または文字列を指定できます。パラメータの周囲を引用符で囲むときは、入れ替えるテキストの中に引用符を入れてください。識別子、数、または文字列 (スペースまたはタブを含む) ではないパラメータは、角カッコ ([ ]) で囲みます。こうすると、スクリプトファイルの中で任意のテキストを置き換えることができます。

十分なパラメータがスクリプトファイルに渡されない場合、Interactive SQL は足りないパラメータの値を要求するメッセージを表示します。

Interactive SQL で `reload.sql` ファイルを実行する場合は、パラメータとして暗号化キーを指定する必要があります。READ 文にキーを指定していない場合、Interactive SQL でキーの入力を要求するプロンプトが表示されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の SQL 文が含まれるスクリプトファイル `myscript.sql` があると想定します。

```
PARAMETERS par1, par2;
BEGIN
DECLARE v_par1 int;
DECLARE v_par2 varchar(200);
SET v_par1 = {par1};
SET v_par2 = {par2};
MESSAGE STRING('PAR1 Value: ', v_par1 ) TO CLIENT;
MESSAGE STRING('PAR2 Value: ', v_par2 ) TO CLIENT;
END;
```

次の READ 文を実行すると、c:¥temp¥myscript.sql でコマンドが実行され、par1 が値 123 に、par2 が値 041028 に置換されます。

```
READ 'c:¥¥temp¥¥myscript.sql' 123 '041028';
```

この READ 文は次の結果を返します。

```
PAR1 Value: 123
PAR2 Value: 041028
```

次の例では、Interactive SQL を起動して、特定の OEM コードページを使用する架空のファイルを処理するよう指示します。

```
dbisql READ ENCODING 'cp437' myfile.sql
```

関連情報

PARAMETERS 文 [Interactive SQL] [1228 ページ]

1.4.4.221 READTEXT 文 [T-SQL]

指定したオフセットから開始して、指定したバイト数のテキスト値とイメージ値をデータベースから読み込みます。この機能は Transact-SQL との互換性のためにのみ提供されているものであり、使用しないことをお奨めします。

構文

```
READTEXT table-name.column-name
text-pointer-offset text-size
[ HOLDLOCK ]
```

備考

READTEXT は、データベースから CHAR、NCHAR、BINARY カラムを読み込むのに使用されます。ビューに対して READTEXT 操作を実行することはできません。

権限

そのテーブルの所有者であるか、そのテーブルに対する SELECT 権限を持っているか、または SELECT ANY TABLE システム権限を持っていることが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の Embedded SQL の例では、TEXTPTR を使用して、MarketingInformation テーブルの ProductID 500 に関連付けられている Description カラムを見つけます。

テキストポインタは、変数 txtptr に格納され、READTEXT 文のパラメータとして指定されます。READTEXT 文は、カラムオフセット 181 から開始して 55 バイトを返します。

```
EXEC SQL BEGIN DECLARE SECTION;
char            hostvar[100];
EXEC SQL END DECLARE SECTION;
EXEC SQL create variable txtptr binary(16);
EXEC SQL set txtptr =
    ( SELECT textptr(Description)
        FROM GROUPO.MarketingInformation
        WHERE ProductID = '500' );
EXEC SQL PREPARE S1 FROM
    'READTEXT GROUPO.MarketingInformation.Description txtptr 181 55';
EXEC SQL EXECUTE S1 INTO :hostvar;
printf( "hostvar: %s¥n", hostvar );
```

READTEXT は次の文字列を返します。

```
Lightweight 100% organically grown cotton construction.
```

関連情報

WRITETEXT 文 [T-SQL] [1412 ページ]
GET DATA 文 [ESQL] [1124 ページ]
TEXTPTR 関数 [テキストとイメージ] [561 ページ]

1.4.4.222 REFRESH MATERIALIZED VIEW 文

クエリ定義を実行することで、マテリアライズドビューのデータを初期化または更新します。

構文

```
REFRESH MATERIALIZED VIEW view-list
 [ WITH {
      ISOLATION LEVEL isolation-level
      | { EXCLUSIVE | SHARE } MODE } ]
 [ FORCE BUILD ]
```

```
view-list :
[ owner.]materialized-view-name [, ... ]
```

```
isolation-level :
READ UNCOMMITTED
| READ COMMITTED
| SERIALIZABLE
| REPEATABLE READ
| SNAPSHOT
```

パラメータ

WITH 句

WITH 句は、基本となるベーステーブルでリフレッシュ中に使用するロックの種類を指定する場合に使用します。ロックの種類によって、マテリアライズドビューの移植方法とトランザクションの同時実行性への影響が決まります。WITH 句の設定は、マテリアライズドビュー自体にかけられるロックの種類には影響しません。このロックは常に排他ロックです。ロックに指定できる句は、次のいずれかです。

ISOLATION LEVEL isolation-level

WITH ISOLATION LEVEL は、リフレッシュ操作を実行する場合の独立性レベルを変更する場合に使用します。元の独立性レベルは、文の完了時に接続に対してリストアされます。

即時ビューの場合、`isolation-level` に指定できるのは SERIALIZABLE のみです。

スナップショットアイソレーションの場合、トランザクションスナップショットレベルのみが REFRESH MATERIALIZED VIEW 文でサポートされています。独立性レベルとして SNAPSHOT を指定してください。statement-snapshot と readonly-statement-snapshot レベルはサポートされていません。

EXCLUSIVE MODE

WITH EXCLUSIVE MODE は、独立性レベルを変更しないが、基本となるテーブルにコミットされたデータと矛盾しないようにデータを確実に更新する場合に使用します。WITH EXCLUSIVE MODE を使用すると、基本となるすべてのベーステーブルに排他テーブルロックがかけられます。リフレッシュ操作が完了するまで、他のトランザクションによって、基本となるテーブルに対する問い合わせ、更新、その他のアクションは実行できなくなります。排他テーブルロックを取得できない場合、再表示操作は失敗し、エラーが返されます。

SHARE MODE

WITH SHARE MODE は、リフレッシュ操作の実行中に、基本となるテーブルを他のトランザクションで読み込めるようにするために使用します。この句を指定すると、再表示操作が実行される前から、再表示操作が完了するまで、基本となるすべてのベーステーブルの共有テーブルロックが取得されます。

FORCE BUILD 句

デフォルトでは、REFRESH MATERIALIZED VIEW 文を実行すると、データベースサーバはマテリアライズドビューが古い (つまり、マテリアライズドビューが最後にリフレッシュされた後、基本となるテーブルが変更されている) かどうかをチェックします。マテリアライズドビューが古くない場合、リフレッシュは行われません。FORCE BUILD 句は、マテリアライズドビューが古いかどうかに関係なく、強制的にマテリアライズドビューをリフレッシュするときに指定します。

備考

この文は、`view-list` にリストされているマテリアライズドビューを初期化またはリフレッシュする場合に使用します。

古くなっていないマテリアライズドビューに対して REFRESH MATERIALIZED VIEW 文を実行した場合、FORCE BUILD 句が指定されていないかぎり、リフレッシュは実行されません。

ロックとデータ同時実行性の、デフォルトのリフレッシュ動作は次のとおりです。

- ビューが即時ビューの場合、スナップショットアイソレーションが有効かどうかに関係なく、デフォルトのリフレッシュ動作は WITH SHARE MODE です。
- ビューが手動ビューであり、スナップショットアイソレーションを使用している場合、デフォルトは WITH ISOLATION LEVEL SNAPSHOT です。
- ビューが手動ビューであり、スナップショットアイソレーションを使用していない場合、デフォルトは WITH SHARE MODE です。

REFRESH MATERIALIZED VIEW が正常に動作する場合、および、最適化にビューを使用する場合は、いくつかのオプションに特定の値が設定されている必要があります。また、マテリアライズドビューを作成するときにマテリアライズドビューごとに格納されるオプションの設定があります。ビューをリフレッシュしたり、ビューを最適化に使用したりするには、これらのオプション設定が現在のオプションと一致している必要があります。

部分的に処理を行った後にリフレッシュが失敗した場合、ビューは初期化されていない状態のままとなり、リフレッシュ開始前の状態にデータをリストアできません。リフレッシュが失敗したときに発生したエラーを調査して、この失敗の原因となった問題を解決し、REFRESH MATERIALIZED VIEW 文を再実行します。

また、ALTER MATERIALIZED VIEW 文の IMMEDIATE REFRESH 句を使用すると、基本となるデータが変更されたときすぐにビューを変更してリフレッシュできます。

接続に WITH HOLD 句を使用して開かれたカーソルがあり、このカーソルで文またはトランザクションのスナップショットを使用している場合、この文は実行できません。

権限

マテリアライズドビューを持っているか、それに対する INSERT 権限を持っている必要があります。さらに、基本となるテーブルの所有者であるか、それらのテーブルに対する SELECT 権限を持っているか、または SELECT ANY TABLE システム権限を持っていることが必要です。

関連する動作

マテリアライズドビューを参照するすべてのオープンカーソルが閉じられます。

チェックポイントは実行の開始時に実行されます。

実行の開始時と終了時に自動コミットが実行されます。

実行中は、接続の blocking オプションを使用してリフレッシュされるマテリアライズドビューに排他スキーマロックがかけれます。また、マテリアライズドビューから参照されるすべてのテーブルにブロックなしで共有スキーマロックがかけられます。WITH 句を指定した場合は、基本となるテーブルに追加のロックを取得することがあります。さらに、リフレッシュが完了するまでマテリアライズドビューは初期化されていない状態になるため、データベースサーバでクエリ最適化またはクエリ実行に使用できなくなります。

スナップショットアイソレーションを使用して REFRESH MATERIALIZED VIEW 文を実行した場合、データベースのトランザクションログには REFRESH 文のテキストと、マテリアライズドビューに挿入される個々のローのすべてのコピーが含まれます。データベースにリカバリが必要な場合、リカバリ後のビューの内容が REFRESH MATERIALIZED VIEW 文が最初に完了したときのビューの内容と正確に一致するようにするには、個々のローが必要です。さらに、データベースがミラーリングされている場合、トランザクションログ内の個々のローは個別に適用されます。そのため、スナップショットアイソレーションを使用した REFRESH MATERIALIZED VIEW 文の頻度を制限したり、BACKUP DATABASE 文を使用してトランザクションログを定期的にトランケートしたりして、トランザクションログに必要なディスク領域の量を削減することが必要になる場合があります。

標準

ANSI/ISO SQL 標準

標準になし。

例

EmployeeConfid99 というマテリアライズドビューを作成し、データを移植するとします。この処理は、次の文で実行できます。

```
CREATE MATERIALIZED VIEW EmployeeConfid99 AS
   SELECT EmployeeID, Employees.DepartmentID, SocialSecurityNumber, Salary,
ManagerID,
      Departments.DepartmentName, Departments.DepartmentHeadID
   FROM GROUPO.Employees, GROUPO.Departments
   WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid99;
```

後に、ビューが使用されるようになってから、READ COMMITTED 独立性レベル (独立性レベル 1) を使用してビューをリフレッシュし、ビューを再構築するとします。この処理は、次の文で実行できます。

```
REFRESH MATERIALIZED VIEW EmployeeConfid99
   WITH ISOLATION LEVEL READ COMMITTED
   FORCE BUILD;
```

警告

この例を実行し終わったら、作成したマテリアライズドビューを削除してください (`DROP MATERIALIZED VIEW EmployeeConfid99`)。このようにしないと、他の例を試すとき、基本となるテーブル Employees および

Departments に対するスキーマ変更を実行できなくなります。有効化されている従属マテリアライズドビューを持つテーブルのスキーマは変更できません。

関連情報

ALTER MATERIALIZED VIEW 文 [659 ページ]
BACKUP DATABASE 文 [739 ページ]
CREATE MATERIALIZED VIEW 文 [854 ページ]
sa_refresh_materialized_views システムプロシージャ [1591 ページ]
sa_materialized_view_info システムプロシージャ [1554 ページ]
sa_materialized_view_can_be_immediate システムプロシージャ [1552 ページ]

1.4.4.223 REFRESH TEXT INDEX 文

テキストインデックスをリフレッシュします。

構文

```
REFRESH TEXT INDEX text-index-name ON [ owner.]table-name
[ WITH {
      ISOLATION LEVEL isolation-level
      | EXCLUSIVE MODE
      | SHARE MODE } ]
[ FORCE { BUILD | INCREMENTAL } ]
```

パラメータ

WITH 句

WITH 句は、基本となるベーステーブルでリフレッシュ中に取得するロックの種類を指定する場合に使用します。取得したロックの種類によって、テキストインデックスの移植方法とトランザクションの同時実行性への影響が決まります。WITH 句を指定しなかった場合、デフォルトは、接続に設定されている独立性レベルに関係なく、WITH ISOLATION LEVEL READ UNCOMMITTED になります。

次の WITH 句オプションを指定できます。

ISOLATION LEVEL isolation-level

WITH ISOLATION LEVEL は、リフレッシュ操作を実行する場合の独立性レベルを変更する場合に使用します。

接続の元の独立性レベルは、文の実行終了時にリストアされます。

EXCLUSIVE MODE

WITH EXCLUSIVE MODE は、独立性レベルを変更しないが、基本となるテーブルにコミットされたデータと矛盾しないようにデータを確実に更新する場合に使用します。WITH EXCLUSIVE MODE を使用すると、基本となるベーステーブルに排他テーブルロックがかけられます。リフレッシュ操作が完了するまで、他のトランザクションによって、基本となるテーブルに対する問い合わせ、更新、その他のアクションは実行できなくなります。テーブルのロックを取得できない場合、更新操作は失敗し、エラーが返されます。

SHARE MODE

WITH SHARE MODE は、リフレッシュ操作の実行中に、基本となるテーブルを他のトランザクションで読み込めるようにするために使用します。この句を指定すると、リフレッシュ操作が実行される前に基本となるベーステーブルの共有テーブルロックが取得され、リフレッシュ操作が完了するまで保持されます。

FORCE 句

この句は、リフレッシュ方法を指定する場合に使用します。この句を指定しなかった場合、データベースサーバは、変更されたテーブルの割合に基づいて、増分更新するか、または完全再構築するかどうかを決定します。

FORCE BUILD 句

テキストインデックスを再作成することでリフレッシュします。この句は、テキストインデックスを強制的に完全再構築する場合に使用します。

FORCE INCREMENTAL 句

基本となるテーブルでの変更内容にのみ基づいてテキストインデックスをリフレッシュします。基本となるテーブルに加えられた更新の量が多くない場合、増分リフレッシュは短期間で完了します。この句は、テキストインデックスを強制的に増分更新する場合に使用します。

増分リフレッシュでは、削除済みのエントリはテキストインデックスから削除されません。このため、テキストインデックスのサイズは、現在のデータと履歴データを含めた予想サイズよりも大きくなる場合があります。通常、この問題は、常に FORCE INCREMENTAL 句を使用して手動でリフレッシュしているテキストインデックスで発生します。自動的にリフレッシュされるテキストインデックスでは、履歴データは、テキストインデックスの合計サイズの 50 % に達すると、自動的に削除されます。

備考

この文は、MANUAL REFRESH または AUTO REFRESH と定義されているテキストインデックスでのみ使用できます。

FORCE 句を使用した場合は、sa_text_index_stats システムプロシージャの結果を確認して、完全再構築 (FORCE BUILD) または増分更新 (FORCE INCREMENTAL) のどちらが最適かを決定できます。

IMMEDIATE REFRESH と定義されているテキストインデックスに対して REFRESH TEXT INDEX 文を実行することはできません。

MANUAL REFRESH テキストインデックスの場合は、テキストインデックスを再表示するかどうかを、sa_text_index_stats システムプロシージャを使用して判断します。pending_length を doc_length で除算して求められるパーセンテージを、再表示が必要かどうかを判断するうえでの目安にします。必要な再構築の種類を判断する場合も、deleted_length と doc_count を使って同じ処理を行います。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

権限

そのテーブルの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- そのテーブルに対する REFERENCES 権限
- CREATE ANY INDEX システム権限
- ALTER ANY INDEX システム権限
- CREATE ANY OBJECT システム権限
- ALTER ANY OBJECT システム権限

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、架空のテキストインデックス MarketingTextIndex をリフレッシュし、強制的に再構築します。

```
REFRESH TEXT INDEX MarketingTextIndex ON GROUPO.MarketingInformation
   FORCE BUILD;
```

関連情報

CREATE TEXT INDEX 文 [976 ページ]
ALTER TEXT INDEX 文 [724 ページ]
DROP TEXT INDEX 文 [1077 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]
sa_text_index_stats システムプロシージャ [1641 ページ]

1.4.4.224 REFRESH TRACING LEVEL 文 (廃止予定)

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。REFRESH TRACING LEVEL 文は、トレーシングセッションが進行中に、sa_diagnostic_tracing_level テーブルからトレーシングレベルをリロードします。

構文

REFRESH TRACING LEVEL

備考

この文は、sa_diagnostic_tracing_level テーブルからトレーシングレベル情報をリロードするときに使用します。この文は、プロファイル対象のデータベースから呼び出す必要があります。

トレースセッションを最初に開始すると、sa_diagnostic_tracing_level テーブルのローはサーバメモリにロードされ、トレースする情報の種類を制御します。トレースセッションを停止したり再開したりすることなくトレースされるデータのタイプを変更するには、sa_diagnostic_tracing_level テーブルの適切なローを手動で削除するかローを挿入し、REFRESH TRACING LEVEL 文を実行して設定をリロードします。

現在のトレーシングレベルを確認するには、次のように sa_diagnostic_tracing_level テーブルに問い合わせます。

```
SELECT * FROM sa_diagnostic_tracing_level WHERE enabled = 1;
```

パフォーマンス問題の解決に取り組んでいるとします。データベース全体のトレーシングレベルを高く設定し、問題の原因となるクエリを取得します。トレースセッションの開始後、システムの全ユーザについて全クエリを取得するとデータベースの処理速度が大幅に遅くなることがわかりました。そのため、トレーシングを 1 ユーザに制限し、そのユーザからレポートされる問題を待機することにしました。ただし、設定を変更するときにトレースセッションを停止するつもりはありません。

これは、コマンドラインから実行することもできます。この場合、scope=DATABASE で enabled=1 の sa_diagnostic_tracing_level テーブルのローを、scope=USER、identifier=`userid`、enabled=1 などという同等のローで置換します。次に REFRESH TRACING LEVEL 文を実行し、新規の設定を使用してトレーシングを継続します。

権限

MANAGE PROFILING システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、トレーシングレベルを再表示します。

```
REFRESH TRACING LEVEL;
```

関連情報

ATTACH TRACING 文 (廃止) [737 ページ]
DETACH TRACING 文 (廃止) [1027 ページ]
sa_diagnostic_tracing_level テーブル (廃止予定) [1426 ページ]

1.4.4.225 RELEASE MUTEX 文

指定した接続スコープミューテックスが現在の接続によってロックされている場合に、その接続スコープミューテックスを解除します。

構文

```
RELEASE MUTEX [ owner.]mutex-name
```

パラメータ

owner

ミューテックスの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

mutex-name

ミューテックスの名前。mutex-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

備考

RELEASE MUTEX 文は、ミューテックスに対するロックの 1 つのインスタンスを解放します。そのため、接続によってミューテックスが複数回ロックされている場合は、RELEASE MUTEX 文ごとにそのミューテックスの 1 つのロックのみが解放されます。

現在の接続によってミューテックスがロックされていない場合や、トランザクションスコープミューテックスに対して解放が要求されている場合は、エラーが返されます。

現在の接続によってロックされている削除されたミューテックスに対する RELEASE MUTEX 文は、成功します。

権限

UPDATE ANY MUTEX SEMAPHORE システム権限を持っているか、ミューテックスの所有者である必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、protect_my_cr_section ミューテックスを解放します。

```
RELEASE MUTEX protect_my_cr_section;
```

関連情報

DROP MUTEX 文 [1048 ページ]
CREATE MUTEX 文 [863 ページ]
LOCK MUTEX 文 [1199 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.226 RELEASE SAVEPOINT 文

現在のトランザクション内のセーブポイントを解放します。

構文

```
RELEASE SAVEPOINT [ savepoint-name ]
```

備考

セーブポイントを解放します。`savepoint-name` は、現在のトランザクションの SAVEPOINT 文で指定された識別子です。`savepoint-name` を省略すると、最新のセーブポイントが解放されます。

セーブポイントを解放しても、いずれの種類の COMMIT も実行されません。現在アクティブなセーブポイントのリストから、セーブポイントを削除するだけです。

現在のトランザクションに、対応する SAVEPOINT を入れておいてください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

RELEASE SAVEPOINT は、オプションの ANSI/ISO SQL 言語機能 T271、"Savepoints" の一部です。

関連情報

BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
COMMIT 文 [771 ページ]
ROLLBACK 文 [1284 ページ]

ROLLBACK TO SAVEPOINT 文 [1285 ページ]
SAVEPOINT 文 [1290 ページ]

1.4.4.227 REMOTE RESET 文 [SQL Remote]

カスタムデータベース抽出プロシージャで、単一トランザクションでの 1 リモートユーザのすべてのサブスクリプションを起動します。

構文

```
REMOTE RESET userid
```

備考

この文は、1 人のリモートユーザのすべてのサブスクリプションを 1 つのトランザクションで起動します。ISYSREMOTEUSER テーブルの log_sent 値と confirm_sent 値をトランザクションログの現在の位置に設定します。また、ISYSSUBSCRIPTION で作成し、起動した値を、このリモートユーザのすべてのサブスクリプションに使用するトランザクションログの現在の位置に設定します。この文はコミットを実行しません。この呼び出しの後、明示的にコミットを実行してください。

ライブデータベース上で安全な抽出処理を書き込むには、サブスクリプションが開始されたのと同じトランザクション内で、データを独立性レベル 3 で抽出します。

この文を START SUBSCRIPTION の代わりに使用できます。START SUBSCRIPTION は、関連動作として暗黙的にコミットを実行するため、リモートユーザが複数のサブスクリプションを持つ場合に START SUBSCRIPTION を使用して 1 つのトランザクションですべてのサブスクリプションを起動することはできません。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

この文のオートコミットはありません。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、リモートユーザ Sam_Singer のサブスクリプションをリセットします。

```
REMOTE RESET Sam_Singer;
COMMIT;
```

関連情報

START SUBSCRIPTION 文 [SQL Remote] [1341 ページ]
SYSREMOTEUSER システムビュー [1830 ページ]

1.4.4.228 REMOVE EXTERNAL OBJECT 文

データベースから外部オブジェクトを削除します。

構文

```
REMOVE EXTERNAL OBJECT object-name
```

パラメータ

object-name

外部オブジェクトの名前。

備考

なし。

権限

外部オブジェクトの所有者であるか、または MANAGE ANY EXTERNAL OBJECT システム権限を持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

ALTER EXTERNAL ENVIRONMENT 文 [647 ページ]
INSTALL EXTERNAL OBJECT 文 [1173 ページ]
START EXTERNAL ENVIRONMENT 文 [1338 ページ]
STOP EXTERNAL ENVIRONMENT 文 [1348 ページ]
SYSEXTERNENV システムビュー [1802 ページ]
SYSEXTERNENVOBJECT システムビュー [1804 ページ]

1.4.4.229 REMOVE JAVA 文

データベースからクラスまたは JAR ファイルを削除します。

構文

```
REMOVE JAVA
    { CLASS java-class-name [ , java-class-name ... ]
    | JAR jar-name [ , jar-name ... ] }
```

パラメータ

CLASS 句

`java-class-name` パラメータは、削除される 1 つ以上の Java クラスの名前です。現在のデータベースにインストールされているクラスを指定します。

JAR 句

`jar-name` は、一重引用符で囲まれた最大の長さが 255 の文字列値です。各 `jar-name` は、現在のデータベース内に保持されている `jar-name` と同じでなければなりません。`jar-name` と同じかどうかは、SQL システムの文字列比較ルールによって決定されます。

備考

データベースからクラスまたは JAR ファイルを削除します。クラスまたは JAR は事前にインストールしておきます。

権限

クラスまたは JAR ファイルの所有者であるか、または MANAGE ANY EXTERNAL OBJECT システム権限を持っている必要があります。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、現在のデータベースから Demo という名前の架空の Java クラスを削除します。

```
REMOVE JAVA CLASS Demo;
```

この例では、現在のデータベースから myJar という名前の架空の Java JAR を削除します。

```
REMOVE JAVA JAR 'myJar';
```

関連情報

INSTALL JAVA 文 [1175 ページ]
SYSJAR システムビュー [1812 ページ]
SYSJARCOMPONENT システムビュー [1813 ページ]
SYSJAVACLASS システムビュー [1814 ページ]

1.4.4.230 REORGANIZE TABLE 文

データベースへの連続アクセスという要件があるために、データベース全体の再構築ができない場合に、テーブルの断片化を解除します。

構文

```
REORGANIZE TABLE [ owner.]table-name
[ { PRIMARY KEY
| FOREIGN KEY foreign-key-name
| INDEX index-name } ]
```

パラメータ

次のいずれかの値に従って、テーブルを再編成します。

PRIMARY KEY 句

テーブルのプライマリキーのインデックスを再編成します。

FOREIGN KEY 句

指定の外部キーを再編成します。

INDEX 句

指定のインデックスを再編成します。

備考

テーブルが断片化されると、パフォーマンスが妨害されることがあります。この文を使って、テーブル内のローの断片化を解除したり、DELETE によって散在したインデックスを圧縮したりします。また、テーブルとそのインデックスを記録するために使われる総ページ数と、インデックスツリーに含まれるレベル数も減らします。ただし、データベースファイル全体のサイズは小さくなりません。sa_table_fragmentation と sa_index_density の各システムプロシージャを使用して、テーブル対象の処理を選択してください。

インデックスまたはキーを指定しない場合は、再編成処理によってローグループが削除されてから再挿入され、テーブル内のローの断片化が解除されます。グループごとに、テーブルの排他ロックが取得されます。グループの処理が完了すると、他の接続がテーブルにアクセスできるように、ロックが解除され、再取得されます (必要な場合は待機します)。グループの処理中はチェックポイントが中断されます。グループが終了すると、チェックポイントが発生することがあります。クラスタドインデックスが存在する場合、ローはクラスタドインデックスの順序で処理されます。存在しない場合は、プライマリキーの順序で処理されます。テーブルにクラスタドインデックスもプライマリキーもない場合、エラーが返されます。処理済みのローはテーブルの最後に再挿入され、処理の最後にローがプライマリキーによってクラスタされます。必要な作業量は、最初にローが断片化されていた程度に関係なく同じです。

インデックスまたはキーを指定すると、そのインデックスが処理されます。オペレーション中は、テーブルの排他ロックが保持され、チェックポイントは中断されます。他の接続がテーブルにアクセスしようとすると、blocking オプションの設定に応じてブロックされるか失敗します。ロック期間は、排他ロックを取得する前にインデックスページをあらかじめ読み込んでおくことで最短に抑えられます。

再編成で多数のページが修正される場合があるため、チェックポイントログが大きくなる可能性があります。これにより、データベースファイルのサイズが大きくなる可能性がありますが、チェックポイントログがシャットダウン時に削除され、ファイルはその時点でトランケートされるため、サイズは一時的に大きくなるだけです。

この文は、トランザクションログには記録されません。

文またはトランザクションのスナップショットを使用する、WITH HOLD 句を使用して開かれたカーソルがある場合、この文は実行できません。

この文の実行中に、進行状況メッセージの表示を要求できます。

また、Progress 接続プロパティを使用して、文がどの程度実行されたかを確認することもできます。

権限

テーブルの所有者であるか、REORGANIZE ANY OBJECT システム権限を持っている必要があります。

関連する動作

再編成を開始する前に、チェックポイントが実行され、空きページ数を最大限に増やそうとします。また、REORGANIZE TABLE 文の実行時は約 100 ローごとに暗黙的なコミットがあるため、大規模なテーブルを認識すると複数のコミットが実行されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、Employees テーブルのプライマリキーのインデックスを再編成します。

```
REORGANIZE TABLE GROUPO.Employees
PRIMARY KEY;
```

次の文は、Employees テーブルのテーブルページを再編成します。

```
REORGANIZE TABLE GROUPO.Employees;
```

次の文は、Products テーブルのインデックス IX_prod_name を再編成します。

```
REORGANIZE TABLE GROUPO.Products
   INDEX IX_product_name;
```

次の文は、Employees テーブルの外部キー FK_DepartmentID_DepartmentID を再編成します。

```
REORGANIZE TABLE GROUPO.Employees
   FOREIGN KEY FK_DepartmentID_DepartmentID;
```

1.4.4.231 RESIGNAL 文 [SP]

例外条件を送り返します。

構文

```
RESIGNAL [ exception-name ]
```

備考

例外ハンドラの中で RESIGNAL を使って、まだアクティブな例外を持つ複合文を終了するか、別の指定された例外のレポートを終了できます。例外は、別の例外ハンドラによって処理されるか、またはアプリケーションに返されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

RESIGNAL 文は、オプションの ANSI/ISO SQL 言語機能 P002、"Computational completeness" の一部です。

Transact-SQL

RESIGNAL 文は、Transact-SQL 複合文およびプロシージャでは使用できません。

例

例のフラグメントは SQLSTATE 52003 を除くすべての例外をアプリケーションに返します。

```
...
DECLARE COLUMN_NOT_FOUND EXCEPTION
   FOR SQLSTATE '52003';
...
EXCEPTION
WHEN COLUMN_NOT_FOUND THEN
SET message='Column not found';
WHEN OTHERS THEN
RESIGNAL;
```

関連情報

SIGNAL 文 [SP] [1332 ページ]
BEGIN 文 [745 ページ]
RAISERROR 文 [1246 ページ]

1.4.4.232 RESTORE DATABASE 文

バックアップされたデータベースをアーカイブからリストアします。

構文

```
RESTORE DATABASE filename
FROM archive-root
[ CATALOG ONLY
  | [ RENAME dbspace-name TO new-dbspace-name ] ... ]
[ HISTORY { ON | OFF } ]
[ KEY encryption-key ]
```

```
filename : string | variable
archive-root : string | variable
new-dbspace-name : string | variable
```

パラメータ

FROM clause

この句は、バックアップのロケーションを指定するために使用します。

CATALOG ONLY clause

指定されたアーカイブに関する情報を取り出し、それをバックアップ履歴ファイル (backup.syb) に保存します。アーカイブからデータをリストアするわけではありません。

RENAME clause

DB 領域ごとに新しいロケーションを指定できます。RENAME 句を DB 領域名の変更に使用することはできません。ただし、ファイル名の変更には RENAME 句を使用できます。

HISTORY clause

RESTORE DATABASE 操作を履歴ファイル backup.syb に記録するかどうかを制御できます。

KEY clause

空きページの削除をオンにした状態でバックアップされ、強力に暗号化されているアーカイブ済みデータベースを、リストアするための暗号化キーを指定できます。空きページの削除をオフにした状態でバックアップされている場合、データベースをリストアするための暗号化キーを指定する必要はありません。このキーには文字列または変数名を使用できます。

バージョン 12 では、バージョン 11 以前のデータベースサーバで作成したアーカイブバックアップをリストアできません。

備考

HISTORY OFF を指定しないかぎり、RESTORE DATABASE 操作を実行するたびにバックアップ履歴ファイル backup.syb が更新されます。backup.syb ファイルには、特定のデータベースサーバで実行された BACKUP 操作と RESTORE 操作が記録されます。次の条件にあてはまる場合は、RESTORE DATABASE 操作を backup.syb に記録しないようにすることを検討してください。

- RESTORE DATABASE 操作が頻繁に発生する
- backup.syb ファイルを定期的にアーカイブまたは削除するプロシージャがない
- ディスク領域が非常に限られている

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

RESTORE DATABASE はリストアするデータベースを置き換えます。インクリメンタルバックアップが必要な場合は、BACKUP コマンドのイメージフォーマットを使用し、トランザクションログだけを保存してください。

この文の実行中に、進行状況メッセージの表示を要求できます。

また、Progress 接続プロパティを使用して、文がどの程度実行されたかを確認することもできます。

リストア中のデータベースには接続できません。別のデータベースに接続されている必要があります。たとえば、ユーティリティデータベースに接続します。実行中のデータベースは暗号化できません。

権限

この文の実行能力は、-gu データベースオプションの設定と SERVER OPERATOR システム権限の有無によって異なります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

BACKUP DATABASE 文 [739 ページ]

1.4.4.233 RESUME 文

結果セットを返すカーソルの実行を再開します。

構文

```
RESUME cursor-name
```

```
cursor-name :  identifier | hostvar
```

備考

この文は、結果セットを返すプロシージャの実行を再開します。プロシージャは、次の結果セット (INTO 句のない SELECT 文) が見つかるまで実行されます。プロシージャが完了しても結果セットが見つからない場合は、SQLSTATE_PROCEDURE_COMPLETE 警告が設定されます。この警告は、SELECT 文に対してカーソルを RESUME するときにも設定されます。

RESUME 文は、Interactive SQL でサポートされていません。

事前にカーソルを開いておきます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次に Embedded SQL の例を示します。

```
EXEC SQL RESUME cur_employee;
EXEC SQL RESUME :cursor_var;
```

関連情報

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
FETCH 文 [ESQL] [SP] [1101 ページ]

1.4.4.234 RETURN 文

関数、プロシージャ、またはバッチを無条件で終了し、オプションで値を返します。

構文

```
RETURN [ expression ]
```

備考

RETURN 文を使うと、SQL のブロックをすぐに終了できます。`expression` を指定する場合、`expression` の値を関数またはプロシージャの値として返します。サブクエリは、`expression` では使用できません。

RETURN が内側の BEGIN ブロックにある場合、外側の BEGIN ブロックが終了します。

RETURN 文の後の文は実行されません。

関数の中では、式を関数の RETURNS データ型と同じデータ型にしてください。

プロシージャ内では、RETURN は Transact-SQL との互換性を確保するためのもので、整数のエラーコードを返すために使用されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。

例

次の関数は、3 つの数値の積を返します。

```
CREATE FUNCTION product (
   a NUMERIC,
   b NUMERIC,
   c NUMERIC )
RETURNS NUMERIC
BEGIN
   RETURN ( a * b * c );
END;
```

3 つの数値の積を計算します。

```
SELECT product(2, 3, 4);
```

product(2, 3, 4)
24.000000

次のプロシージャは、RETURN 文を使って、意味のない複雑なクエリを実行するのを避けます。

```
CREATE PROCEDURE customer_products
( in customer_ID integer DEFAULT NULL)
RESULT ( ID integer, quantity_ordered integer )
BEGIN
   IF customer_ID NOT IN (SELECT ID FROM Customers)
   OR customer_ID IS NULL THEN
      RETURN
   ELSE
      SELECT Products.ID,sum(
         SalesOrderItems.Quantity )
      FROM  GROUPO.Products,
            SalesOrderItems,
            SalesOrders
      WHERE SalesOrders.CustomerID=customer_ID
      AND SalesOrders.ID=SalesOrderItems.ID
      AND SalesOrderItems.ProductID=Products.ID
      GROUP BY Products.ID
   END IF
END;
```

関連情報

CREATE FUNCTION 文 [837 ページ]
CREATE FUNCTION 文 [外部呼び出し] [818 ページ]
CREATE FUNCTION 文 [Web サービス] [826 ページ]
CREATE PROCEDURE 文 [891 ページ]
CREATE PROCEDURE 文 [外部呼び出し] [868 ページ]
CREATE PROCEDURE 文 [Web サービス] [878 ページ]
BEGIN 文 [745 ページ]

1.4.4.235 REVOKE CONSOLIDATE 文 [SQL Remote]

このデータベースからの SQL Remote メッセージを統合データベースが受信するのを中止させます。

構文

```
REVOKE CONSOLIDATE FROM userid
```

備考

統合データベースを表すユーザ ID に対して CONSOLIDATE 権限をリモートデータベースで付与します。REVOKE CONSOLIDATE 文は、現在のデータベースからメッセージを受信するユーザリストから統合データベースのユーザ ID を削除します。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。指定したユーザのすべてのサブスクリプションを削除します。

標準

ANSI/ISO SQL 標準

標準になし。

例

- 次の文は、ユーザ ID Sam_Singer の統合ステータスを取り消します。

```
REVOKE CONSOLIDATE FROM Sam_Singer;
```

関連情報

GRAPH CONSOLIDATE 文 [SQL Remote] [1141 ページ]
GRANT ROLE 文 [1136 ページ]
REVOKE PUBLISH 文 [SQL Remote] [1276 ページ]
REVOKE REMOTE 文 [SQL Remote] [1277 ページ]

1.4.4.236 REVOKE PUBLISH 文 [SQL Remote]

名前を指定したユーザ ID を現在のパブリッシャとして識別するのを中止します。パブリッシャ権限を取り消すには、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

構文

```
REVOKE PUBLISH FROM userid
```

備考

SQL Remote インストール環境内の各データベースは、出力メッセージ内でパブリッシャユーザ ID により識別されます。現在のパブリッシャユーザ ID は、次のように CURRENT PUBLISHER 特別値のクエリによって決定できます。

```
SELECT CURRENT PUBLISHER;
```

REVOKE PUBLISH 文により、名前を指定したユーザ ID がパブリッシャとして識別されなくなります。パブリッシャを変更するには、現在のパブリッシャからパブリッシャ権限を取り消して、それらを新しいパブリッシャに付与します。

SQL Remote インストール環境の統合データベースまたはリモートデータベースでパブリッシャユーザ ID を変更する場合はそのデータベースからメッセージを受信するすべてのデータベースで、新しいパブリッシャユーザ ID に REMOTE 権限が付与されていることを確認します。この変更を行うには、すべてのサブスクリプションを一度削除し、再作成する必要があります。

データベースにアクティブな SQL Remote パブリケーションまたはサブスクリプションがあるときは、そのデータベースからパブリッシャ権限を取り消さないでください。

パブリッシャ権限の取り消しとそれを新しいユーザに付与しないことで、SQL Remote インストール環境に次のような影響を及ぼします。

- CURRENT PUBLISHER カラムがプライマリキーの一部になっているテーブルにデータを挿入できなくなります。また、出力メッセージがパブリッシャユーザ ID で識別できなくなり、受信側データベースによって受け取られなくなります。

この文を実行すると、PUBLIC.db_publisher データベース オプションの値が変更されます。

権限

SET ANY SYSTEM OPTION システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

現在のパブリッシャとしての publisher_ID の識別を終了します。

```
REVOKE PUBLISH FROM publisher_ID;
```

関連情報

GRANT ROLE 文 [1136 ページ]
GRANT PUBLISH 文 [SQL Remote] [1147 ページ]
REVOKE REMOTE 文 [SQL Remote] [1277 ページ]
REVOKE CONSOLIDATE 文 [SQL Remote] [1274 ページ]
CURRENT PUBLISHER 特別値 [87 ページ]

1.4.4.237 REVOKE REMOTE 文 [SQL Remote]

このデータベースからの SQL Remote メッセージをユーザが受信できないようにします。

構文

```
REVOKE REMOTE FROM userid, ...
```

備考

ユーザ ID が SQL Remote レプリケーションインストール環境でメッセージを受信するには、REMOTE 権限が必須です。
REVOKE REMOTE 文は、現在のデータベースからメッセージを受信するユーザリストからユーザ ID を削除します。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。指定したユーザのすべてのサブスクリプションを削除します。

標準

ANSI/ISO SQL 標準

標準になし。

例

```
REVOKE REMOTE FROM Sam_Singer;
```

関連情報

REVOKE ROLE 文 [1281 ページ]
REVOKE PUBLISH 文 [SQL Remote] [1276 ページ]
GRANT REMOTE 文 [SQL Remote] [1149 ページ]
REVOKE CONSOLIDATE 文 [SQL Remote] [1274 ページ]

1.4.4.238 REVOKE 文

ユーザとロールのシステム権限とオブジェクトレベル権限を取り消します。

構文

システム権限の取り消し

```
REVOKE [ { EXERCISE | ADMIN } OPTION FOR ] privilege
FROM grantee, ...
```

```
grantee :
{ system-role | userid }
```

オブジェクトレベル権限の取り消し

```
REVOKE object-level-privilege[,...]
ON [ owner.]table-or-view
FROM userid[,...]
```

```
object-level-privilege :
ALL [ PRIVILEGES ]
| ALTER
```

```
| DELETE
| INSERT
| LOAD
| REFERENCES [ ( column-name[,...] ) ]
| SELECT [ ( column-name[,...] ) ]
| TRUNCATE
| UPDATE [ ( column-name[,...] ) ]
```

CONNECT、INTEGRATED LOGIN、KERBEROS LOGIN の取り消し

```
REVOKE capability FROM userid[,...]
```

```
capability :
CONNECT
| INTEGRATED LOGIN
| KERBEROS LOGIN
```

プロシージャの EXECUTE の取り消し

```
REVOKE EXECUTE
ON [ owner.]procedure-name[,...]
FROM userid[,...]
```

シーケンスの USAGE の取り消し

```
REVOKE USAGE ON SEQUENCE sequence-name[,...]
FROM userid[,...]
```

パラメータ

{ EXERCISE | ADMIN } OPTION FOR clause

権限の管理者権限を取り消し、行使権限は残す場合は、ADMIN OPTION FOR 句を指定します。権限の行使権限を取り消し、管理者権限は残す場合は、EXERCISE OPTION FOR 句を指定します。この句を指定しない場合は、両方の権利が取り消されます。

REVOKE CONNECT

REVOKE CONNECT はデータベースからユーザ ID を削除するとともに、そのユーザが所有しているオブジェクト (テーブル、ビュー、プロシージャなど) を破棄します。ただし、ユーザを削除する場合は、REVOKE CONNECT 文ではなく DROP USER 文を使用することをおすすめします。ユーザによって付与されたシステム権限は有効のままですが、ユーザによって付与されたオブジェクトレベルの権限は取り消されます。

削除されているユーザが、別のユーザが所有するビューによって参照されるテーブルを所有する場合、そのユーザに対して REVOKE CONNECT 文を実行することはできません。

ユーティリティデータベースに接続するときに REVOKE CONNECT FROM DBA を実行すると、以降のユーティリティデータベースへの接続が無効になります。以降、REVOKE CONNECT の実行前に存在していた接続を使用するか、データベースサーバを再起動しないかぎり、ユーティリティデータベースへの接続はできなくなります。

REVOKE USAGE ON SEQUENCE

権限を削除してシーケンスの現在または次の値を評価する場合は、この構文を指定します。

備考

取り消された権限が grantee に付与されていない場合、文は何も実行せず、またエラーも返しません。

文を実行した結果 REVOKE がエラーで失敗した場合、システム権限を取り消された管理者の数は、min_role_admins データベースオプションで設定されている必要最小人数を下回ります。

権限の管理権限を持っているユーザのオブジェクトレベル権限を取り消すと、そのユーザが権限を付与したすべてのユーザの権限も取り消されます。そして、その権限を付与されている grantee の権限も同様に取り消されます。

ユーザの接続関連権限を取り消すと、そのユーザはデータベースに接続できなくなります。

UPGRADE ROLE システム権限のシステム権限を取り消すとき、SYS_UPGRADE_ROLE_ROLE という特別な内部表現を使用する必要があります。たとえば `REVOKE privilege-name FROM SYS_UPGRADE_ROLE_ROLE;` のように指定します。

バージョン 16.0 以前のソフトウェアで使用された権限、パーミッション、グループに関連する REVOKE 構文は、引き続きサポートされますが推奨されません。

権限

取り消すシステム権限の管理権限を持っていることが必要です。

オブジェクトレベル権限を取り消すときは、次の 1 つを持っていることが必要です。

- オブジェクトの所有権
- そのオブジェクトのオブジェクトレベル権限の管理権限
- MANAGE ANY OBJECT PRIVILEGE システム権限

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

REVOKE capability は標準の一部ではありません。REVOKE object-level-privilege および REVOKE EXECUTE は、ANSI/ISO SQL 標準のコア機能です。PRIVILEGES キーワードは REVOKE ALL (すべてのオブジェクトレベル権限を取り消す) ではオプションですが、標準では必須です。

REVOKE USAGE ON SEQUENCE は、オプションの ANSI/ISO SQL 言語機能 T176、"Sequence generator support" の一部です。

例

この例では、ユーザ Dave が Employees テーブルを更新できないようにします。

```
REVOKE UPDATE ON GROUPO.Employees FROM Dave;
```

この例では、架空のユーザ拡張ロール Finance の ShowCustomers プロシージャの実行を禁止します。

```
REVOKE EXECUTE ON ShowCustomers FROM Finance;
```

この例では、データベースからユーザ FranW を削除します。この構文は廃止されています。代わりに DROP USER 文の使用してください。

```
REVOKE CONNECT FROM FranW;
```

この例では、架空の Kerberos ユーザ pchin のデータベースログイン権限を取り消します。

```
REVOKE KERBEROS LOGIN
FROM "pchin@MYREALM.COM";
```

関連情報

REVOKE ROLE 文 [1281 ページ]

1.4.4.239 REVOKE ROLE 文

ユーザやロールからロールや権限を取り消します。

構文

システムロールの取り消し

```
REVOKE ROLE system-role
FROM grantee, ...
```

```
grantee :
{ system-role | userid }
```

```
system-role :
dbo
| DIAGNOSTICS
| PUBLIC
| rs_systabgroup
| SA_DEBUG
| SYS
| SYS_REPLICATION_ADMIN_ROLE
| SYS_RUN_REPLICATION_ROLE
| SYS_SAMONITOR_ADMIN_ROLE
| SYS_SPATIAL_ADMIN_ROLE
```

ユーザ定義ロールの取り消し

```
REVOKE [ { EXERCISE | ADMIN } OPTION FOR ] ROLE user-defined-role
FROM grantee, ...
```

```
grantee :
{ system-role | userid }
```

互換ロールの取り消し

```
REVOKE [ { EXERCISE | ADMIN } OPTION FOR ] ROLE compatibility-role-name
FROM grantee, ...
```

```
compatibility-role-name :
SYS_AUTH_BACKUP_ROLE
| SYS_AUTH_DBA_ROLE
| SYS_AUTH_PROFILE_ROLE
| SYS_AUTH_READCLIENTFILE_ROLE
| SYS_AUTH_READFILE_ROLE
| SYS_AUTH_RESOURCE_ROLE
| SYS_AUTH_SA_ROLE
| SYS_AUTH_SSO_ROLE
| SYS_AUTH_VALIDATE_ROLE
| SYS_AUTH_WRITECLIENTFILE_ROLE
| SYS_AUTH_WRITEFILE_ROLE
```

```
grantee :
{ system-role | userid }
```

パラメータ

{ EXERCISE | ADMIN } OPTION FOR clause

ロールの管理者権限を取り消し、行使権限は残す場合は、ADMIN OPTION FOR 句を指定します。ロールの行使権限を取り消し、管理者権限は残す場合は、EXERCISE OPTION FOR 句を指定します。この句を指定しない場合は、両方の権利が取り消されます。

備考

取り消されたロールが `grantee` に付与されていない場合、文は何も実行せず、またエラーも返しません。

文を実行した結果 REVOKE ROLE がエラーで失敗した場合、ロールを取り消された管理者の数は、min_role_admins データベースオプションで設定されている必要最小人数を下回ります。

MANAGE ROLES システム権限のロールを取り消すとき、SYS_MANAGE_ROLES_ROLE という特別な内部表現を使用する必要があります。例: `REVOKE ROLE role-name FROM SYS_MANAGE_ROLES_ROLE;`。

バージョン 16.0 以前のソフトウェアで使用された権限、パーミッション、グループに関連する REVOKE 構文は、引き続きサポートされますが推奨されません。

権限

取り消すロールの管理権限を持っていることが必要です。

SYS_REPLICATION_RUN_ROLE システムロールを付与する場合は、SYS_REPLICATION_ADMIN_ROLE システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、架空のユーザ Jim の SYS_AUTH_VALIDATE_ROLE 互換ロールを取り消します。

```
REVOKE ROLE SYS_AUTH_VALIDATE_ROLE FROM Jim;
```

この例では、架空のユーザ Administrator の DIAGNOSTICS システムロールを取り消します。

```
REVOKE ROLE DIAGNOSTICS FROM Administrator;
```

次の文は、ユーザ Sam_Singer から SYS_REPLICATION_ADMIN_ROLE を取り消します。

```
REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE FROM Sam_Singer;
```

次の文は、ユーザ Sam_Singer の SYS_RUN_REPLICATION_ROLE を取り消します。

```
REVOKE ROLE SYS_RUN_REPLICATION_ROLE FROM Sam_Singer;
```

関連情報

REVOKE 文 [1278 ページ]
REVOKE PUBLISH 文 [SQL Remote] [1276 ページ]
REVOKE REMOTE 文 [SQL Remote] [1277 ページ]
REVOKE CONSOLIDATE 文 [SQL Remote] [1274 ページ]

1.4.4.240 ROLLBACK 文

トランザクションを終了し、最後に行った COMMIT または ROLLBACK 以降の変更を元に戻します。

構文

```
ROLLBACK [ WORK ]
```

備考

トランザクションとは、データベース接続においてデータベースに対して実行する作業の論理単位であり、COMMIT 文または ROLLBACK 文で囲まれた部分のことです。ROLLBACK 文は、現在のトランザクションを終了し、前回の COMMIT または ROLLBACK 以降にデータベースに対して行われたすべての変更を元に戻します。

権限

なし。

関連する動作

WITH HOLD 句で開かれた以外のすべてのカーソルを閉じます。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

COMMIT 文 [771 ページ]
ROLLBACK TO SAVEPOINT 文 [1285 ページ]

1.4.4.241 ROLLBACK TO SAVEPOINT 文

SAVEPOINT の後に加えられた変更を取り消します。

構文

```
ROLLBACK TO SAVEPOINT [ savepoint-name ]
```

備考

ROLLBACK TO SAVEPOINT 文は、SAVEPOINT が作成されてから加えられた変更を取り消します。SAVEPOINT の前に加えられた変更は取り消されず、そのまま残ります。

`savepoint-name` は、現在のトランザクションの SAVEPOINT 文で指定された識別子です。`savepoint-name` を省略すると、最新のセーブポイントが使用されます。指定したセーブポイントの後にあるセーブポイントは自動的に解放されます。

現在のトランザクションに、対応する SAVEPOINT を入れておいてください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

ROLLBACK TO SAVEPOINT は、オプションの ANSI/ISO SQL 言語機能 T271 の一部です。

関連情報

BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
COMMIT 文 [771 ページ]
RELEASE SAVEPOINT 文 [1261 ページ]

ROLLBACK 文 [1284 ページ]
SAVEPOINT 文 [1290 ページ]

1.4.4.242 ROLLBACK TRANSACTION 文 [T-SQL]

SAVE TRANSACTION の後に加えられた変更を取り消します。

構文

```
ROLLBACK TRANSACTION [ savepoint-name ]
```

備考

ROLLBACK TRANSACTION 文は、SAVE TRANSACTION を使用してセーブポイントが作成されてから加えられた変更を取り消します。SAVE TRANSACTION の前に加えられた変更は取り消されず、そのまま残ります。

`savepoint-name` は、現在のトランザクションの SAVE TRANSACTION 文で指定された識別子です。`savepoint-name` を省略した場合、未確定の変更はすべてロールバックされます。指定したセーブポイントの後にあるセーブポイントは自動的に解放されます。

現在のトランザクションに、対応する SAVE TRANSACTION を入れておいてください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、値 10、20 などの 5 つのローを表示します。DELETE の効果は ROLLBACK TRANSACTION 文によって取り消されますが、前の INSERT または UPDATE の効果は取り消されません。

```
BEGIN
    SELECT row_num INTO #tmp
    FROM sa_rowgenerator( 1, 5 )
    UPDATE #tmp SET row_num=row_num*10
    SAVE TRANSACTION before_delete
    DELETE FROM #tmp WHERE row_num >= 3
    ROLLBACK TRANSACTION before_delete
    SELECT * FROM #tmp
END
```

関連情報

ROLLBACK TO SAVEPOINT 文 [1285 ページ]
BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
COMMIT 文 [771 ページ]
SAVE TRANSACTION 文 [T-SQL] [1288 ページ]

1.4.4.243 ROLLBACK TRIGGER 文

トリガによって加えられた変更を取り消します。

構文

```
ROLLBACK TRIGGER [ WITH raiserror-statement ]
```

備考

ROLLBACK TRIGGER 文は、トリガを起動するデータの操作など、トリガの中で実行された処理を元に戻します。

必要に応じて、RAISERROR 文を実行できます。RAISERROR 文を実行すると、エラーがアプリケーションに返されます。RAISERROR 文を実行しないと、エラーは返されません。

ネストしたトリガの中で ROLLBACK TRIGGER 文を使用し、RAISERROR 文を発行しないと、最も内側のトリガとそのトリガを起動した文だけが元に戻されます。

権限

なし。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

Transact-SQL

ROLLBACK TRIGGER は、Watcom SQL と Transact-SQL の両方のストアドプロシージャでサポートされています。
ROLLBACK TRIGGER は Adaptive Server Enterprise でサポートされています。

関連情報

CREATE TRIGGER 文 [982 ページ]
ROLLBACK 文 [1284 ページ]
ROLLBACK TO SAVEPOINT 文 [1285 ページ]
RAISERROR 文 [1246 ページ]

1.4.4.244 SAVE TRANSACTION 文 [T-SQL]

現在のトランザクション内でセーブポイントを確立します。

構文

```
SAVE TRANSACTION savepoint-name
```

備考

現在のデータベース内でセーブポイントを確立します。`savepoint-name` は ROLLBACK TRANSACTION 文の中で使用できる識別子です。トランザクションが終了すると、すべてのセーブポイントは自動的に解放されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、値 10、20 などの 5 つのローを表示します。DELETE の効果は ROLLBACK TRANSACTION 文によって取り消されますが、前の INSERT または UPDATE の効果は取り消されません。

```
BEGIN
    SELECT row_num INTO #tmp
    FROM sa_rowgenerator( 1, 5 )
    UPDATE #tmp SET row_num=row_num*10
    SAVE TRANSACTION before_delete
    DELETE FROM #tmp WHERE row_num >= 3
    ROLLBACK TRANSACTION before_delete
    SELECT * FROM #tmp
END
```

関連情報

SAVEPOINT 文 [1290 ページ]
BEGIN TRANSACTION 文 [T-SQL] [752 ページ]
COMMIT 文 [771 ページ]
ROLLBACK TRANSACTION 文 [T-SQL] [1286 ページ]

1.4.4.245 SAVEPOINT 文

現在のトランザクション内でセーブポイントを確立します。

構文

```
SAVEPOINT [ savepoint-name ]
```

備考

現在のデータベース内でセーブポイントを確立します。savepoint-name は RELEASE SAVEPOINT または ROLLBACK TO SAVEPOINT 文の中で使用できる識別子です。トランザクションが終了すると、すべてのセーブポイントは自動的に解放されます。

トリガまたはアトミックな複合文の実行中に確立されたセーブポイントは、アトミックオペレーションが終了すると自動的に解放されます。

セーブポイント内からプロキシテーブルのデータは修正できません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

SAVEPOINT 文は、オプションの ANSI/ISO SQL 言語機能 T271、"Savepoints" の一部です。

Transact-SQL

Transact-SQL では、SAVE TRANSACTION 文を使用してセーブポイントを作成します。

関連情報

RELEASE SAVEPOINT 文 [1261 ページ]
ROLLBACK TO SAVEPOINT 文 [1285 ページ]
SAVE TRANSACTION 文 [T-SQL] [1288 ページ]

1.4.4.246 SELECT 文

データベースから情報を取得します。

構文

```
[ WITH temporary-views ]
  SELECT [ ALL | DISTINCT ] [ row-limitation-option-1 ] select-list
[ INTO { hostvar-list | variable-list | table-name } ]
[ INTO LOCAL TEMPORARY TABLE { table-name } ]
[ INTO TABLE table-name ]
[ INTO VARIABLE variable-list ]
[ FROM from-expression ]
[ WHERE search-condition ]
[ GROUP BY group-by-expression ]
[ HAVING search-condition ]
[ WINDOW window-expression ]
[ ORDER BY { expression | integer } [ ASC | DESC ], ... ]
[ FOR READ ONLY | for-update-clause ]
[ FOR XML xml-mode ]
[ FOR JSON json-mode ]
[ row-limitation-option-2 ]
[ OPTION( query-hint, ... ) ]
```

```
temporary-views :
  regular-view, ...
| RECURSIVE { regular-view | recursive-view }, ...
```

```
regular-view :
  view-name [ ( column-name, ... ) ]
  AS ( query-block )
```

```
recursive-view :
  view-name ( column-name, ... )
  AS ( initial-query-block UNION ALL recursive-query-block )
```

query-block: SQL 構文の共通要素のマニュアルを参照

```
row-limitation-option-1 :
FIRST
| TOP { ALL | limit-expression } [ START AT startat-expression ]
```

```
row-limitation-option-2 :
LIMIT { [ offset-expression, ] limit-expression | limit-expression OFFSET offset-
expression }
```

```
limit-expression : simple-expression
```

```
startat-expression : simple-expression
```

```
offset-expression : simple-expression
```

```
simple-expression :
integer
| variable
| ( simple-expression )
| ( simple-expression { + | - | * } simple-expression )
```

```
select-list :
expression [ [ AS ] alias-name ], ...
| *
| window-function OVER { window-name | window-spec }
   [ [ AS ] alias-name ]
|  sequence-expression
```

```
sequence-expression
sequence-name [ CURRVAL | NEXTVAL ]
FROM table-name
```

```
window-function :
RANK( )
| DENSE_RANK( )
| PERCENT_RANK( )
| CUME_DIST( )
| ROW_NUMBER( )
| aggregate-function
```

```
for-update-clause
FOR UPDATE
| FOR UPDATE cursor-concurrency
| FOR UPDATE OF [ ( column-name, ... ) ]
```

```
 cursor-concurrency :
BY { VALUES | TIMESTAMP | LOCK }
```

```
xml-mode :
RAW [ , ELEMENTS ]
| AUTO [ , ELEMENTS ]
| EXPLICIT
```

```
json-mode :
AUTO | EXPLICIT | RAW
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| FORCE NO OPTIMIZATION
| option-name=option-value
```

```
option-name : identifier
```

```
option-value :
```

```
hostvar （許可されたインジケータ）
| string
| identifier
| number
```

```
from-expression : FROM 句の説明を参照
group-by-expression : GROUP BY 句の説明を参照
window-expression : WINDOW 句の説明を参照
```

パラメータ

WITH or WITH RECURSIVE clause

1 つ以上の共通テーブル式を定義します。共通テーブル式はテンポラリビューでもあり、文の残りの部分に使用されます。これらの式は、非再帰的か自己再帰的のいずれかに指定できます。再帰的な共通テーブル式は、RECURSIVE キーワードが指定されている場合にのみ、単独で表示されるか、非再帰的なテーブル式と混合で表示されます。相互再帰的な共通テーブル式は、サポートされていません。

SELECT クエリブロックが次のいずれかの場所に表示されている場合のみ、この句を使用できます。

- ビュー定義の最上位レベルの SELECT クエリブロックを含む最上位レベルの SELECT クエリブロック内
- INSERT クエリブロック内の最上位レベルの SELECT 文内
- 任意のタイプの SQL 文で派生テーブルを定義しているネストされた SELECT クエリブロック内

再帰的な式は、初期のサブクエリと再帰的なサブクエリから構成されます。初期クエリは、ビューのスキーマを暗黙的に定義します。再帰的なサブクエリには、FROM 句内のビューへの参照を入れてください。それぞれの反復中に、この参照によって前の反復でビューに追加されたローだけが参照されます。参照は、外部ジョインの NULL 入力側では表示できません。再帰的な共通テーブル式は、集合関数を使用できません。また、GROUP BY、ORDER BY、DISTINCT の各句を含むことはできません。

リモートテーブルでは WITH 句はサポートされません。WITH 句は、INTERSECT、UNION、EXCEPT クエリブロック内でも使用されることがあります。

この機能は Watcom SQL ダイアレクトでのみ使用可能です。

```
WITH RECURSIVE
  manager ( EmployeeID, ManagerID,
            GivenName, Surname, mgmt_level ) AS
( ( SELECT EmployeeID, ManagerID,       -- initial subquery
           GivenName, Surname, 0
    FROM Employees AS e
    WHERE ManagerID = EmployeeID )
  UNION ALL
  ( SELECT e.EmployeeID, e.ManagerID,   -- recursive subquery
           e.GivenName, e.Surname, m.mgmt_level + 1
    FROM Employees AS e JOIN manager AS m
     ON   e.ManagerID =  m.EmployeeID
      AND e.ManagerID <> e.EmployeeID
      AND m.mgmt_level < 20 ) )
SELECT 'Manager', * FROM manager
WHERE mgmt_level > 0
UNION ALL
SELECT 'Employee', * FROM manager
WHERE mgmt_level = 0
ORDER BY mgmt_level, Surname, GivenName;
```

ALL or DISTINCT clause

ALL (デフォルト) は、SELECT 文の句を満たすすべてのローを返します。DISTINCT を指定すると、重複した出力ローが削除されます。多くの文では、DISTINCT を指定すると、実行時間が非常に長くなります。そのため、DISTINCT を使用するのは必要な場合だけにしてください。

row-limitation clauses

row-limitation 句を使用することによって、WHERE 句を満たすローのサブセットのみを返すことができます。`row-limitation` 句は、一度に 1 つのみ指定できます。これらの句を指定する場合は、ローの順序を意味のあるものにするために ORDER BY 句も指定する必要があります。

row-limitation-option-1

TOP 引数と START AT 引数には、ホスト変数、整数定数、または整数変数を使用した簡単な算術演算を指定できます。TOP 引数は 0 以上にします。START AT 引数は 0 より大きい値にします。

`startat-expression` を指定しない場合、デフォルトは 1 です。TOP の引数が ALL の場合、`startat-expression` で始まるすべてのローが返されます。`TOP limit-expression START AT startat-expression` 句は、`LIMIT ( startat-expression -1 ), limit-expression`、または `LIMIT limit-expression OFFSET ( startat-expression -1 )` と同等です。

row-limitation-option-2

LIMIT 引数と OFFSET 引数には、ホスト変数、整数定数、または整数変数を使用した簡単な算術演算を指定できます。LIMIT 引数は 0 以上にします。OFFSET 引数は 0 以上にします。`offset-expression` を指定しない場合、デフォルトは 0 です。

ロー制限句 `LIMIT offset-expression, limit-expression` は `LIMIT limit-expression OFFSET offset-expression` と同等です。どちらの構成も `TOP limit-expression START AT ( offset-expression + 1 )` と同等です。

LIMIT キーワードはデフォルトでは無効になっています。LIMIT キーワードを有効にするには、reserved_keywords オプションを使用します。

select-list clause

`select-list` は、カンマで区切った式のリストであり、データベースから何を取り出すかを指定します。アスタリスク (*) は、FROM 句に記述された全テーブルのすべてのカラムを選択することを意味します。

集合関数は、`select-list` で許可されています。サブクエリも、`select-list` で許可されます。それぞれのサブクエリは、括弧で囲みます。

エイリアス名はクエリを通じて使用でき、エイリアスの式を表します。

エイリアス名も、SELECT 文から出力された各カラムの最上部に、Interactive SQL で表示されます。オプションのエイリアス名を式の後で指定しないと、Interactive SQL は式を表示します。

> **注記**
>
> 次の文字は、エイリアスでは使用できません。
>
> - " (二重引用符)
> - 制御文字 (0X20 未満の文字)
> - 円記号
> - 角括弧

- 逆引用符

INTO clause

次の 3 つの INTO 句を使用します。

INTO hostvar-list clause

この句は Embedded SQL 内でのみ使用します。これは SELECT 文の結果が移動する場所を指定します。`select-list` 内のそれぞれの項目に対して、1 つのホスト変数項目を指定してください。`select-list` 項目は、順番にホスト変数の中に置かれます。インジケータホスト変数も各ホスト変数と一緒に使用でき、プログラムは `select-list` 項目が NULL であったかどうかを通知できます。

クエリの結果、選択されたローが存在しない場合、変数は更新されず、ローが見つからないことを示す警告が表示されます。

INTO variable-list clause

この句はプロシージャとトリガの中でだけ使用されます。これは SELECT 文の結果が移動する場所を指定します。`select-list` 内の項目ごとに変数が 1 つずつ必要です。`select-list` 項目は、順番に変数の中に置かれます。

INTO table-name clause

この句は、テーブルの作成とデータの挿入に使用します。

テーブルを作成するには、クエリが次のいずれかの条件を満たしている必要があります。

- `table-name` がテンポラリテーブル名であるか、**`owner`**`.table-name` として指定されている。
- `select-list` に 1 つの * または複数の項目が含まれている。

カラムが 1 つの永久テーブルを作成するには、テーブル名を `owner.table` として指定する必要があります。

この文を使用すると、実行の前に、テーブル作成に関連する動作として COMMIT が行われます。新規テーブルに権限は付与されません。この文は、CREATE TABLE とそれに続く INSERT...SELECT の省略形です。

この句を使用して作成されるテーブルには、プライマリキーは定義されていません。ALTER TABLE を使用してプライマリキーを追加できます。プライマリキーは、テーブルに更新または削除を適用する前に追加してください。そうしないと、これらのオペレーションによって、影響を受けるローのトランザクションログにすべてのカラム値が記録されます。

INTO LOCAL TEMPORARY TABLE clause

この句は、ローカルテンポラリテーブルを作成し、クエリの結果を移植するために使用します。この句を使用する場合、テンポラリテーブル名の先頭に # を付ける必要はありません。

INTO TABLE table-name clause

この句は、常にテーブルを作成し、そのテーブルにクエリの結果を格納します。INTO TABLE 句は INTO `table-name` 句と同じように動作します。ただし、クエリが INTO `table-name` 句の `select-list` 条件を満たす必要がない場合は例外です。

INTO VARIABLE variable-list clause

この句は、SELECT 文の結果が移動する場所を指定します。クエリによって返されるカラム値が順番にロー変数のフィールドに割り当てられます。

INTO 句に単一の項目があり、`select-list` に複数の項目がある場合は、単一の項目がロー変数と見なされ、返されるカラム値が順番にロー変数のフィールドに割り当てられます。INTO 句に複数の項目が含まれる場合は、INTO `variable-list` セマンティックが適用されます。

> **注記**
>
> ロー変数にはローや配列などの別の複合データ型を含めることはできず、単一変数のコレクションとして構成する必要があります。それ以外の場合は、エラーが返されます。

FROM clause

`table-expression` の中で指定されるテーブルとビューからローを取り出します。FROM 句を指定しない SELECT 文を使用して、テーブルから抽出されなかった式の値を表示できます。たとえば、この 2 つの文は同じです。また、グローバル変数 @@version の値を表示します。

```
SELECT @@version;
SELECT @@version FROM SYS.DUMMY;
```

WHERE clause

この句は、FROM 句の中で指定したテーブルから選択するローを指定します。この句を FROM 句の一部である ON フレーズの代わりに使用すると、複数のテーブルをジョインできます。

GROUP BY clause

カラム、エイリアス名、または関数によってグループ分けできます。クエリの結果には、指定したカラム、エイリアス、または関数の中の個別の値の各セットに対し 1 つのローが入ります。DISTINCT や、UNION、INTERSECT、EXCEPT などの集合操作と同じように、GROUP BY 句は NULL 値を各ドメインの他の値と同様に扱います。つまり、グループ化属性に複数の NULL 値が存在すると 1 つのグループが形成されます。集合関数をこれらのグループに適用して、意味のある結果を取得することができます。

GROUP BY を使う場合、`select-list`、HAVING 句、ORDER BY 句が参照できるのは、GROUP BY 句の中で指定した識別子だけです。ただし、`select-list` と HAVING 句は集合関数を持つことができます。

HAVING clause

この句は、個々のロー値ではなくグループ値に基づいてローを選択します。HAVING 句を使用できるのは、文に GROUP BY 句があるか、`select-list` が集合関数のみから成る場合だけです。HAVING 句の中で参照されるカラム名は、GROUP BY 句の中に入れるか、または HAVING 句の中の集合関数に対するパラメータとして使用する必要があります。

WINDOW clause

この句は、AVG や RANK などの Window 関数を使用するウィンドウのすべてまたは一部を定義します。

ORDER BY clause

この句はクエリの結果をソートします。ORDER BY リストの各項目には、昇順の場合 (デフォルト) は ASC、降順の場合は DESC のラベルを付けることができます。式が整数 `n` である場合、クエリ結果は `select-list` の `n` 番目の項目でソートされます

特定の順序でローが返されるようにする唯一の方法は ORDER BY を使用することです。ORDER BY 句がない場合は、データベースサーバが最も効率のよい順序でローを返します。結果セットの内容は、最後にアクセスしたローやその他の要因によって変わることがあります。

ORDER BY 句の項目は、テーブル参照変数 (TABLE REF 型として定義される変数) にすることはできません。

Embedded SQL の場合は、データベースから結果を取得し、その値を INTO 句でホスト変数に格納するために、SELECT 文を使用します。SELECT 文が返すことができるローは、1 つだけです。複数のローを対象にクエリを実行する場合は、カーソルを使います。

FOR UPDATE or FOR READ ONLY clause

これらの句は、クエリに対して開かれているカーソル経由で更新を許可するかどうかを指定し、許可する場合、使用する同時実行性のセマンティックを指定します。この句は、FOR XML 句と一緒に使用できません。

SELECT 文の FOR 句を使用しない場合、カーソルの更新可能性は、カーソルの宣言とカーソルの同時実行性を API が指定する方法によって変わります。ODBC、JDBC、OLE DB、ADO.NET、Embedded SQL では、文の更新可能性は明示的になり、アプリケーションが上書きしなければ、読み込み専用カーソルが使用されます。Open Client とストアドプロシージャでは、カーソルの更新可能性を指定する必要はありません。また、デフォルトは FOR UPDATE です。

Open Client とストアドプロシージャでは、カーソルの更新可能性と文の更新可能性は ansi_update_constraints データベースオプションの設定、文の特徴に応じて変わります。たとえば、ORDER BY、DISTINCT、GROUP BY、HAVING、UNION、集合関数、ジョイン、非更新可能なビューが文に含まれるかどうかなどです。ストアドプロシージャでは、ORDER BY 句なしの単一テーブルのクエリの場合、または ansi_update_constraints オプションが Off に設定されている場合、カーソルのデフォルトは FOR UPDATE です。ansi_update_constraints オプションが Cursors または Strict に設定されている場合、ORDER BY 句を含むクエリ上のカーソルのデフォルトは READ ONLY です。ただし、FOR UPDATE 句を使用して、カーソルを更新可能と明示的に示すこともできます。ORDER BY 句またはジョインを使用してカーソルに対する更新を可能にするにはコストがかかるため、2 つ以上のテーブルのジョインを含むクエリに対するカーソルは READ ONLY であり、ansi_update_constraints データベースオプションが Off でないかぎり、更新可能にすることはできません。

FOR READ ONLY として宣言されたカーソルは、UPDATE (位置付け) 文、DELETE (位置付け) 文、または PUT 文には使用できません。FOR READ ONLY は Embedded SQL のデフォルトです。

FOR UPDATE 句は、明示的にカーソルを更新可能にします。FOR UPDATE を単独で使用しても、そのこと自体では、文の結果セットのローに対する同時実行性の制御には影響を及ぼしません。このことを行うには、FOR UPDATE BY LOCK または FOR UPDATE BY [VALUES | TIMESTAMP] のいずれかを指定します。

FOR UPDATE BY LOCK clause

データベースサーバは、結果セットのフェッチされたローに対する意図的ローロックを取得します。このロックは長期間のロックであり、トランザクションがコミットまたはロールバックされるまで保持されます。SELECT 文が INTO 句を使用する場合、実行可能な位置付け UPDATE がないため、意図したローロックが取得されません。

FOR UPDATE BY { VALUES | TIMESTAMP } clause

FOR UPDATE BY TIMESTAMP または FOR UPDATE BY VALUES を指定すると、データベースサーバはキーセット駆動型の (value-sensitive) カーソルを使用して最適な同時実行性を使用します。このような状況では、アプリケーションが (別の文を使用して) カーソルの範囲外のローを変更した場合、または別の接続でローが変更されたことを示す警告やエラーがサーバによって生成されてもアプリケーションがそれを無視する場合、更新内容が消失する可能性があります。

文で意図的なロックをかけるには、次のいずれかを実行します。

- クエリで FOR UPDATE BY LOCK を指定
- クエリの FROM 句で HOLDLOCK、WITH (HOLDLOCK)、WITH (UPDLOCK)、または WITH (XLOCK) を指定
- CONCUR_LOCK を指定する API の読み出しでカーソルを開く
- 更新のフェッチを示す属性でローをフェッチ

FOR UPDATE OF 句は、UPDATE (位置付け) 文、DELETE (位置付け) 文、または PUT 文で変更可能なカラムに明示的に名前を付けます。この句は、他の FOR UPDATE 句、FOR READ ONLY 句、または FOR XML 句と一緒には使用できません。

FOR UPDATE OF column-list clause

FOR UPDATE OF 句を指定すると、データベースサーバは、位置付け UPDATE 文または位置付け DELETE 文で変更可能なカラムを、この句で明示的に指定されたカラムに制限します。別のカラムを変更しようとすると、カラムが見つからないというエラーになります。リスト内で参照されているカラムが実際に存在するかどうか、および、そのカラムのテーブルが更新可能なテーブルであるかどうかを調べるチェックは行われません。

FOR XML clause

この句は、結果セットが XML ドキュメントとして返されるように指定します。XML のフォーマットは、指定するモードによって異なります。この句は、FOR UPDATE 句または FOR READ ONLY 句と一緒には使用できません。FOR XML で宣言したカーソルは暗黙的に READ ONLY になります。

RAW モードを指定すると、結果セットの各ローは XML <row> 要素として表され、各カラムは <row> 要素の属性として表されます。

AUTO モードを指定すると、クエリの結果は、ネストされた XML 要素として返されます。`select-list` 内で参照される各テーブルは、XML 内で要素として表されます。要素のネスト順は、テーブルが `select-list` 内で参照される順序に基づいています。

EXPLICIT モードを指定すると、生成された XML ドキュメントの形式を制御できます。EXPLICIT モードにすると、RAW モードや AUTO モードに比べて、要素の名前付けとネスト構造の指定がより柔軟にできます。

FOR JSON clause

この句は、結果セットが JSON フォーマットで返されるように指定します。JSON フォーマットは、指定したモードによって異なります。この句は、FOR UPDATE 句または FOR READ ONLY 句と一緒には使用できません。FOR JSON で宣言したカーソルは暗黙的に READ ONLY になります。

RAW モードを指定した場合、結果セットの各ローは、フラットにされた JSON 表現として返されます。

AUTO モードは、クエリのジョインに基づいて、クエリの結果をネストされた JSON オブジェクトとして返します。

EXPLICIT モードを指定すると、生成された JSON ドキュメントの形式を制御できます。EXPLICIT モードを使用すると、カラムの指定と階層オブジェクトのネストをより柔軟に行い、同一または異なる配列を作成することができます。

OPTION clause

この句は、クエリの処理方法についてヒントを示します。次のクエリヒントがサポートされます。

MATERIALIZED VIEW OPTIMIZATION clause

MATERIALIZED VIEW OPTIMIZATION 句を使用して、オプティマイザがクエリを処理するときにマテリアライズドビューを使用する方法を指定します。指定した `option-value` は、このクエリでのみ materialized_view_optimization データベースオプションを上書きします。`option-value` の可能な値は、materialized_view_optimization database オプションに使用できる値と同じです。

FORCE OPTIMIZATION clause

クエリ指定に単純なクエリしか含まれない場合 (特定のローを識別する WHERE 句に等号条件を含んだ単一ブロック、単一テーブルのクエリ)、通常、処理中にコストベースの最適化はバイパスされます。場合によっては、コストベースの最適化を実行する必要があることがあります。たとえば、クエリ処理時にマテリアライズドビューを考慮する場合、ビューのマッチングが発生します。ただし、ビューの一致が発生するのはコストベースの最適化中のみです。クエリに対してコストベースの最適化を実行しても、クエリ指定に単純なクエリのみを含める場合、FORCE OPTIMIZATION オプションを指定して、オプティマイザが確実にクエリに対してコストベースの最適化を実行できるようにします。

同様に、プロシージャ内の SELECT 文に FORCE OPTIMIZATION オプションを指定すると、プロシージャに対するすべての呼び出しにオプティマイザが強制的に使用されます。この場合、文のプランはキャッシュされません。

FORCE NO OPTIMIZATION clause

FORCE NO OPTIMIZATION 句は、文でオプティマイザをバイパスする場合に指定します。文が複雑すぎて (データベースオプションの設定またはスキーマやクエリの特徴などにより) このような処理が実行できない場合、文は失敗し、データベースサーバはエラーを返します。

option-name = option-value

オプション設定を指定します。指定する設定は、現在の文にのみ適用され、ODBC 実行可能アプリケーションによる設定など、パブリックオプションまたはテンポラリオプションの設定よりも優先されます。

サポートされるオプションは次のとおりです。

- isolation_level オプション
- max_query_tasks オプション
- optimization_goal オプション
- optimization_level オプション
- optimization_workload オプション
- user_estimates オプション
- parameterization_level

クエリで isolation_level オプションを指定すると、そのクエリで指定した値が、現在のクエリの他のどの独立性レベル設定 (データベースの isolation_level オプションの設定やカーソルの設定など) よりも優先されます。

sequence-expression

シーケンスジェネレータから現在の値 (CURRVAL) または次の値 (NEXTVAL) を選択できます。

備考

SELECT 文は、次の目的で使用できます。

- データベースから結果を取得します。
- Interactive SQL で、データベース内のデータをブラウズ、またはデータベースから外部ファイルにデータをエクスポートします。
- プロシージャとトリガ、または Embedded SQL 内で使用します。INTO 句のある SELECT 文を使ってデータベースから結果を取り出します。このとき、SELECT 文はローを 1 つだけ返します。複数のローを対象にクエリを実行する場合は、カーソルを使います。
- プロシージャから結果セットを返します。

権限

SELECT 文で参照されるオブジェクトに対する適切な SELECT 権限を持っている必要があります。

シーケンスジェネレータから CURRVAL 値または NEXTVAL 値を選択するには、USE ANY SEQUENCE システム権限が必要です。または、シーケンスの所有者であるか、シーケンスジェネレータを使用するために必要な権限が付与されている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。個々の句を標準に照らしてチェックします。たとえば、GROUP BY 句で指定できる ROLLUP キーワードは、オプションの ANSI/ISO SQL8 言語機能 T431 の一部です。ソフトウェアでサポートされている、オプションの ANSI/ISO SQL 言語機能の一部を次に示します。

- WINDOW 句と WINDOW 集合関数は、オプションの ANSI/ISO SQL 言語機能 T611 および T612 を構成します。
- シーケンス式は、ANSI/ISO SQL 機能 T176 の一部です。
- 共通テーブル式は、オプションの ANSI/ISO SQL 言語機能 T121 です。ネストされたクエリ式に含まれる共通テーブル式は、機能 T122 です。WITH RECURSIVE は、オプションの ANSI/ISO SQL 言語機能 T131 です。ネストされたクエリに含まれている場合は、機能 T132 を構成します。
- UNION、EXCEPT、または INTERSECT を含むクエリ式とともに ORDER BY 句を指定する機能は、オプションの ANSI/ISO SQL 機能 F850 です。サブクエリで ORDER BY を指定する機能は、ANSI/ISO SQL 機能 F851 です。
- ANSI/ISO SQL 標準では、FOR UPDATE と FOR READ ONLY はカーソル宣言の一部です。

ソフトウェアでは、SELECT 文の ANSI/ISO SQL 定義に対する多くの拡張がサポートされています。その一部を次に示します。

- オプションの `cursor-concurrency` 句 (FOR UPDATE BY { LOCK | VALUES | TIMESTAMP}) は ANSI/ISO SQL 標準の一部ではありません。
- FOR XML 句、OPTION 句、および INTO 句は、ANSI/ISO SQL 標準の一部ではありません。
- ロー制限句は、ANSI/ISO SQL 標準の一部ではありません。ANSI/ISO SQL 標準では、ロー制限は、オプションの言語機能 F856 である FETCH FIRST 構文を使用してサポートされます。機能 F856 の構文はソフトウェアではサポートされていません。
- ORDER BY `n` を指定する機能は、ANSI/ISO SQL 標準の一部ではありません。
- ANSI/ISO SQL 標準では、INSENSITIVE カーソル以外のすべてのカーソルがデフォルトで更新可能になっています。Embedded SQL プログラムでデフォルトで使用される読み込み専用カーソルは、ANSI/ISO SQL 標準の一部ではありません。

Transact-SQL

SQL Anywhere と Adaptive Server Enterprise での SELECT 文のサポートには大きな違いがあります。SELECT 文のいくつかの機能は、Adaptive Server Enterprise ではサポートされていません。

これらの違いを次に示します。

- SAP ASE では、SQL Anywhere の cursor-concurrency 句はサポートされていません。フェッチされたローのロックを取得するには、HOLDLOCK テーブルヒントを使用します。
- Adaptive Server Enterprise では、再帰クエリも共通テーブル式もサポートされていません。
- Adaptive Server Enterprise と SQL Anywhere では、Transact-SQL 外部ジョインに関して違いがあります。

Transact-SQL では、Watcom SQL SET 文ではなく、SELECT 文を使用して値を変数に割り当てます。

例

次のクエリは、Employees テーブルの全従業員数を返します。

```
SELECT COUNT(*)
FROM GROUPO.Employees;
```

次のクエリは、すべての顧客とその顧客からの注文の総額をリストします。

```
SELECT CompanyName,
   CAST( SUM( SalesOrderItems.Quantity *
   Products.UnitPrice ) AS INTEGER ) VALUE
FROM GROUPO.Customers
   JOIN GROUPO.SalesOrders
   JOIN GROUPO.SalesOrderItems
   JOIN GROUPO.Products
GROUP BY CompanyName
ORDER BY VALUE DESC;
```

次の文は、Embedded SQL SELECT 文を示します。Employees テーブル内の従業員数が選択され、:size ホスト変数に格納されます。

```
SELECT COUNT(*) INTO :size
FROM GROUPO.Employees;
```

次の文は、結果セットの最初のローを迅速に返すように最適化されています。

```
SELECT Name
FROM GROUPO.Products
GROUP BY Name
HAVING COUNT( * ) > 1
AND MAX( UnitPrice ) > 10
OPTION( optimization_goal = 'first-row' );
```

次の文は、ロー変数を宣言し、INTO VARIABLE 句を使用して更新されたローの値を返すクエリを実行する、GetCustomer 関数を作成します。

```
CREATE FUNCTION GetCustomer ( @custid Customers.ID%TYPE )
RETURNS Customer%ROWTYPE
BEGIN
   DECLARE @customer Customers%ROWTYPE;
   SELECT * INTO VARIABLE @customer
   FROM Customers
   WHERE id = @custid;
   RETURN @customer;
END;
```

関連情報

SQL 文の式 [32 ページ]
FROM 句 [1112 ページ]
GROUP BY 句 [1152 ページ]
探索条件 [53 ページ]
WINDOW 句 [1409 ページ]

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
UNION 文 [1373 ページ]
EXCEPT 文 [1088 ページ]
INTERSECT 文 [1177 ページ]

1.4.4.247 SET CONNECTION 文 [Interactive SQL] [ESQL]

アクティブなデータベース接続を変更します。

構文

```
SET CONNECTION [ connection-name ]
```

```
connection-name :
identifier
| string
| hostvar
```

備考

SET CONNECTION 文は、アクティブなデータベース接続を connection-name に変更します。現在の接続状態を保存し、再びアクティブな接続になるときにこれを再開します。connection-name が省略され、指定されない接続がある場合は、この接続がアクティブな接続になります。

カーソルを Embedded SQL でオープンするとき、カーソルを現在の接続と関連付けます。接続が変更されると、前のアクティブな接続のカーソル名にはアクセスできません。これらのカーソルはアクティブなまま配置され、関連付けられている接続が再びアクティブになると、アクセスできるようになります。

この SQL 文は SAP HANA データベースではサポートされていません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

SET CONNECTION は、オプションの ANSI/ISO SQL 言語機能 F771、"Connection management" の一部です。Interactive SQL セッション内での使用は、標準にありません。

例

次の例は、Embedded SQL でのフラグメントです。

```
EXEC SQL SET CONNECTION :conn_name;
```

Interactive SQL から、現在の接続を架空の接続 conn1 に設定します。

```
SET CONNECTION conn1;
```

関連情報

CONNECT 文 [ESQL] [Interactive SQL] [774 ページ]
DISCONNECT 文 [ESQL] [Interactive SQL] [1028 ページ]

1.4.4.248 SET DESCRIPTOR 文 [ESQL]

SQL 記述子領域内の変数を記述したり、データを記述子領域に格納したりします。

構文

```
SET DESCRIPTOR descriptor-name
{ COUNT = { integer | hostvar }
| VALUE { integer | hostvar } assignment, ... }
```

```
assignment :
{ TYPE | SCALE | PRECISION | LENGTH | INDICATOR }
      = { integer | hostvar }
| DATA = hostvar
```

```
descriptor-name : identifier
```

備考

SET DESCRIPTOR 文は、記述子領域内の変数を記述したり、データを記述子領域に格納するために使用します。

SET...COUNT 文を使用すると、記述子領域内の記述される変数の数を設定できます。カウントの対象となる変数の数は、記述子領域を割り付けたときに指定した変数の数を超えることはできません。

値 { integer | hostvar } は、代入の対象となる記述子領域内の変数を指定します。

SET...DATA の実行中に型チェックが実行され、記述子領域の変数とホスト変数のデータ型が同じかどうか確認されます。LONG VARCHAR と LONG BINARY は SET DESCRIPTOR...DATA ではサポートされていません。

エラーが発生すると、エラーコードが SQLCA に返されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

SET DESCRIPTOR は、オプションの ANSI/ISO SQL 言語機能 B031、"Basic dynamic SQL" の一部です。

例

次の例は、sqlda の位置 col_num にあるカラムの型を設定します。

```
void set_type( SQLDA *sqlda, int col_num, int new_type )
{
    EXEC SQL BEGIN DECLARE SECTION;
    INT new_type1 = new_type;
    INT col = col_num;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL SET DESCRIPTOR sqlda VALUE :col TYPE = :new_type1;
}
```

関連情報

ALLOCATE DESCRIPTOR 文 [ESQL] [628 ページ]
DEALLOCATE DESCRIPTOR 文 [ESQL] [999 ページ]

1.4.4.249 SET MIRROR OPTION 文

データベースミラーリングと読み込み専用のスケールアウトの設定を制御するオプションの値を変更します。

構文

SET MIRROR OPTION `option-name={ option-value | NULL }`

`option-name` :
authentication_string
auto_add_fan_out
auto_add_server
auto_failover
child_creation
max_disconnected_time
max_logfile_size
max_retry_connect_time
page_timeout
promotion_time
synchronization_mode
lagtime

パラメータ

NULL

オプションのデフォルト値を指定します。`option-name` を NULL に設定すると、オプション値がデフォルト値に設定されます。

option-name	適用対象	値	デフォルト	説明
authentication_string	データベースミラーリング	string	null	データベースミラーリングシステムのすべてのサーバで使用される認証文字列を指定します。データベースミラーリングには、認証文字列は必須です。
auto_add_fan_out	読み込み専用のスケールアウト	integer	10	分岐ごとの子の最大数を指定します。指定できる最小の値は 2 です。
auto_add_server	読み込み専用のスケールアウト	string	null	自動割り当てツリーの親として動作するデータベースサーバの名前を指定します。

option-name	適用対象	値	デフォルト	説明
auto_failover	データベースミラーリング	on, off	null	現在のプライマリサーバで障害が発生した場合にミラーサーバが自動的にプライマリサーバになるかどうかを指定します。このオプションは同期モードには適用されません。 このオプションでは、ブール値を使用できます (YES、ON、TRUE、または 1 の場合は自動フェイルオーバーがオンになり、NO、OFF、FALSE、または 0 の場合は自動フェイルオーバーがオフになります)。パラメータは大文字と小文字が区別されません。 非同期モードまたは非同期フルページモードを使用している場合は、auto_failover オプションを on に設定してください。それによって、プライマリサーバで障害が発生した場合、ミラーサーバが自動的にプライマリサーバとなります。
child_creation	読み込み専用のスケールアウト	automatic, off, manual	automatic	コピーノードを自動的に作成するかどうかを制御します。
max_disconnected_time	読み込み専用のスケールアウト	integer 以上の整数 (秒) または max_retry_connect_time 以上の整数 (秒)	時間制限なし	前回コピーノードが親、代替の親、またはルートデータベースに接続されてからコピーノードが停止するまでの時間を指定できるようになりました。

option-name	適用対象	値	デフォルト	説明
max_logfile_size	データベースミラーリングと読み込み専用のスケールアウト	integer とオプションの K、M、G サフィックス	最大サイズの制限なし	ミラーデバッグログファイルの最大サイズをバイト単位で指定します。キロバイト、メガバイト、またはギガバイトで指定するには、K、M、G のいずれかを使用してください。最小サイズは 10 KB です。 ログファイルが最大サイズに達すると、ファイルが拡張子 .old の付いた名前に変更され、新しいファイルが作成されます。 ミラーデバッグログファイルを指定するには、CREATE MIRROR SERVER 文または ALTER MIRROR SERVER 文に *logfile*server-option を設定します。
max_retry_connect_time	読み込み専用のスケールアウト	integer、秒単位	120	親が使用できなくなった後、コピーノードが親に再接続を試みる期間を指定します。
page_timeout	データベースミラーリング	integer、秒単位	5	トランザクションログページを満杯かどうかにかかわらずミラーサーバに送信する間隔を秒単位で指定します。このオプションは、非同期フルページモードを使用している場合にのみ適用されます。

option-name	適用対象	値	デフォルト	説明
promotion_time	読み込み専用のスケールアウト	integer 以上の整数 (秒) または max_retry_connect_time 以上の整数 (秒)	3600	親の接続が切れた後、プロンプトを表示しないでコピーノードをルートデータベースサーバに接続された状態にする時間を指定します (スケールアウトツリーを調整して、親の接続が切断されないようにします)。このオプションは、プライマリデータベースの負荷を長時間増大させることなく、コピーノードが短時間ダウンした場合 (浅いスケールアウトツリーとなります) のスケールアウトツリーの調整を不要にするのに役立ちます。プロンプトを表示しないようにするには、このオプションを 315,360,000 (10 年) 以上に設定します。 このオプションは、バージョン 16 でのみサポートされています。
synchronization_mode	データベースミラーリング	synchronous, asynchronous, asyncfullpage	synchronous	データベースミラーリングで使用する同期 (sync)、非同期 (async)、または非同期フルページ (page) のいずれかの同期モードを指定します。同期モードでは、ミラーサーバでトランザクションを記録するタイミングと方法を制御します。

option-name	適用対象	値	デフォルト	説明
ラグタイム	データベースミラーリング	integer、秒単位	60	トランザクションログの適用時おける、プライマリサーバに対するミラーサーバのおよそのラグタイムを指定します。ラグタイムが設定した値に近づくと、プライマリサーバはトランザクションのレートを減少させます。0 または、15 および 3600 秒 (1 時間) の間のいずれかの整数に値を設定します。値 0 は時間無制限に相当します。

備考

CREATE MIRROR SERVER 文を使用して、データベースミラーリングシステムまたは読み込み専用のスケールアウトシステム用のデータベースサーバを作成した後、SET MIRROR OPTION 文を使用してそのシステムの設定を指定できます。

読み込み専用のスケールアウトとデータベースミラーリングには、それぞれ別途ライセンスが必要です。

権限

サーバを起動しているか、MANAGE ANY MIRROR SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、データベースミラーリングシステムの認証文字列を abc に設定します。

```
SET MIRROR OPTION authentication_string = 'abc';
```

関連情報

CREATE MIRROR SERVER 文 [858 ページ]
ALTER MIRROR SERVER 文 [662 ページ]
DROP MIRROR SERVER 文 [1047 ページ]
SYSMIRROROPTION システムビュー [1817 ページ]

1.4.4.250 SET OPTION 文

データベースオプションと接続オプションの値を変更します。

構文

オプションの指定

```
SET [ EXISTING ] [ TEMPORARY ] OPTION
  [ userid.| PUBLIC.]option-name = [ option-value ]
```

識別子の指定 (廃止予定)

```
SET [ EXISTING ] [ TEMPORARY ] OPTION
  [ userid.| PUBLIC.]option-name = [ identifier ]
```

```
userid : identifier
```

```
option-name : identifier
```

```
option-value :ON, OFF, NULL、string literal、number、hostvar、または @variable-
name
```

パラメータ

option-value

オプションを指定する構文を使用する場合、option-value には次のいずれかを指定できます。

- キーワード ON、OFF、または NULL
- string literal 値 (一重引用符で囲む)
- 任意の有効なフォーマット (NUMERIC など) の number
- Embedded SQL プログラム内では、ホスト変数 hostvar の値
- @ 記号で始まる変数名を持つ SQL 変数の値

識別子を指定する構文を指定する場合、任意の有効な識別子をオプション値として指定できます。また、識別子の名前は、一重引用符で囲んだ文字列リテラルであるかのように扱われます。たとえば、次の 2 つの文は同じです。

```
SET TEMPORARY OPTION ansi_update_constraints = 'strict';
```

```
SET TEMPORARY OPTION ansi_update_constraints = strict;
```

備考

SET OPTION 文を使用すると、データベースサーバの動作に影響を及ぼすオプションを変更できます。オプションの値を設定すると、すべてのユーザ (パブリック)、個々のユーザ (ユーザ拡張ロールを含む)、または現在の接続の動作を変更できます。新しい設定は、テンポラリ設定にも永久設定にもできます。

ロールに対するオプションの設定は、ユーザ拡張ロールの場合にのみ意味があり、ユーザがログインしている場合にのみ適用されます。ロールを継承するユーザは、オプション設定は継承しません。

SET OPTION 文で設定できるオプションのクラスは次のとおりです。

- Transact-SQL 互換性オプション
- 接続オプションとデータベースオプション
- 同期オプション
- ユーザ定義のオプション

  オプションのスコープ

  ほとんどのオプションでは、3 つのレベルのスコープ (パブリック、ユーザ、接続) で値を設定できます。login_mode などの一部の特定のオプションはパブリックレベルのみに制限されます。接続オプションは他の 2 つのレベルよりも優先され、ユーザオプションはパブリックオプションよりも優先されます。接続レベルのオプションは、TEMPORARY キーワードを使用して設定します。現在のユーザに対してユーザレベルのオプションを設定すると、対応する接続レベルのオプションも同時に設定されます。

  デフォルトでは、オプション値は SET OPTION 文を実行した現在ログオン中のユーザ ID に適用されます。ユーザ ID を指定すると、そのユーザにオプション値が適用されます。PUBLIC に指定すると、オプション値はそのオプションに個々の設定を持たないすべてのユーザに適用されます。

  TEMPORARY オプション

  デフォルトでは、TEMPORARY キーワードを指定しないかぎり、新しいオプション値は永久値になります。SET OPTION 文に TEMPORARY キーワードを追加すると、変更の適用期間に影響します。

  SET TEMPORARY OPTION 文がユーザ ID で修飾されていない場合、新しいオプション値は現在の接続のみで有効です。

  PUBLIC ロールを使用して SET TEMPORARY OPTION を使用すると、データベースの実行中はその変更が継続します。データベースを停止すると、PUBLIC ロールの TEMPORARY オプションは永久値に戻ります。

PUBLIC ロールのテンポラリオプションを設定すると、セキュリティの利点を提供します。たとえば、login_mode オプションが有効なときは、データベースは実行中のシステムのログインセキュリティに依存します。このオプションを一時的に有効にすると、Windows ドメインのセキュリティに依存しているデータベースは、データベースが停止し、ローカルコンピュータにコピーされるといった場合でも、危険にさらされることはありません。この場合、login_mode オプションを一時的に有効にしてあると、オプションは永久値の Standard に戻ります。このモードでは、統合化ログインを使用できません。

オプション設定の削除

option-value を省略すると、指定されたオプション設定がデータベースから削除されます。これがユーザレベルのオプション設定の場合、値は PUBLIC 設定に戻ります。TEMPORARY オプションを削除すると、オプション設定はそのユーザの永久設定に戻ります。

オプションのデータ型

オプションは、ブール値、数値、または文字列値にできますが、データベースには常に文字列として格納されます。オプション設定は、プロパティ関数の結果として返される場合や、関数またはシステムストアドプロシージャの結果として返される場合、常に文字列として返されます。オプション値は、データベースページサイズよりも大きい値には設定できません。

ユーザ定義のオプション

どのようなオプションでも、ユーザ定義であるかどうかにかかわらず、パブリック設定がなければユーザ固有の値を割り当てることはできません。データベースサーバでは、ユーザ定義オプションへの TEMPORARY 値の設定はサポートされていません。たとえば、ApplicationControl という名前のユーザ定義のオプションを作成するには、まず次の文を実行します。

```
SET OPTION PUBLIC.ApplicationControl = 'Default';
```

この文は、すべてのユーザに対して ApplicationControl オプションを Default に設定し、サーバへの新しい接続ごとに有効にします。その後、個々のユーザで別個に SET OPTION 文を実行して、このオプションの独自の設定を確立できます。

制限

EXISTING キーワードを使用した場合、個別のユーザ ID のオプション値は、そのオプションにすでに PUBLIC 設定が行われていないかぎり、設定できません。

> ⚠ 警告
>
> カーソルが開いている間はオプション値を変更しないでください。カーソルが開いている間にオプション値を変更した場合、カーソルの中で結果の不整合が生じる可能性があります。たとえば、date_format オプションを変更した場合、返されるローの一部の形式が古い形式になったり新しい形式になったりすることがあります。新しいオプション値を使用して結果セット内のローが整合性を保って計算されるように、オプション値を変更してからカーソルを開いてください。

接続またはユーザの特定のオプションの値に対するクエリを実行するには、いくつかの方法があります。

SET OPTION 文は、SQL Flagger で無視されます。

権限

どのユーザも自分の独自のオプションを設定できます。

PUBLIC ロールを含む他のユーザまたはロールに対してデータベースオプションを設定する場合は、オプションで必要になる権限に応じて次のいずれかのシステム権限が必要です。

- SET ANY SYSTEM OPTION
- SET ANY PUBLIC OPTION
- SET ANY SECURITY OPTION
- SET ANY USER DEFINED OPTION

関連する動作

TEMPORARY を指定しない場合、オートコミットが実行されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

個別に設定することなく、すべてのユーザの日付形式オプションを設定します。

```
SET OPTION PUBLIC.date_format = 'Mmm dd yyyy';
```

wait_for_commit オプションを On に設定します。

```
SET OPTION wait_for_commit = 'On';
```

次のフラグメントは、Embedded SQL の例です。

```
EXEC SQL SET TEMPORARY OPTION date_format = :value;
```

現在接続されているユーザに対して date_format オプションを設定します。同じユーザ ID による以降の接続では、このオプション値が使用されます。

```
SET OPTION date_format = 'yyyy/mm/dd';
```

次の文は、現在のユーザ ID の date_format オプションを削除します。この文の実行後、PUBLIC の date_format の設定が代わりに使用されます。

```
SET OPTION date_format=;
```

次の文は、すべてのユーザに対する login_mode オプションを標準に変更します。

```
SET OPTION PUBLIC.login_mode = 'Standard';
```

関連情報

SYSOPTION システムビュー [1822 ページ]
sa_conn_options システムプロシージャ [1469 ページ]
sa_conn_options システムプロシージャ [1469 ページ]
CONNECTION_PROPERTY 関数 [システム] [281 ページ]
GET OPTION 文 [ESQL] [1128 ページ]
SET OPTION 文 [Interactive SQL] [1314 ページ]
SET 文 [T-SQL] [1327 ページ]
SET REMOTE OPTION 文 [SQL Remote] [1316 ページ]

1.4.4.251 SET OPTION 文 [Interactive SQL]

Interactive SQL オプションの値を変更します。

構文

Interactive SQL オプションの設定

```
SET OPTION option-name = [ option-value ] | SET TEMPORARY OPTION option-name =
[ option-value ]
```

```
option-name : identifier | string | hostvar
```

```
option-value : string | identifier | number
```

現在の Interactive SQL オプションの永久的な保存

```
SET PERMANENT
```

現在のデータベースオプション設定の表示

```
SET
```

備考

SET OPTION 構文を使用してオプションを設定すると、そのオプション設定は永久的に格納され、他の SET OPTION 文によって変更されるまで変わりません。

SET TEMPORARY OPTION 構文を使用すると、オプション設定を一時的に変更できます。一時的な設定は、Interactive SQL を閉じるまで有効です。Interactive SQL を次回起動するとき、オプションはその恒久的な設定に戻ります。

SET PERMANENT 構文を使用すると、現在の Interactive SQL オプション設定がすべて永久的に保存されます (一時的な設定もすべて永久的になります)。

option-value を省略すると、指定したオプションはデフォルト値に設定されます。

現在のデータベースオプション設定を表示する際、データベースサーバにテンポラリオプション設定がある場合、永久設定ではなく、テンポラリ設定が表示されます。

Interactive SQL のオプション設定は、データベースにではなく、クライアントコンピュータに保存されます。

次の表は、Interactive SQL のオプションを示します。

オプション	値	デフォルト
auto_commit オプション [Interactive SQL]	On、Off	Off
auto_refetch オプション [Interactive SQL]	On、Off	On
bell オプション [Interactive SQL]	On、Off	On
command_delimiter オプション [Interactive SQL]	String	' ; '
commit_on_exit オプション [Interactive SQL]	On、Off	On
default_isql_encoding オプション [Interactive SQL]	String	空の文字列
echo オプション [Interactive SQL]	On、Off	On
input_format オプション [Interactive SQL]	TEXT、FIXED	TEXT
isql_allow_read_client_file オプション [Interactive SQL]	On、Off、Prompt	Prompt
isql_allow_write_client_file オプション [Interactive SQL]	On、Off、Prompt	Prompt
isql_command_timing オプション [Interactive SQL]	On、Off	On
isql_escape_character オプション [Interactive SQL]	Character	' ¥ '
isql_field_separator オプション [Interactive SQL]	String	' , '
isql_maximum_displayed_rows オプション [Interactive SQL]	All または負でない整数	500
isql_print_result_set オプション [Interactive SQL]	Last、All、None	Last
isql_quote オプション [Interactive SQL]	String	'
isql_show_multiple_result_sets オプション [Interactive SQL]	On、Off	Off
nulls オプション [Interactive SQL]	String	'(NULL)'
on_error オプション [Interactive SQL]	Stop、Continue、Prompt、Exit、Notify_Continue、Notify_Stop、Notify_Exit	Prompt

オプション	値	デフォルト
output_format オプション [Interactive SQL]	TEXT、FIXED、HTML、SQL、XML	TEXT
output_length オプション [Interactive SQL]	Integer	0
output_nulls オプション [Interactive SQL]	String	空の文字列
truncation_length オプション [Interactive SQL]	Integer	256

権限

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、on_error オプションの値を変更して続けます。

```
SET OPTION on_error='continue';
```

1.4.4.252 SET REMOTE OPTION 文 [SQL Remote]

SQL Remote メッセージリンクのメッセージ制御パラメータを設定します。

構文

```
SET REMOTE message-system OPTION
[ userid.| PUBLIC.]option-name = option-value
```

```
message-system :
FILE
| FTP
| HTTP
| SMTP
```

```
option-name :
common-options
```

```
| file-options
| ftp-options
| smtp-options
```

```
common-options :
debug
| encode_dll
| max_retries
| output_log_send_on_error
| output_log_send_limit
| output_log_send_now
| pause_after_failure
```

```
file-options :
directory
| invalid_extensions
| unlink_delay
```

```
ftp-options :
active_mode
| host
| invalid_extensions
| password
| port
| root_directory
| reconnect_retries
| reconnect_pause
| suppress_dialogs
| user
```

```
http-options :
| certificate
| client_port
| https
| password
| proxy
| reconnect_retries
| reconnect_pause
| root_directory
| url
| user
```

```
smtp-options :
 local_host
| pop3_host
| pop3_password
| pop3_port
| pop3_userid
| smtp_authenticate
| smtp_option
| smtp_password
| smtp_port
| smtp_userid
| suppress_dialogs
| top_supported
```

```
option-value : string
```

パラメータ

userid

`userid` を指定しない場合、現在のパブリッシャが使用されます。

common-options

FILE、FTP、HTTP、SMTP メッセージシステムに共通のオプションは、次のとおりです。

debug

このパラメータは、YES または NO のいずれかに設定されます。デフォルトは NO です。YES を設定すると、メッセージシステムに固有のデバッグ出力が表示されます。この情報は、メッセージシステムのトラブルシューティングに使用できます。

max_retries

デフォルトでは、SQL Remote が継続モードで実行されていて、メッセージシステムにアクセスするときにエラーが発生する場合、送受信フェーズの後に停止します。このパラメータを使用して、停止前に SQL Remote が再試行する送受信フェーズの回数を指定します。

output_log_send_on_error

エラーが発生すると、情報を送信します。

output_log_send_limit

統合データベースに送信される情報の量を制限します。output_log_send_limit オプションには、統合データベースに送信されるバイト数を指定します。これは、出力ログの最後に表示されます (つまり、最後のエントリです)。デフォルトは 5K です。

output_log_send_now

YES に設定されると、統合データベースに出力ログ情報を送信します。次のポーリング時にリモートデータベースは出力ログ情報を送信し、その後、output_log_send_now オプションが NO にリセットされます。

pause_after_failure

このパラメータが使えるのは、max_retries がゼロ以外の値に指定され、SQL Remote が継続モードで実行されている場合です。メッセージシステムでエラーが発生すると、このパラメータは、SQL Remote が送受信フェーズを再試行する間に待機する時間を秒で指定します。

encode_dll

カスタムエンコードスキームを実装している場合は、作成したカスタムエンコード DLL のフルパスをこの値に設定する必要があります。

file-options

これらのオプションは、ファイルメッセージシステムにのみ適用されます。

directory

メッセージが格納されるディレクトリです。このパラメータは、SQLREMOTE 環境変数の代替となります。

invalid_extensions

メッセージングシステムでのファイルの生成時に SQL Remote Message Agent (dbremote) で使用されないようにするファイル拡張子の、カンマで区切られたリスト。

unlink_delay

ファイル削除に失敗した後、次にファイルの削除を試みるまでの秒数。unlink_delay に対して値が定義されていない場合、デフォルト動作は、最初の試行失敗の後に 1 秒間、2 回目の試行失敗の後に 2 秒間、3 回目の試行失敗の後に 3 秒間、4 回目の試行失敗の後に 4 秒間一時停止するように設定されています。

ftp-options

これらのオプションは、FTP メッセージシステムにのみ適用されます。

active_mode

このパラメータは、SQL Remote がクライアント/サーバ接続を確立する方法を制御します。このパラメータは、YES または NO のいずれかに設定されます。デフォルトは NO (受動モード) です。受動モードは推奨の転送モードで、FTP メッセージシステムにデフォルトとして設定されています。受動モードでは、すべてのデータ転送接続はクライアント (この場合はメッセージシステム) から開始されます。アクティブモードでは、FTP サーバがすべてのデータ接続を開始します。

host

FTP サーバを実行しているコンピュータのホスト名。このパラメータには、ホスト名 (ftp.sap.com など) または IP アドレス (192.138.151.66 など) を使用できます。

invalid_extensions

メッセージングシステムでのファイルの生成時に dbremote で使用されないようにするファイル拡張子の、カンマで区切られたリスト。

password

FTP ホストにアクセスするためのパスワード。

port

FTP の接続に使用される IP ポート番号。このパラーメータは通常は不要です。

reconnect_retries

失敗になる前に、メッセージシステムがサーバでソケットを開こうと試行する回数。デフォルト値は 4 です。このパラメータを設定すると、再接続のみに影響します。FTP メッセージシステムによる最初の接続には影響しません。

reconnect_pause

接続の試行失敗後に次の接続まで待機する秒数。デフォルトは 30 秒です。このパラメータを設定すると、再接続のみに影響します。FTP メッセージシステムによる最初の接続には影響しません。

root_directory

メッセージが保存される、FTP ホストサイトのルートディレクトリ。

suppress_dialogs

このパラメータは、TRUE または FALSE に設定されます。TRUE に設定されている場合は、FTP サーバへの接続の試行失敗後に、接続ウィンドウは表示されません。代わりに、エラーが発生します。

user

FTP ホストにアクセスするためのユーザ名。

http-options

これらのオプションは、HTTP メッセージシステムにのみ適用されます。

certificate

安全な (HTTPS) 要求を行うには、HTTPS サーバで使用される証明書にクライアントがアクセスできる必要があります。必要な情報は、セミコロンで区切られたキーワード/値のペアの文字列で指定されます。ファイルキーワードを使用して証明書のファイル名を指定できます。ファイルキーワードと証明書キーワードを一緒に指定することはできません。次のキーワードを使用できます。

キーワード	省略形	説明
file		証明書のファイル名
certificate	cert	証明書自体
company	co	証明書で指定された会社
unit		証明書で指定された会社の部署
name		証明書で指定された通称

証明書は、HTTPS サーバに対する要求、または安全でないサーバから安全なサーバにリダイレクトされる可能性がある要求に対してのみ必要です。PEM でフォーマットされた証明書のみがサポートされています (たとえば、certificate='`file=filename`'

SQL Anywhere データベース内で証明書名を作成するには:

```
CREATE OR REPLACE CERTIFICATE certificate_name FROM FILE 'certificate_file';
```

HTTPS メッセージタイプの証明書名を使用するには:

```
SET REMOTE HTTP OPTION user_name.certificate= 'cert_name=certificate_name';
```

client_port

SQL Remote が HTTP を使用して通信するポート番号を識別します。これは "送信" TCP/IP 接続をフィルタするファイアウォール経由で接続するためのもので、この用途にかぎり推奨されています。単一のポート番号、ポート番号の範囲、または両方の組み合わせを指定できます。指定したクライアントポートの数が少ないと、SQL Remote が前回の実行でポートを閉じた後すぐにオペレーティングシステムがポートを解放しなかった場合に、SQL Remote でメッセージの送受信ができなくなる可能性があります。

debug

YES に設定すると、すべての HTTP コマンドと応答が出力ログに表示されます。この情報は、HTTP のサポート問題のトラブルシューティングに使用できます。デフォルトは NO です。

https

HTTPS (**`https=yes`**) または HTTP (**`https=no`**) を使用するかどうかを指定します。

password

メッセージサーバのデータベースパスワード。パスワードは、RFC 2617 の基本認証を使用してサードパーティの HTTP サーバとゲートウェイに対する認証を行います。

proxy_host

プロキシサーバの URI を指定します。SQL Remote がプロキシサーバを介してネットワークにアクセスする場合に使用します。SQL Remote がプロキシサーバに接続し、そのプロキシサーバを介してメッセージサーバに要求を送信することを示します。

reconnect_retries

失敗になる前に、メッセージシステムがサーバでソケットを開こうと試行する回数。デフォルト値は 4 です。このパラメータを設定すると、再接続のみに影響します。FTP メッセージシステムによる最初の接続には影響しません。

reconnect_pause

接続の試行失敗後に次の接続まで待機する秒数。デフォルトは 30 秒です。このパラメータを設定すると、再接続のみに影響します。FTP メッセージシステムによる最初の接続には影響しません。

root_directory

この HTTP 制御パラメータは、クライアント側で指定された場合には無視されます。sr_add_message_server または sr_update_message_server ストアドプロシージャを呼び出す前に、メッセージサーバでこの制御パラメータを定義します。HTTP メッセージシステムを使用する場合、リモートユーザまたはパブリッシャに指定するアドレスには、1 つのサブディレクトリのみを含めることができます。複数のサブディレクトリを含めることはできません。

url

サーバ名または IP アドレスを指定し、任意で、使用する HTTP サーバのポート番号をセミコロンで区切って指定します。要求が Relay Server を経由する場合は、URL 拡張子を任意で追加して、要求の渡し先のサーバファームを示すことができます。

user

メッセージサーバのデータベースユーザ ID。RFC 2617 の基本認証を使用してサードパーティの HTTP サーバとゲートウェイに対する認証を行います。

smtp-options

これらのオプションは、SMTP メッセージシステムにのみ適用されます。

local_host

ローカルコンピュータの名前。SQL Remote がローカルホスト名を決定できないコンピュータで設定すると便利です。ローカルホスト名は、任意の SMTP サーバとのセッションを開始するのに必要です。ほとんどのネットワーク環境では、ローカルホスト名が自動的に決定されるため、このエントリは不要です。

pop3_host

POP ホストを実行しているコンピュータの名前。一般的には、SMTP ホストと同じ名前です。SMTP/POP3 ログインウィンドウの POP3 ホストフィールドに対応しています。

pop3_password

メールの受信に使用するパスワード。SMTP/POP3 ログインウィンドウのパスワードフィールドに対応しています。

pop3_port

POP サーバの受信対象ポートの番号。デフォルトは 110 です。SMTP/POP3 ログインウィンドウのポートフィールドに対応しています。

pop3_userid

メールの受信に使用するユーザ ID。POP ユーザ ID は、SMTP/POP3 ログインウィンドウのユーザ ID フィールドに対応しています。必ず POP ホスト管理者からユーザ ID を取得してください。

smtp_host

SMTP サーバが動作しているコンピュータの名前。SMTP/POP3 ログインウィンドウの SMTP ホストフィールドに対応しています。

top_supported

受信メッセージを列挙するときに、SQL Remote は TOP という POP3 コマンドを使用します。TOP コマンドは、すべての POP サーバでサポートされているわけではありません。top_supported パラメータを NO に設定すると、SQL

Remote は RETR コマンドを使用します。このコマンドは TOP よりも効率は落ちますが、すべての POP サーバで動作します。デフォルトは YES です。

smtp_authenticate

SMTP メッセージシステムがユーザを認証するかどうかを決定します。デフォルト値は YES です。SMTP 認証を無効にする場合は、このパラメータを NO に設定します。

smtp_userid

SMTP 認証のユーザ ID。デフォルトでは、このパラメータは pop3_userid パラメータと同じ値を取ります。smtp_userid は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

smtp_password

SMTP 認証のパスワード。デフォルトでは、このパラメータは pop3_password パラメータと同じ値を取ります。smtp_password は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

smtp_port

SMTP サーバの現在の受信対象ポートの番号。デフォルトは 25 です。SMTP/POP3 ログインウィンドウのポートフィールドに対応しています。

suppress_dialogs

このパラメータが true に設定されている場合は、メールサーバへの接続の試行失敗後に、接続ウィンドウは表示されません。代わりに、エラーが発生します。

備考

メッセージシステムの初回使用時に、ユーザがメッセージシステムパラメータを [メッセージシステム] ウィンドウに入力すると、SQL Remote (dbremote) Message Agent はそのパラメータを保存します。その場合はこの文を明示的に使用する必要はありません。この文は、たくさんのデータベースから抽出を行うための統合データベースを作成する場合に非常に効果的です。

オプション名では、大文字と小文字が区別されます。オプションの大文字小文字の区別は、オプションによって異なります。Boolean 値では大文字と小文字が区別されず、パスワード、ディレクトリ、その他の文字列の大文字と小文字の区別は、ファイルシステム (ディレクトリ名の場合) またはデータベース (ユーザ ID とパスワードの場合) の大文字と小文字の区別に依存します。

権限

パブリッシャは自分自身のオプションを設定できます。それ以外の場合、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザ Sam_Singer 用の FTP リンクに対し、FTP ホストを `ftp.mycompany.com` に設定します。

```
SET REMOTE FTP OPTION Sam_Singer.host = 'ftp.mycompany.com';
```

次の文は、生成されるメッセージの特定ファイル拡張子を SQL Remote が使用しないようにします。

```
SET REMOTE FTP OPTION "PUBLIC"."invalid_extensions"='exe,pif,dll,bat,cmd,vbs';
```

次の文は、ユーザ Sam_Singer の HTTP リンクのローカルホストを示す URL を設定します。

```
SET REMOTE HTTP OPTION Sam_Singer.url='localhost:8033';
```

次の文は、要求を srhttp ファームに転送する Relay Server を指すように HTTP URL を設定します。

```
SET REMOTE HTTP OPTION PUBLIC.url='iis7.company.com:80/rs/client/rs.dll/srhttp';
```

関連情報

SET OPTION 文 [1310 ページ]
CREATE CERTIFICATE 文 [779 ページ]

1.4.4.253 SET SQLCA 文 [ESQL]

デフォルトのグローバル `sqlca` 以外の SQLCA を使用するように、SQL プリプロセッサに通知します。

構文

```
SET SQLCA sqlca
```

```
sqlca : identifier | string
```

備考

SET SQLCA 文は、デフォルトのグローバル `sqlca` 以外の SQLCA を使うように、SQL プリプロセッサに通知します。sqlca は、SQLCA ポインタに対する C 言語リファレンスである識別子または文字列を指定します。

Embedded SQL 文ごとに、現在の SQLCA ポインタを暗黙のうちにデータベースインタフェースライブラリに渡します。C 言語ソースにおいて、この文の後に記述されているすべての Embedded SQL 文は、新しい SQLCA を使います。

この文を使用するのは、再入可能コードを記述するときのみです。

sqlca は、ローカル変数を参照しなければなりません。グローバルまたはモジュール静的変数は別のスレッドによる修正を受けます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

Windows DLL には独自の関数を含めることもできます。DLL を使用する各アプリケーションは独自の SQLCA を持ちます。

```
an_sql_code FAR PASCAL ExecuteSQL( an_application *app, char *com )
{
   EXEC SQL BEGIN DECLARE SECTION;
   char *sqlcommand;
   EXEC SQL END DECLARE SECTION;
   EXEC SQL SET SQLCA "&app->.sqlca";
   sqlcommand = com;
   EXEC SQL WHENEVER SQLERROR CONTINUE;
   EXEC SQL EXECUTE IMMEDIATE :sqlcommand;
return( SQLCODE );
}
```

1.4.4.254 SET 文

SQL 変数に値を代入します。

構文

```
SET [ owner.]identifier = expression
```

備考

SET 文は新しい値を変数に代入します。この変数は、CREATE VARIABLE 文または DECLARE 文を使って事前に作成されたものであるか、プロシージャの OUTPUT パラメータであることが必要です。この変数名は、オプションとして、名前の前に @ を付ける Transact-SQL の規則を使用できます。例:`SET @localvar = 42.`

変数は、カラム名を使用できる場所なら SQL 文のどこででも使うことができます。変数と同じ名前のカラム名がある場合は、カラム値を使用できます。

`owner` は、所有しているデータベーススコープ変数でのみ指定できます。

変数は、Embedded SQL プログラムから INSERT または UPDATE 文の対象となるサイズの大きなテキストまたはバイナリオブジェクトを作成するために必要です。これは、Embedded SQL のホスト変数のサイズが 32767 バイトに制限されているためです。

変数は、現在の接続に固有でローカルなもので、データベースとの接続を切断したり、DROP VARIABLE 文を使用すると自動的に消えます。変数は、COMMIT 文や ROLLBACK 文の対象になりません。

ただし、データベーススコープ変数を設定した場合は、接続を切断した後も存続します。データベースが再起動すると、データベーススコープ変数の値は NULL またはデフォルト (定義されている場合) に戻されます。SYSDATABASEVARIABLE システムビューには、すべてのデータベーススコープ変数のリストとそれらの初期値が含まれます。

別のユーザが所有するデータベーススコープ変数は設定できません。

権限

自分が所有するデータベーススコープ変数の設定には、権限は不要です。PUBLIC が所有するデータベーススコープ変数の設定には、UPDATE PUBLIC DATABASE VARIABLE システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

オプションの言語機能 P002、"Computational completeness" の一部。

Transact-SQL

SET 文は SAP Adaptive Server Enterprise でサポートされています。Adaptive Server Enterprise では、1 つの SET 文で個々の割り当て句をカンマで区切って、複数の変数に値を割り当てることができます。

例

この単純な例では、birthday という変数の作成を示し、日付を CURRENT DATE に設定します。

```
CREATE VARIABLE @birthday DATE;
SET @birthday = CURRENT DATE;
```

次のコードフラグメントは、データベースに大きいテキスト値を挿入します。

```
size_t  size;
FILE *  fp;
EXEC SQL BEGIN DECLARE SECTION;
DECL_VARCHAR( 5000 ) buffer;
EXEC SQL END DECLARE SECTION;
fp = fopen( "blob.dat", "r" );
EXEC SQL CREATE VARIABLE hold_blob LONG VARCHAR;
EXEC SQL SET hold_blob = '';
for(;;) {
    size = fread( (void *)buffer.array, 1, 5000, fp );
    if( size <= 0 ) break;
    buffer.len = (a_sql_ulen) size;
    EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES( 1, hold_blob );
EXEC SQL COMMIT;
EXEC SQL DROP VARIABLE hold_blob;
fclose( fp );
```

関連情報

SQL 変数 [118 ページ]
CREATE VARIABLE 文 [992 ページ]
DECLARE 文 [1010 ページ]
SET 文 [T-SQL] [1327 ページ]
DROP VARIABLE 文 [1085 ページ]
SQL 文の式 [32 ページ]

1.4.4.255 SET 文 [T-SQL]

Adaptive Server Enterprise との互換性を保つ方法で現在の接続にデータベースオプションを設定します。

構文

```
SET option-name option-value
```

備考

使用可能なオプションは以下のとおりです。

オプション名	オプション値
ansinull	On または Off
ansi_permissions	On または Off
close_on_endtrans	On または Off
datefirst	1、2、3、4、5、6、または 7 このオプションの設定は、DATEPART 関数で weekday 値を取得するときに影響を与えます。
quoted_identifier	On または Off
rowcount	integer
self_recursion	On または Off
string_rtruncation	On または Off
textsize	integer
トランザクションの独立性レベル	0、1、2、3、snapshot、statement snapshot、または read only statement snapshot

SQL Anywhere のデータベースオプションは、SET OPTION 文を使用して設定します。ただし、互換性を確保するため、便利なオプションについては SQL Anywhere でも Adaptive Server Enterprise の SET 文がサポートされています。

次のオプションは、SQL Anywhere と Adaptive Server Enterprise で Transact-SQL SET 文を使用して設定できます。

SET ansinull

SQL Anywhere と Adaptive Server Enterprise では、値を NULL と比較するデフォルトの動作が異なります。ansinull を Off に設定する場合、NULL との比較に対して Transact-SQL との互換性が提供されます。

SQL Anywhere では、次の構文もサポートされています。

```
SET ansi_nulls { On | Off }
```

SET ansi_permissions

カラム参照を含む UPDATE または DELETE を実行するのに必要な権限に関する、SQL Anywhere と Adaptive Server Enterprise のデフォルト動作は異なります。ansi_permissions を Off に設定すると、UPDATE と DELETE の権限に関して Transact-SQL との互換性が提供されます。

SET close_on_endtrans

トランザクションの終わりでカーソルを閉じるデフォルト動作は、SQL Anywhere と Adaptive Server Enterprise で異なります。close_on_endtrans を Off に設定すると、Transact-SQL との互換性が提供されます。

SET datefirst

デフォルトは 7 です。これは、週の開始日が日曜日であることを意味します。

SET quoted_identifier

二重引用符で囲まれた文字列を識別子 (On) として解釈するか、単なる文字列 (Off) として解釈するかを制御します。

SET rowcount

`integer` Transact-SQL の ROWCOUNT オプションは、カーソルについてフェッチするローの数の上限を指定の整数に設定します。カーソルを再位置付けしてフェッチしたローにも適用されます。この上限を超えたフェッチには警告が発せられます。このオプション設定は OPEN の要求が出て、カーソルがフェッチするロー数の推測値が返されるときに考慮されます。

また、SET ROWCOUNT では、検索する UPDATE 文または DELETE 文の対象となるローの数を `integer` に制限します。たとえば、この機能を使用すると、一定の間隔で COMMIT 文を実行し、ロールバックログとロックテーブルのサイズを制限することができます。アプリケーション (またはプロシージャ) は、最初の操作が影響するローに更新/削除を再発行させるループを指定する必要があります。次に簡単な例を示します。

```
BEGIN
   DECLARE @count INTEGER
   SET rowcount 20
   WHILE(1=1) BEGIN
      UPDATE GROUPO.Employees SET Surname='new_name'
      WHERE Surname <> 'old_name'
      /* Stop when no rows changed */
      SELECT @count = @@rowcount
      IF @count = 0 BREAK
      PRINT string('Updated ',
               @count,' rows; repeating...')
      COMMIT
   END
   SET rowcount 0
END
```

ROWCOUNT 設定が Interactive SQL が表示できるローの数より大きい場合、Interactive SQL は追加のフェッチを行ってカーソルを再配置することができます。そのため、実際に表示されるローの数は、要求した数より少なくなることがあります。また、切り捨ての警告のために、ローが再度フェッチされる場合、カウントは正しくない場合があります。

値を 0 にすると、オプションをリセットし、すべてのローを取得します。

SET self_recursion

self_recursion オプションをトリガ内で使用して、そのトリガに関連するテーブルの操作が他のトリガを起動できるか (On) できないか (Off) を設定します。

SET string_rtruncation

SQL 文字列データ代入時にスペース以外の文字がトランケートされるときのデフォルト動作は、SQL Anywhere と Adaptive Server Enterprise では異なります。string_rtruncation を On に設定すると、文字列の比較に関して、Transact-SQL との互換性が提供されます。

SET textsize

SELECT 文で返される TEXT 型データまたは IMAGE 型データの最大サイズをバイト単位で指定します。@@textsize グローバル変数は、現在の設定を格納します。デフォルトサイズ (32 KB) にリセットするには、次のコマンドを使用します。

SET トランザクションの独立性レベル

現在の接続のロック独立性レベルを設定します。

Adaptive Server Enterprise では、1 と 3 だけが有効なオプションです。SQL Anywhere の場合、0、1、2、3、snapshot、statement snapshot、read only statement snapshot が有効なオプションです。

SET 文は、SQL Anywhere において互換性のために prefetch オプションで使用できますが、効力はありません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

関連情報

DATEPART 関数 [日付と時刻] [310 ページ]
SET OPTION 文 [1310 ページ]

1.4.4.256 SETUSER 文

認証された別のユーザ ID の使用をユーザに許可します (同一化)。

構文

```
{ SETUSER | SET SESSION AUTHORIZATION }
```

```
[ [ WITH OPTION ] userid ]
```

パラメータ

SETUSER or SET SESSION AUTHORIZATION

SETUSER と SET SESSION AUTHORIZATION は、セマンティック上同じものです。ただし、SETUSER に指定する値は識別子 (たとえば、`SETUSER JoeS` または `SETUSER "JoeS"`) としてフォーマットする必要があります。ここで、SET SESSION AUTHORIZATION に指定する値は、文字列 (たとえば、`SET SESSION AUTHORIZATION 'JoeS'`) としてフォーマットする必要があります。

WITH OPTION clause

同一セッション中に、有効な同一化のデータベースオプション設定が、結果に影響を与える `userid` の設定と異なる場合があります。データベースオプションを変更して `userid` で有効なオプションを反映する場合は、WITH OPTION を指定します。

備考

SETUSER 文は、管理作業のために提供されており、接続プーリングには使用しないでください。SETUSER 文を実行した後、以下のいずれかのコマンドを実行すると、使用しているユーザ認証を確認できます。

- SELECT USER
- SELECT CURRENT USER

ユーザ ID のない SETUSER は、前に実行したすべての SETUSER 文を元に戻します。

成功した同一化は、手動で終了するか (ID なしで SETUSER 文を実行して)、セッションが終了するまで有効になります。

プロシージャ、トリガ、イベントハンドラ、またはバッチ内では SETUSER 文を使用できません。

SETUSER 文の使用法には、次のようなものがあります。

オブジェクトの作成

SETUSER を使用して、別のユーザが所有することになるデータベースオブジェクトを作成できます。

権限のチェック

ユーザは、別のユーザの権限と継承を使用し、クエリ、プロシージャ、ビューなどの権限や名前の決定をテストできます。

管理者に安全な環境を提供

データベース管理者はデータベースでのすべての動作を行うパーミッションを持ちます。不注意から意図しない動作を行うことを確実に回避したい場合には、SETUSER を使用して権限の少ない別のユーザ ID に切り替えます。

権限

SET USER システム権限が必要です。ただし、SETUSER 文を正常に実行できる (同一化セッションを開始する) かどうかは、同一化の対象ユーザの最小限の条件を満たしているかどうかに依存します。この条件が満たされない場合、SETUSER 文は失敗します。

標準

ANSI/ISO SQL 標準

SET SESSION AUTHORIZATION 構文は、ANSI/ISO SQL のオプションの言語機能 F321、"User authorization" の一部です。SETUSER 構文は標準の一部ではありません。WITH OPTION 構文はいずれの変形でも使用できますが、WITH OPTION は標準の一部ではありません。

例

この例の最初の文 (`SETUSER "Joe"`) で、SET USER システム権限を持つユーザが Joe を同一化し、Joe の権限を使用していくつかの操作を実行します。2 番目の文 (`SETUSER WITH OPTION "Jane"`) で、ユーザは Jane を同一化して、現在有効な Jane の権限とデータベースオプションを使用して、いくつかの操作を実行します。3 番目の文 (`SETUSER`) で、ユーザは Joe と Jane のユーザ ID、権限、データベースオプションを戻します。

```
SETUSER "Joe"
//  Some operations are run using Joes privileges ...
SETUSER WITH OPTION "Jane"
//  Some operations are run using Jane's privileges, and the
//  database options in effect are changed to the current
//  database options for Jane
SETUSER
```

関連情報

文字列 [10 ページ]
EXECUTE IMMEDIATE 文 [SP] [1090 ページ]
GRANT 文 [1131 ページ]
REVOKE 文 [1278 ページ]
SET OPTION 文 [1310 ページ]
識別子 [6 ページ]

1.4.4.257 SIGNAL 文 [SP]

例外条件を通知します。

構文

```
SIGNAL exception-name
```

備考

SIGNAL を使うと、例外を起こすことができます。

exception-name は、現在の複合文の始めにある DECLARE 文を使用して宣言された例外の名前を指定する場合に使用します。この例外は、システム定義の SQLSTATE またはユーザ定義の SQLSTATE と合致していなければなりません。ユーザ定義 SQLSTATE の値は 99000 ～ 99999 の範囲になければなりません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

SIGNAL 文は、オプションの ANSI/ISO SQL 言語機能 P002、"Computational completeness" の一部です。

Transact-SQL

SIGNAL 文は、Transact-SQL 複合文およびプロシージャでは使用できません。

例

次の複合文は、ユーザ定義の例外を宣言し、その信号を送ります。この例を Interactive SQL から実行すると、結果領域の履歴タブに "例外が通知されました" というメッセージが表示されます。

```
BEGIN
   DECLARE myexception EXCEPTION
```

```
   FOR SQLSTATE '99001';
   SIGNAL myexception;
   EXCEPTION
      WHEN myexception THEN
         MESSAGE 'My exception signaled'
         TO CLIENT;
END
```

関連情報

RESIGNAL 文 [SP] [1268 ページ]
BEGIN 文 [745 ページ]
RAISERROR 文 [1246 ページ]

1.4.4.258 START DATABASE 文

データベースを現在のデータベースサーバで起動します。

構文

```
START DATABASE database-file [ start-options ... ]
```

```
start-options :
[ AS database-name ]
[ WITH TRUNCATE AT CHECKPOINT ]
[ FOR READ ONLY ]
[ AUTOSTOP { ON | OFF } ]
[ KEY key ]
[ WITH SERVER NAME alternative-database-server-name ]
[ DIRECTORY dbspace-directory ]
[ CHECKSUM { ON | OFF } ]
[ DISKSANDBOX { ON | OFF | DEFAULT } ]
[ MIRROR ON ]
```

パラメータ

database-file

database-file パラメータは、文字列です。database-file に相対パスを指定する場合、そのパスはデータベースサーバの開始ディレクトリを基準にします。

start-options clause

start-options 句は任意の順序で表示できます。

AS clause

database-name を指定しない場合、デフォルト名がデータベースに割り当てられます。このデフォルト名は、データベースファイルのルートです。たとえば、ファイル C:¥Database Files¥*demo.db* 内のデータベースにはデフォルト名の demo が付きます。database-name パラメータは識別子です。

WITH TRUNCATE AT CHECKPOINT clause

チェックポイントのログの切り捨てを有効にしてデータベースを起動します。

FOR READ ONLY clause

読み込み専用モードでデータベースを起動します。リカバリが必要なデータベースで使用した場合は文が失敗し、エラーが返されます。

AUTOSTOP clause

AUTOSTOP 句のデフォルト設定は ON です。AUTOSTOP を ON に設定すると、最後の接続が削除されるときに、データベースが自動的にアンロードされます。AUTOSTOP を OFF に設定すると、データベースはアンロードされません。

Interactive SQL では、ON と OFF の代わりに YES と NO も使用できます。

KEY clause

データベースが強力に暗号化されている場合は、この句を使用して KEY 値 (パスワード) を入力します。

WITH SERVER NAME clause

この句は、データベースサーバに接続するときにそのサーバの代替名を指定するときに使用します。

この句は、ミラーリングされたデータベースと一緒に使用しないでください。

DIRECTORY clause

この句を使用して、起動しようとしているデータベースの DB 領域が配置されているディレクトリを指定します。たとえば、データベースサーバが DB 領域と同じディレクトリで起動されている場合に DIRECTORY '.' 句があると、データベースサーバに対して現在のディレクトリにあるすべての DB 領域を検出するように指示することになります。

CHECKSUM clause

この句は、グローバルチェックサムを有効にして作成されていないデータベースに新しく書き込まれるページに対して、書き込みチェックサムを有効にするときに使用します。この句を指定すると、-wc データベースオプションと同じ動作になります。

グローバルチェックサムを有効にしてデータベースを作成することと CHECKSUM 句の違いは、CHECKSUM ON を指定した場合、ディスクに書き込まれるときにのみ、データベースページにチェックサムが追加されることです。ディスクから読み込まれるページは、ページが書き込まれる前にチェックサム値が計算された場合にのみ検証されます。データベースでグローバルチェックサムが有効にされている場合、書き込み時にすべてのページに対してチェックサムが計算され、読み込み時にすべてのページに対してチェックサムが検証されます。

データベースがリムーバブルストレージデバイス (ネットワーク共有や USB デバイスなど) で実行されていることがデータベースサーバで検出されると、データベースサーバでは、すべてのデータベースページに対して書き込みチェックサムが自動的に有効になります。

デフォルトでは、バージョン 10 または 11 の SQL Anywhere で作成されたデータベースではグローバルチェックサムは有効になっていません。バージョン 12 以降のデータベースサーバで、SQL Anywhere 10 または 11 で作成されたデータベースを起動すると、ページがディスクに書き込まれるときに、デフォルトでデータベースサーバは、ページに対する書き込みチェックサムを作成します (CHECKSUM ON)。バージョン 12 以降のデータベースは、デフォルトでグローバルチェックサムが有効にされており、デフォルトですべてのデータベースページにチェックサムがあるため、データベースサーバはこれらのデータベースに対して CHECKSUM OFF をデフォルト設定します。デフォルトのチェ

ックサム設定を使用しない場合は、-wc オプションまたは START DATABASE 文を使用してデータベースサーバのチェックサム動作を変更できます。

次の文を実行することによって、データベースがグローバルチェックサムを有効にして作成されたかどうかをチェックできます。

```
SELECT DB_PROPERTY ( 'Checksum' );
```

次の文を実行すると、書き込みチェックサムが有効にされているかどうかをチェックできます。

```
SELECT DB_PROPERTY ( 'WriteChecksum' );
```

DISKSANDBOX clause

メインデータベースファイルがあるディレクトリ以外の場所でデータベースファイルの読み込み/書き込み操作を制限するには、DISKSANDBOX を ON に設定します。すべてのディレクトリへのアクセスを許可するには、DISKSANDBOX を OFF に設定します。DISKSANDBOX を DEFAULT に設定すると、-sbx データベースサーバオプションによって指定されたディスクサンドボックスが使用されます。

> **注記**
>
> -sbx データベースサーバオプションを使用してデータベースサーバを起動する場合は、manage_disk_sandbox セキュリティ機能のセキュリティ機能キーを指定して、DISKSANDBOX を OFF に設定してデータベースを起動する必要があります。

MIRROR ON clause

すでに実行されており、ミラーリングされている可能性があるデータベースをホスティングしているデータベースサーバに、さらにミラーリングされたデータベースを追加するには、MIRROR ON 句を使用します。この句を使用する場合は、AUTOSTOP OFF 句を指定する必要があります。

備考

指定したデータベースを現在のデータベースサーバで起動します。

START DATABASE 文は、指定されたデータベースに現在のアプリケーションを接続しません。START DATABASE 文を使用する場合も明示的な接続が必要です。

データベースに接続していない状態で、START DATABASE 文を使用する場合、まずユーティリティデータベースなどのデータベースに接続します。

ユーティリティデータベースへの接続に使用できるのは SQL Anywhere utility_db という名前のデータベースのみです。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

権限

ネットワークサーバ上のデータベースを起動するのに必要な権限は、データベースサーバの -gd オプションによって決まります。デフォルトでは、ネットワークサーバ上でデータベースを起動するには SERVER OPERATOR システム権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、現在のサーバ上で架空のデータベースファイル `C:¥temp¥sample_2.db` を起動します。

```
START DATABASE 'c:¥¥temp¥¥sample_2.db';
```

この例では、同じデータベースを起動しますが、sam2 となります。

```
START DATABASE 'c:¥¥temp¥¥sample_2.db'
AS sam2;
```

関連情報

STOP DATABASE 文 [1347 ページ]
CREATE MIRROR SERVER 文 [858 ページ]
CONNECT 文 [ESQL] [Interactive SQL] [774 ページ]
VALIDATE 文 [1400 ページ]

1.4.4.259 START SERVER 文 [Interactive SQL]

データベースサーバを起動します。

構文

```
START SERVER AS database-server-name [ STARTLINE command-string ]
```

備考

START SERVER 文は、データベースサーバを起動します。データベースサーバに対してオプションのセットを指定する場合、STARTLINE キーワードをコマンド文字列と一緒に使います。

START ENGINE は互換性のために受け入れられますが、推奨されません。

この SQL 文は SAP HANA データベースではサポートされていません。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、sample という名前のデータベースサーバを、1 つもデータベースを起動しない状態で起動します。

```
START SERVER AS sample;
```

この例では、STARTLINE 句の使い方を示します。

```
START SERVER AS eng1 STARTLINE 'dbsrv17 -c 8M';
```

関連情報

STOP SERVER 文 [1352 ページ]

1.4.4.260 START EXTERNAL ENVIRONMENT 文

外部環境を開始します。

構文

```
START EXTERNAL ENVIRONMENT environment-name
```

```
environment-name :
C_ESQL32
| C_ESQL64
| C_ODBC32
| C_ODBC64
| CLR
| JAVA
| JS
| PERL
| PHP
```

パラメータ

environment-name

開始する外部環境の名前。

備考

なし。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

Perl 外部環境を開始します。

```
START EXTERNAL ENVIRONMENT PERL;
```

関連情報

ALTER EXTERNAL ENVIRONMENT 文 [647 ページ]
STOP EXTERNAL ENVIRONMENT 文 [1348 ページ]
INSTALL EXTERNAL OBJECT 文 [1173 ページ]
REMOVE EXTERNAL OBJECT 文 [1263 ページ]
SYSEXTERNENV システムビュー [1802 ページ]
SYSEXTERNENVOBJECT システムビュー [1804 ページ]

1.4.4.261 START JAVA 文

Java VM を起動します。

構文

```
START JAVA
```

備考

START JAVA 文は、Java VM を起動します。主な使い方として、Java VM を適宜ロードし、ユーザが Java の機能を使い始めるときに、Java VM ロード中の一時停止が発生しないようにします。

Java VM の場所を特定できるようにデータベースサーバを設定します。データベースごとに異なる Java VM を指定できるため、ALTER EXTERNAL ENVIRONMENT 文を使用して Java VM のロケーション (パス) を指定できます。

Java VM をインストールする必要があります。

権限

なし

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、Java VM を起動します。

```
START JAVA;
```

関連情報

STOP JAVA 文 [1350 ページ]

1.4.4.262 START LOGGING 文 [Interactive SQL]

実行された SQL 文およびメッセージのログファイルへのロギングを開始します。

構文

```
START LOGGING filename
```

備考

START LOGGING 文は、それ以降に実行されるすべての SQL 文およびメッセージの指定されたログファイルへのコピーを開始します。このファイルが存在しない場合は、Interactive SQL によって作成されます。このファイルが存在する場合は、

Interactive SQL によって追加されます。ロギングは、STOP LOGGING 文で明示的に停止するか、現在の Interactive SQL セッションを終了するまで継続します。

SQL > ロギングの開始 と *SQL* > ロギングの停止 をクリックして、ロギングを開始したり停止したりすることもできます。

ロギングと実行時間レポートの両方が有効である場合は、実行時間がログファイルに記録されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、ファイル `c:¥temp¥sql.log` へのロギングを開始します。

```
START LOGGING 'c:¥¥temp¥¥sql.log';
```

関連情報

STOP LOGGING 文 [Interactive SQL] [1351 ページ]

1.4.4.263 START SUBSCRIPTION 文 [SQL Remote]

パブリケーションに対するユーザのサブスクリプションを開始します。

構文

START SUBSCRIPTION

```
TO publication-name [ ( subscription-value ) ]
FOR subscriber-id, ...
```

パラメータ

publication-name

このパブリケーション名でユーザのサブスクリプションが作成されます。パブリケーションの所有者を含めることもできます。

subscription-value

パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。

subscriber-id

パブリケーションに対するサブスクライバのユーザ ID。このユーザは、パブリケーションへのサブスクリプションを持っている必要があります。

備考

SQL Remote サブスクリプションは、パブリケーションの更新が統合データベースからリモートデータベースへ送信されると、開始されます。

START SUBSCRIPTION 文は、サブスクリプションを管理する一連の文の 1 つです。CREATE SUBSCRIPTION 文は、サブスクライバが受信するデータを定義します。SYNCHRONIZE SUBSCRIPTION 文は、統合データベースとリモートデータベース間の一貫性を確保するためのものです。メッセージのサブスクライバへの送信を開始するには、START SUBSCRIPTION 文が必ず必要です。

各サブスクリプションの最後のデータは、サブスクリプションを開始する前のデータと一致していなければなりません。データベース抽出ユーティリティを使用してサブスクリプションの作成、同期、起動を管理します。データベース抽出ユーティリティを使用すれば明示的に START SUBSCRIPTION 文を実行する必要はなくなります。また、Message Agent も、サブスクリプションが同期されるとサブスクリプションを起動します。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、パブリケーション pub_contact に対するユーザ Sam_Singer のサブスクリプションを起動します。

```
START SUBSCRIPTION TO pub_contact
FOR Sam_Singer;
```

関連情報

CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]
REMOTE RESET 文 [SQL Remote] [1262 ページ]
SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote] [1362 ページ]
STOP SUBSCRIPTION 文 [SQL Remote] [1353 ページ]

1.4.4.264 START SYNCHRONIZATION DELETE 文 [Mobile Link]

Mobile Link 同期で行われた削除のロギングを再起動します。

構文

START SYNCHRONIZATION DELETE

備考

通常、SQL Anywhere と Ultra Light は、同期の一部であるテーブルやカラムに対して行われたすべての変更のログを自動的に取り、次の同期中にこれらの変更を統合データベースにアップロードします。STOP SYNCHRONIZATION DELETE 文を使用すると、削除処理の自動ロギングを一時的に中断できます。START SYNCHRONIZATION DELETE 文を使用すると、自動ロギングを再起動できます。

STOP SYNCHRONIZATION DELETE 文を実行すると、その接続に対してそれ以降に実施された削除操作は同期されません。この効果は、START SYNCHRONIZATION DELETE 文が実行されるまで継続します。STOP SYNCHRONIZATION DELETE を繰り返してもそれ以上効果はありません。

STOP SYNCHRONIZATION DELETE 文の実行回数にかかわらず、START SYNCHRONIZATION DELETE 文を 1 回実行すれば、ロギングが再起動します。

アプリケーションでデータを同期しない場合は、START SYNCHRONIZATION DELETE を使用しないでください。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の一連の SQL 文は、START SYNCHRONIZATION DELETE と STOP SYNCHRONIZATION DELETE の使用方法を示します。

```
-- Prevent deletes from being sent
-- to the consolidated database
STOP SYNCHRONIZATION DELETE;
-- Remove all records older than 1 month
-- from the remote database,
-- NOT the consolidated database
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT TIMESTAMP, -1 )
-- Re-enable all deletes to be sent
-- to the consolidated database
-- DO NOT FORGET to start this
START SYNCHRONIZATION DELETE;
-- Commit the entire operation,
-- otherwise rollback everything
-- including the stopping of the deletes
commit;
```

関連情報

STOP SYNCHRONIZATION DELETE 文 [Mobile Link] [1355 ページ]

1.4.4.265 START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]

Mobile Link 同期スキーマの変更を開始します。

構文

```
START SYNCHRONIZATION SCHEMA CHANGE
FOR TABLES table-list
set-script-version
 | set-script-version-on-subscription, ...
```

```
set-script-version :
SET SCRIPT VERSION = script-version
```

```
set-script-version-on-subscription :
SET SCRIPT VERSION = script-version ON SUBSCRIPTION subscription_name
```

```
script-version : string | NULL
```

```
subscription-name : identifier
```

パラメータ

FOR TABLES clause

この句は、スキーマの変更によって影響を受けるテーブルを指定します。

SET SCRIPT VERSION clause

FOR TABLES 句で指定されたテーブルを含むすべてのサブスクリプションの新しいスクリプトバージョンを指定します。新しいスクリプトバージョンは、既存のスクリプトバージョンと同じでもかまいません。

SET SCRIPT VERSION...ON SUBSCRIPTION clause

指定されたサブスクリプションに新しいスクリプトバージョンを指定します。この句を使用する場合、FOR TABLES 句で指定されたテーブルを含むサブスクリプションごとに、カンマで区切られたこの句を繰り返す必要があります。新しいスクリプトバージョンは、既存のスクリプトバージョンと同じでもかまいません。

備考

スキーマの変更を適用するすべてのテーブルを `table-list` にリストしてください。テーブルを複数回リストすることはできません。`table-list` に指定されているテーブルに既存のロックがある場合、エラーメッセージが表示されます。

データベースで一度に実行できる同期スキーマの変更は 1 つのみです。別のスキーマの変更が処理中の場合、START SYNCHRONIZATION SCHEMA CHANGE 文は失敗します。

データベースサーバは、table-list に指定されているすべてのテーブルのロックを取得します。データベースサーバは、ロックを取得するとき、ブロックオプションの設定を無視します。ロックを取得できない場合、それまでに取得したすべてのロックが解放され、エラーメッセージが表示されます。

同期スキーマの変更中は、

- データ操作文を実行できません。
- 別の START SYNCHRONIZATION SCHEMA CHANGE 文を実行できません。
- パブリケーションを変更して、table-list にあるテーブルのカラムのサブセット方法を変更できます。
- パブリケーションを変更して、table-list にあるテーブルを削除できます。
- table-list にリストされている任意のテーブルを変更できます。

START SYNCHRONIZATION SCHEMA CHANGE 文の実行の前後に、暗黙的なコミットが実行されます。同期スキーマの変更は、STOP SYNCHRONIZATION SCHEMA CHANGE 文によって終了します。STOP SYNCHRONIZATION SCHEMA CHANGE 文が実行されると、テーブルロックがすべて解放されます。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の一連の SQL 文は、START SYNCHRONIZATION SCHEMA CHANGE と STOP SYNCHRONIZATION SCHEMA CHANGE の使用方法を示します。

```
START SYNCHRONIZATION SCHEMA CHANGE
  FOR TABLES GROUPO.SalesOrders, GROUPO.Products
  SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub1,
  SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub2;
ALTER TABLE GROUPO.SalesOrders ADD SUBTOTAL NUMERIC (10,2);
ALTER TABLE GROUPO.Products ALTER QUANTITY BIGINT;
STOP SYNCHRONIZATION SCHEMA CHANGE;
```

関連情報

STOP SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link] [1357 ページ]

1.4.4.266 STOP DATABASE 文

現在のデータベースサーバ上のデータベースを停止します。

構文

```
STOP DATABASE database-name
[ ON database-server-name ]
[ UNCONDITIONALLY ]
```

パラメータ

STOP DATABASE clause

database-name は、現在のサーバで動作している (現在のデータベース以外の) データベースの名前です。

ON clause

この句は Interactive SQL でのみサポートされます。Interactive SQL で database-server-name を指定しないと、すべての実行しているサーバについて、指定したデータベースを検索します。

Interactive SQL でこの文を使用しない場合、データベースは現在のデータベースサーバで起動された場合のみ停止されます。

UNCONDITIONALLY clause

データベースとの接続がある場合でも、そのデータベースを停止します。デフォルトでは、接続があるとデータベースは停止しません。

備考

STOP DATABASE 文は、現在のデータベースサーバ上の指定したデータベースを停止します。

現在接続しているデータベースに対して STOP DATABASE を使用することはできません。

データベースを停止するかどうかを判断するときに、データベースサーバで特定の接続を無視したい場合は、connection_type オプションを使用できます。

権限

ネットワークサーバ上のデータベースを停止するのに必要な権限は、データベースサーバの -gd オプションによって決まります。ネットワークサーバ上のデータベースを停止するためのデフォルトのシステム権限は SERVER OPERATOR です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

現在のサーバ上のデータベース `sample` を停止します。

```
STOP DATABASE sample;
```

関連情報

START DATABASE 文 [1333 ページ]
DISCONNECT 文 [ESQL] [Interactive SQL] [1028 ページ]

1.4.4.267 STOP EXTERNAL ENVIRONMENT 文

外部環境を停止します。

構文

```
STOP EXTERNAL ENVIRONMENT environment-name
```

```
environment-name :
C_ESQL32
| C_ESQL64
| C_ODBC32
| C_ODBC64
| CLR
```

```
| JAVA
| JS
| PERL
| PHP
```

パラメータ

environment-name

停止する外部環境の名前。

備考

なし

権限

なし

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

この例では、Perl 外部環境を停止します。

```
STOP EXTERNAL ENVIRONMENT PERL;
```

関連情報

ALTER EXTERNAL ENVIRONMENT 文 [647 ページ]
START EXTERNAL ENVIRONMENT 文 [1338 ページ]
INSTALL EXTERNAL OBJECT 文 [1173 ページ]
REMOVE EXTERNAL OBJECT 文 [1263 ページ]
SYSEXTERNENV システムビュー [1802 ページ]
SYSEXTERNENVOBJECT システムビュー [1804 ページ]

1.4.4.268 STOP JAVA 文

Java VM を停止します。

構文

STOP JAVA

備考

STOP JAVA 文は、現在の接続の ClassLoader をアンロードします。現在の接続が、Java VM を使用する最後の接続である場合、STOP JAVA 文は Java VM の終了も行います。STOP JAVA 文の主な目的は、システムリソースを経済的に使用することです。

権限

なし

関連する動作

なし

標準

- ANSI/ISO SQL 標準

標準になし。

例

この例では、Java VM を停止します。

```
STOP JAVA;
```

関連情報

START JAVA 文 [1339 ページ]

1.4.4.269 STOP LOGGING 文 [Interactive SQL]

現在のセッションで実行される SQL 文およびメッセージのロギングを停止します。

構文

STOP LOGGING

備考

STOP LOGGING 文は、実行した各 SQL 文およびメッセージのログファイルへのロギングを停止します。ロギングを開始するには、START LOGGING 文を使用します。

SQL > **ロギングの停止** をクリックしてロギングを停止することもできます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、現在のロギングセッションを停止します。

```
STOP LOGGING;
```

関連情報

START LOGGING 文 [Interactive SQL] [1340 ページ]

1.4.4.270 STOP SERVER 文

データベースサーバを停止します。

構文

```
STOP SERVER [ database-server-name ] [ UNCONDITIONALLY ]
```

パラメータ

UNCONDITIONALLY clause

停止するデータベースサーバに接続しているユーザが他にいない場合、UNCONDITIONALLY を使用する必要はありません。他にも接続しているユーザがいる場合は、UNCONDITIONALLY キーワードを指定した場合のみデータベースサーバが停止します。

備考

`database-server-name` は Interactive SQL でのみ使用できます。この文を Interactive SQL から実行しなかった場合、現在のデータベースサーバが停止されます。

デフォルトでは、データベースサーバは他の接続がある場合には停止しません。UNCONDITIONALLY 句を使用すると、データベースサーバはデータベースサーバへの他の接続がある場合でも停止します。STOP SERVER 文をクライアント接続で実行した場合、サーバが正常に停止すると、通信エラーが発生します。

データベースを停止するかどうかを判断するときに、データベースサーバで特定の接続を無視したい場合は、connection_type オプションを使用できます。

STOP SERVER 文は、ストアドプロシージャ、トリガ、イベント、またはバッチでは使用できません。

互換性を保つために STOP ENGINE はまだ使用できますが、廃止される予定です。

権限

ネットワークサーバ (dbsrv17) を停止するために必要な権限は、データベースサーバコマンドラインの -gk 設定によって指定されます。デフォルトでは、ネットワークサーバを停止するには SERVER OPERATOR 権限が必要です。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

現在のデータベースサーバを他に接続がない場合のみ停止します。

```
STOP SERVER;
```

関連情報

START SERVER 文 [Interactive SQL] [1336 ページ]

1.4.4.271 STOP SUBSCRIPTION 文 [SQL Remote]

パブリケーションに対するユーザのサブスクリプションを停止します。

構文

STOP SUBSCRIPTION

```
TO publication-name [ ( subscription-value ) ]
FOR subscriber-id, ...
```

パラメータ

publication-name

このパブリケーション名でユーザのサブスクリプションが作成されます。パブリケーションの所有者を含めることもできます。

subscription-value

パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。

subscriber-id

パブリケーションに対するサブスクライバのユーザ ID。このユーザは、パブリケーションへのサブスクリプションを持っている必要があります。

備考

SQL Remote サブスクリプションは、パブリケーションの更新が統合データベースからリモートデータベースへ送信されると、開始されます。

サブスクライバに送るメッセージを停止するには、STOP SUBSCRIPTION 文が必要です。サブスクライバに送るメッセージを再開するには、START SUBSCRIPTION 文が必要です。サブスクリプションを再開する場合は足りないメッセージがないように、サブスクリプションが正しく同期していることを確認してください。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、パブリケーション pub_contact に対するユーザ SamS のサブスクリプションを停止します。

```
STOP SUBSCRIPTION TO pub_contact
FOR Sam_Singer;
```

関連情報

DROP SUBSCRIPTION 文 [SQL Remote] [1069 ページ]
START SUBSCRIPTION 文 [SQL Remote] [1341 ページ]
SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote] [1362 ページ]

1.4.4.272 STOP SYNCHRONIZATION DELETE 文 [Mobile Link]

Mobile Link 同期で行われた削除のロギングを一時的に停止します。

構文

STOP SYNCHRONIZATION DELETE

備考

通常、SQL Anywhere と Ultra Light リモートデータベースは、同期中のテーブルやカラムに対して行われたすべての変更のログを自動的に取り、次の同期中にこれらの変更を統合データベースにアップロードします。この文を使用して、SQL Anywhere または Ultra Light リモートデータベースに対する削除操作のロギングを一時的に中断できます。

接続が STOP SYNCHRONIZATION DELETE を実行するときと START SYNCHRONIZATION DELETE を実行するときの間に、その接続で実行された削除操作は同期されません。

STOP SYNCHRONIZATION DELETE 文の実行回数にかかわらず、START SYNCHRONIZATION DELETE 文を 1 回実行すれば、ロギングが再起動します。

この文は、リモートデータベースに対して訂正を行うには便利ですが、Mobile Link 同期を事実上無効にしてしまうので、使用の際には注意してください。

アプリケーションでデータを同期しない場合は、STOP SYNCHRONIZATION DELETE を使用しないでください。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の一連の SQL 文は、START SYNCHRONIZATION DELETE と STOP SYNCHRONIZATION DELETE の使用方法を示します。

```
-- Prevent deletes from being sent
-- to the consolidated database
STOP SYNCHRONIZATION DELETE;
-- Remove all records older than 1 month
-- from the remote database,
-- NOT the consolidated database
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT TIMESTAMP, -1 )
-- Re-enable all deletes to be sent
-- to the consolidated database
-- DO NOT FORGET to start this
START SYNCHRONIZATION DELETE;
-- Commit the entire operation,
-- otherwise rollback everything
-- including the stopping of the deletes
commit;
```

関連情報

START SYNCHRONIZATION DELETE 文 [Mobile Link] [1343 ページ]

1.4.4.273 STOP SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]

Mobile Link の同期スキーマ変更を停止します。

構文

STOP SYNCHRONIZATION SCHEMA CHANGE

備考

STOP SYNCHRONIZATION SCHEMA CHANGE 文は、START SYNCHRONIZATION SCHEMA CHANGE 文で開始したスキーマ変更を停止します。START SYNCHRONIZATION SCHEMA CHANGE 文によって取得されるすべてのロックがリリースされます。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の一連の SQL 文は、START SYNCHRONIZATION SCHEMA CHANGE と STOP SYNCHRONIZATION SCHEMA CHANGE の使用方法を示します。

```
START SYNCHRONIZATION SCHEMA CHANGE
  FOR TABLES GROUPO.SalesOrders, GROUPO.Products
  SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub1,
  SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub2;
ALTER TABLE GROUPO.SalesOrders ADD SUBTOTAL NUMERIC (10,2);
ALTER TABLE GROUPO.Products ALTER QUANTITY BIGINT;
```

```
STOP SYNCHRONIZATION SCHEMA CHANGE;
```

関連情報

START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link] [1345 ページ]
SYSARTICLE システムビュー [1793 ページ]

1.4.4.274 SYNCHRONIZE 文 [Mobile Link]

SQL Anywhere データベースを Mobile Link サーバと同期させます。文自体に同期オプションを指定できます。

構文

```
SYNCHRONIZE {
PROFILE sync-profile-name [ MERGE sync-option [ ;... ] ]
| USING sync-option [ ;... ]
| START
| STOP
}
```

```
[ PORT port-number ]
[ VERBOSITY { LOW | NORMAL | HIGH } ]
[ TIMEOUT timeout ]
[ USER user-name IDENTIFIED BY password ]
```

```
sync-option : string
```

パラメータ

sync-profile-name

この同期に使用する同期プロファイルの名前。
MERGE clause

この句は、同期プロファイルオプションを追加または上書きするために使用します。
USING clause

この句は、同期プロファイルを使用しないで同期プロファイルオプションを指定するために使用します。
sync-option

1 つ以上の同期プロファイルオプションの値ペアをセミコロンで区切った文字列です。たとえば、`'option1=value1;option2=value2'` のように記述します。
PORT clause

この句は、データベースサーバが dbmlsync ユーティリティと通信するために使用するポート番号を指定するために使用します。デフォルトは 4433 です。

VERBOSITY clause

この句では、同期中に synchronize_results および synchronize_parameters 共有グローバルテンポラリテーブルに追加される情報量を指定します。

次に、各 VERBOSITY オプションで返されるクライアント API イベントのリストを示します。

オプション	戻り値
LOW	• DBSC_EVENTTYPE_SYNC_START • DBSC_EVENTTYPE_SYNC_DONE • DBSC_EVENTTYPE_ERROR_MSG • DBSC_EVENTTYPE_WARNING_MSG
NORMAL (デフォルト)	• DBSC_EVENTTYPE_SYNC_START • DBSC_EVENTTYPE_SYNC_DONE • DBSC_EVENTTYPE_ERROR_MSG • DBSC_EVENTTYPE_WARNING_MSG • DBSC_EVENTTYPE_INFO_MSG
HIGH	• DBSC_EVENTTYPE_SYNC_START • DBSC_EVENTTYPE_SYNC_DONE • DBSC_EVENTTYPE_ERROR_MSG • DBSC_EVENTTYPE_WARNING_MSG • DBSC_EVENTTYPE_INFO_MSG • DBSC_EVENTTYPE_PROGRESS_INDEX • DBSC_EVENTTYPE_PROGRESS_TEXT • DBSC_EVENTTYPE_TITLE

SYNCHRONIZE 文の VERBOSITY 句と、同期プロファイルに指定できる VERBOSITY オプションとを混同しないように注意してください。SYNCHRONIZE 文の VERBOSITY 句は、synchronize_results および synchronize_parameters テーブルに記録されるイベントの型を制御します。同期プロファイルの VERBOSITY オプションは、同期中に生成される DBSC_EVENTTYPE_INFO_MSG イベントの数を制御します。

```
SYNCHRONIZE PROFILE SalesData VERBOSITY NORMAL;
```

```
SYNCHRONIZE PROFILE SalesData MERGE 'Verbosity=BASIC,ROW_DATA' VERBOSITY NORMAL;
```

TIMEOUT clause

この句では、同期のキャンセルを試行するまでにデータベースサーバが同期の完了を待機する時間を秒数で指定します。デフォルトは 240 秒です。

USER/IDENTIFIED BY clause

この句は、dbmlsync ユーティリティでリモートデータベースに接続して同期を実行するために使用する、データベースユーザ ID とパスワードを指定するときに使用します。指定するユーザ ID には、SYS_RUN_REPLICATION_ROLE システムロールが必要です。デフォルトでは、同期には SYNCHRONIZE 文を実行したデータベース接続のユーザ ID が使用されます。

START clause

dbmlsync ユーティリティのサーバモードでの実行を開始し、実行したままにします。同期は実行されません。短期間に複数の同期を実行している場合、この句を使用して、明示的に dbmlsync サーバを起動して同期を実行し、STOP 句を使用して明示的に dbmlsync サーバを停止することにより、パフォーマンスを向上させることができます。

STOP clause

以前に START 句を使用して起動した dbmlsync サーバを停止します。同期は実行されません。

備考

この文は、Dbmlsync C++ API を含む SQL Anywhere 用 Mobile Link クライアントがインストールされている場合にのみ使用できます。

データベースサーバが実行される可能性があるすべてのプラットフォームで、SQL Anywhere 用 Mobile Link クライアントを利用できません。

同期が完了したら、synchronize_results と synchronize_parameters 共有グローバルテンポラリテーブルに格納されている同期の結果を表示できます。synchronize_results と synchronize_parameters テーブルには、データベースが起動されて以降、SYNCHRONIZE 文で実行されたすべての同期の結果が格納されます。synchronize_results と synchronize_parameters テーブルは、データベースサーバがシャットダウンするたびにトランケートされます。

synchronize_results テーブルには次のカラムが含まれます。

カラム名	データ型	説明
row_id	UNSIGNED BIGINT	ローがテーブルに挿入された順序を判断するために使用するテーブルのプライマリキー。
conn_id	UNSIGNED INT	このイベントを生成した SYNCHRONIZE 文を実行した接続の接続 ID 番号。
result_time	TIMESTAMP	イベントが synchronize_results テーブルに追加された時刻。
result_type	CHAR(128)	イベントのタイプ。

synchronize_results テーブルに表示される各イベントには、イベントの追加情報を含む関連パラメータが存在することがあります。このパラメータは、次のカラムを含む synchronize_parameters テーブルに格納されます。

カラム名	データ型	説明
row_id	UNSIGNED BIGINT	synchronize_results テーブルの row_id カラムに対する外部キー。各パラメータをそれぞれが属していたイベントに戻す場合は、この値を使用します。
parm_id	UNSIGNED INT	パラメータの ID 番号を含みます。イベントに複数のパラメータが含まれる場合は、この値を使用して目的のパラメータを検索します。
parm_message	LONG VARCHAR	パラメータに関連付けられている値。

過去または現在の同期の情報を表示するには、synchronize_results および synchronize_parameters テーブルに対して直接クエリを実行する代わりに、sp_get_last_synchronize_result システムプロシージャを使用できます。

または、次の文を使用して、データベースサーバの起動後に行われたすべての同期の結果を表示できます。

```
SELECT *
     FROM synchronize_results sr
     KEY JOIN synchronize_parameters sp
     ORDER BY sr.row_id , sp.parm_id
```

synchronize_results と synchronize_parameters テーブルを使用すると、現在の接続とは別の接続に対する同期の進行状況をモニタできます。別の接続に対する同期の進行状況をモニタするには、次の手順に従います。

- SELECT CONNECTION_PROPERTY 文を実行して、現在の接続の接続 ID を確認します。
- SYNCHRONIZE 文を実行して同期を開始します。
- 別の接続で、use the sp_get_last_synchronize_results system プロシージャを使用して、上で確認した接続 ID を使用した結果を取得します。

特定の接続で完了した同期または進行中の同期の結果を表示するには、sp_get_last_synchronize_results システムプロシージャを使用します。

SYNCHRONIZE 文は、Ultra Light の SYNCHRONIZE 文に似ています。ただし、SQL Anywhere の SYNCHRONIZE 文は、dbmlsync ユーティリティをサーバモードで起動して同期を実行します。Ultra Light の SYNCHRONIZE 文は、Ultra Light ランタイムを使用します。

SQL Anywhere の複数バージョンが同一システムにインストールされている場合や、dbmlsync 実行プログラムがデータベースサーバと同じディレクトリに配置されていないために PATH 環境変数が見つからない場合は、ALTER EXTERNAL ENVIRONMENT コマンドを使用して dbmlsync 実行プログラムの場所を指定してください。

データベースサーバは Dbmlsync API クライアントとして機能し、TCP/IP を使用して dbmlsync サーバと通信します。デフォルトでは、通信はポート 4433 で行われます。別のポートを指定するには、PORT 句を使用してください。

SYNCHRONIZE PROFILE 文と SYNCHRONIZE USING 文を使用して、同期を実行します。SYNCHRONIZE START と SYNCHRONIZE STOP 文を使用して、dbmlsync サーバを起動または停止します。SYNCHRONIZE PROFILE 文または SYNCHRONIZE USING 文を実行すると、データベースサーバはすでに実行中の dbmlsync サーバへの接続を試みます。すでに実行中の dbmlsync サーバが見つからない場合は、dbmlsync サーバが起動します。同期が完了すると、データベースサーバは起動した dbmlsync サーバをシャットダウンします。すでに実行中の dbmlsync サーバに文が接続した場合、dbmlsync サーバはシャットダウンされません。複数の同期を実行しており、同期ごとに dbmlsync サーバを起動または停止しないようにする場合は、SYNCHRONIZE START 文に続けて、複数の SYNCHRONIZE PROFILE 文または SYNCHRONIZE USING 文を実行し、最後に SYNCHRONIZE STOP 文を実行します。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用する場合は、文字列リテラルとしてパスワードを指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE REPLICATION システム権限または SYS_RUN_REPLICATION_ROLE システムロールを持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Test1 という同期プロファイルで同期するための構文を示します。

```
SYNCHRONIZE PROFILE Test1;
```

関連情報

CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link] [946 ページ]
sp_get_last_synchronize_result システムプロシージャ [1682 ページ]
ALTER EXTERNAL ENVIRONMENT 文 [647 ページ]

1.4.4.275 SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote]

パブリケーションに対するユーザのサブスクリプションを同期します。

構文

```
SYNCHRONIZE SUBSCRIPTION
TO publication-name [ ( subscription-value ) ]
FOR remote-user, ...
```

パラメータ

publication-name

このパブリケーション名でユーザのサブスクリプションが作成されます。パブリケーションの所有者を含めることもできます。

subscription-value

パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。

remote-user

パブリケーションに対するサブスクライバのユーザ ID。このユーザは、パブリケーションへのサブスクリプションを持っている必要があります。

備考

SQL Remote サブスクリプションは、リモートデータベースのデータが統合データベースのデータと一致し、統合データベースからリモートデータベースにパブリケーションの更新を送信しても競合もエラーも発生しない状態の場合、同期されています。

サブスクリプションを同期するには、統合データベース側のパブリケーションデータのコピーをリモートデータベースに送信します。SYNCHRONIZE SUBSCRIPTION 文は、これをメッセージシステムで自動的に実行します。できるかぎり、データベース抽出ユーティリティ (dbxtract) を使用して、メッセージシステムを使用せずにサブスクリプションを同期してください。

権限

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、パブリケーション pub_contact に対するユーザ Sam_Singer のサブスクリプションを同期します。

```
SYNCHRONIZE SUBSCRIPTION
   TO pub_contact
   FOR Sam_Singer;
```

関連情報

CREATE SUBSCRIPTION 文 [SQL Remote] [943 ページ]
START SUBSCRIPTION 文 [SQL Remote] [1341 ページ]
STOP SUBSCRIPTION 文 [SQL Remote] [1353 ページ]

1.4.4.276 SYSTEM 文 [Interactive SQL]

Interactive SQL から実行ファイルを起動します。

構文

```
SYSTEM '[ path ] filename'
```

備考

指定した実行ファイルを起動します。パスとファイル名を一重引用符で囲んでください。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、メモ帳の実行ファイルが起動可能なパスにある場合に、メモ帳プログラムを起動します。

```
SYSTEM 'notepad.exe';
```

関連情報

CONNECT 文 [ESQL] [Interactive SQL] [774 ページ]

1.4.4.277 TRIGGER EVENT 文

指定したイベントをトリガします。イベントは、イベントトリガに対して定義したものでも、スケジュールされたイベントでもかまいません。

構文

```
TRIGGER EVENT event-name [ ( parm = value, ... ) ]
```

パラメータ

parm = value

トリガ条件によってイベントハンドラが実行されるときには、データベースサーバが event_parameter 関数を使用して、イベントハンドラにコンテキスト情報を渡すことができます。TRIGGER EVENT 文では、これらのパラメータを明示的に指定し、イベントハンドラのコンテキストをシミュレートできます。

備考

トリガ条件またはスケジュールには、CREATE EVENT 文でアクションを関連付けます。TRIGGER EVENT 文を使用すると、スケジュールされた時刻またはトリガ条件が発生していなくても、イベントハンドラを強制的に実行することができます。TRIGGER EVENT は無効にされたイベントハンドラを実行しません。

各 `value` は文字列です。各 `value` の最大長は、-gp サーバオプションで指定された最大ページサイズによって制限されます。`value` の長さがページサイズを超える場合、ページが一杯になった時点で文字列はトランケートされます。

権限

イベントの所有者であるか、MANAGE ANY EVENT システム権限を持っている必要があります。

関連する動作

なし

標準

ANSI/ISO SQL Standard

標準になし。

例

次の例は、イベントに文字列パラメータを渡す方法を示します。イベントは、トリガされた時刻をデータベースサーバメッセージウィンドウに表示します。

```
CREATE EVENT ev_PassedParameter
HANDLER
BEGIN
  MESSAGE 'ev_PassedParameter - was triggered at ' || event_parameter( 'time' );
END;
TRIGGER EVENT ev_PassedParameter( "Time"=string( current timestamp ) );
```

または、次の `keyword=>value` 構文を使用します。

```
CREATE EVENT ev_PassedParameter
HANDLER
BEGIN
  MESSAGE 'ev_PassedParameter - was triggered at ' ||
event_parameter( 'what_time' );
END;
TRIGGER EVENT ev_PassedParameter( what_time => string( current timestamp ) );
```

関連情報

ALTER EVENT 文 [643 ページ]
CREATE EVENT 文 [805 ページ]
EVENT_PARAMETER 関数 [システム] [356 ページ]

1.4.4.278 TRUNCATE 文

テーブル定義を削除しないで、テーブルからすべてのローを削除します。

構文

```
TRUNCATE
TABLE [ owner.]table-name
| MATERIALIZED VIEW [ owner.]materialized-view-name
```

備考

TRUNCATE 文は、テーブルまたはマテリアライズドビューからすべてのローを削除します。

> **注記**
>
> TRUNCATE TABLE 文はテーブルからすべてのローを削除するため、同期またはレプリケーションに関連するデータベースでは慎重に使用してください。この文は DELETE 文と似ていますが、WHERE 句を指定できません。ただし、TRUNCATE 文の結果としてトリガは発生しません。高速トランケートが可能であることをデータベースサーバが判別すると (このトピックについては後で詳しく説明します)、ローの削除はトランザクションログに記録されず、したがって同期や複製もされません。結果として、同期やレプリケーションが失敗するような矛盾が発生する可能性があります。

TRUNCATE 文を実行した後、オブジェクトのスキーマとすべてのインデックスは、DROP 文が実行されるまで引き続き存在します。スキーマ定義と制約はそのまま残り、トリガと権限は有効なままです。

`table-name` には、ベーステーブルまたはテンポラリテーブルの名前を指定できます。

TRUNCATE TABLE を使用すると、次の基準のすべてが満たされている場合に、データベースサーバにより高速方式のテーブルトランケーションが実行されます。

- そのテーブルへの外部キーも、そのテーブルからの外部キーも存在しない。
- TRUNCATE TABLE 文がトリガ内で実行されない。
- TRUNCATE TABLE 文がアトミック文の中で実行されない。

高速トランケーションが実行される場合、個々の DELETE はトランザクションログに記録されません。また、このオペレーションの前後に COMMIT が実行されます。高速トランケーションが不可能である場合、各 DELETE はトランザクションログに記録されます。

スナップショットトランザクション内では高速トランケーションを使用できません。

即時テキストインデックスが構築されているテーブルか、または即時ビューから参照されているテーブルで TRUNCATE TABLE を使用しようとすると、トランケーションが失敗します。即時テキストインデックス以外のインデックスとマテリアライズドビューでは、このようなことは発生しません。ただし、従属するインデックスやマテリアライズドビューのデータをトランケートしてからテーブルで TRUNCATE TABLE 文を実行し、その後、インデックスとマテリアライズドビューをリフレッシュすることを強くおすすめします。

ベーステーブルとマテリアライズドビューの場合、TRUNCATE 文は、操作がアトミックであるため (すべてのローが削除されるか、まったくされないか)、テーブルへの排他的なアクセスが必要です。トランケートされるテーブルを参照する、事前に開かれたカーソルを閉じ、COMMIT または ROLLBACK を実行して、テーブルへの参照を解除します。

テンポラリテーブルの場合、各ユーザはデータの独自のコピーを所有しているため、TRUNCATE 文を実行するときに排他的アクセスは不要です。

権限

この文を実行する場合は、次の条件の 1 つを満たしている必要があります。

- テーブルの所有者である。
- そのテーブルに対する ALTER 権限を持っている。

- そのテーブルに対する TRUNCATE 権限を持っている。
- ALTER ANY TABLE システム権限を持っている。
- TRUNCATE ANY TABLE システム権限を持っている。
- ALTER ANY OBJECT システム権限を持っている。

関連する動作

- マテリアライズドビューをトランケートする場合は、ビューのステータスを未初期化に変更します。
- TRUNCATE 文は、トリガ削除を実行しません。
- COMMIT は TRUNCATE 文の実行前後に実行されます。
- ローの個々の削除はトランザクションログには記録されません。したがって、TRUNCATE オペレーションはレプリケートされません。SQL Remote レプリケーションまたは Mobile Link リモートデータベースでは、この文を使用しないでください。
- テーブルに DEFAULT AUTOINCREMENT または DEFAULT GLOBAL AUTOINCREMENT と定義されたカラムがある場合、トランケーション操作はそのカラムの次に使用可能な値をリセットします。

標準

ANSI/ISO SQL 標準

TRUNCATE TABLE はオプションの言語機能 F200 です。TRUNCATE MATERIALIZED VIEW は標準にありません。

例

SalesOrderItems テーブルからすべてのローを削除します。

```
TRUNCATE TABLE GROUPO.SalesOrderItems;
```

関連情報

DELETE 文 [1016 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]

1.4.4.279 TRUNCATE TEXT INDEX 文

MANUAL または AUTO REFRESH テキストインデックスのデータを削除します。

構文

```
TRUNCATE TEXT INDEX text-index-name
ON [ owner.]table-name
```

パラメータ

ON clause

テキストインデックスを構築するテーブルの名前。

備考

TRUNCATE TEXT INDEX 文は、テキストインデックス定義を削除しないで、手動テキストインデックスからデータを削除する場合に使用します。たとえば、テキストインデックスのテキスト設定オブジェクトを変更してストップリストを更新する場合、テキストインデックスをトランケートし、そのインデックスが参照しているテキスト設定オブジェクトを変更します。次にテキストインデックスをリフレッシュして、新しいデータをインデックスに移植します。

IMMEDIATE REFRESH (デフォルト) と定義されているテキストインデックスに対して TRUNCATE TEXT INDEX 文を実行することはできません。IMMEDIATE REFRESH テキストインデックスの場合は、インデックスを削除する必要があります。

TRUNCATE TEXT INDEX ではテーブルへの排他アクセスが必要です。トランケートされるテーブルを参照するカーソルが開いている場合は、それらのカーソルをすべて閉じ、COMMIT または ROLLBACK 文を実行してテーブルへの参照を解除します。

権限

そのテーブルの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- そのテーブルに対する ALTER 権限
- ALTER ANY INDEX システム権限
- ALTER ANY TABLE システム権限

関連する動作

オートコミット

標準

ANSI/ISO SQL 標準

標準になし。

例

最初の文は、txt_index_manual テキストインデックスを作成します。2 番目の文では、テキストインデックスにデータを移植します。3 番目の文では、テキストインデックスデータをトランケートします。

```
CREATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation ( Description )
   MANUAL REFRESH;
REFRESH TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation;
TRUNCATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation;
```

トランケートされたテキストインデックスは、次回リフレッシュされるとき、データが再移植されます。

関連情報

CREATE TEXT INDEX 文 [976 ページ]
ALTER TEXT INDEX 文 [724 ページ]
DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]

1.4.4.280 TRY 文

複合文のエラー処理を実装します (TRY ブロックでエラーが発生した場合、CATCH ブロックに囲まれている別の文のグループに制御を渡します)。

構文

```
[ statement-label : ]
BEGIN TRY
   [ local-declaration; ... ]
   [ statement-list ]
END TRY
BEGIN CATCH
   [ statement-list ]
END CATCH
```

```
local-declaration :
variable-declaration
| cursor-declaration
| exception-declaration
| temporary-table-declaration
```

: 有効な DECLARE 文

パラメータ

statement-label

終了の `statement-label` を指定する場合は、開始の `statement-label` と一致させる必要があります。LEAVE 文を使うと、複合文に続く最初の文から実行を再開できます。プロシージャまたはトリガの本文である複合文は、プロシージャまたはトリガの名前と同じ暗黙のラベルを持っています。

local-declaration

BEGIN TRY のすぐ後で、複合文は複合文の中にだけ存在するオブジェクトをローカルに宣言できます。複合文は、変数、カーソル、テンポラリテーブル、または例外に対してローカル宣言を行います。ローカル宣言は、複合文またはその中でネストされる複合文の中のどの文からでも参照できます。複合文のローカル宣言は、文の例外ハンドラに表示されます。ローカル宣言は、複合文の中から呼び出される他のプロシージャには表示されません。

variable-declaration

有効な DECLARE 文。

exception-declaration

有効な DECLARE 文。

cursor-declaration

有効な DECLARE CURSOR 文。

temporary-table-declaration

有効な DECLARE LOCAL TEMPORARY TABLE 文。

備考

CATCH ブロックは、TRY 文のエラーハンドラです。

TRY...CATCH 文は、BEGIN...END 文を使用できる場所であれば、どこでもネストして使用できます。TRY ブロック内の文は、on_tsql_error と continue_after_raiserror のデータベースオプション、および CREATE PROCEDURE 文の ON EXCEPTION RESUME 句を無視します。TRY...CATCH 文はアトミックではありません。

TRY ブロックでエラーが発生していない場合、CATCH ブロックはスキップされます。制御は CATCH ブロックに続く文に渡されるか、そのような文が存在しない場合は呼び出し側に渡されます。TRY ブロック内の文の 1 つでエラーが発生した場合、制御は CATCH ブロック内の最初の文に渡されます。CATCH ブロックパラメータが完了したら、制御は CATCH ブロックに続く文に渡されるか、そのような文が存在しない場合は呼び出し側に渡されます。エラーを生成した文に優先される文の影響は、例外ハンドラがエラーを生成してアトミックブロック内でネストされないか、明示的な ROLLBACK 文が呼び出されない場合は、戻されません。この場合、アトミックトランザクションブロック内のすべての文が戻されます。

CATCH ブロックのエラーは、エラーを生成した文が追加の TRY...CATCH 文で囲まれない場合、接続設定とプロシージャ設定に従って処理されます。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

これらの例は、次のテーブルを使用します。

```
CREATE TABLE t( col1 DOUBLE );
```

次の複合文を実行すると、値 6 がテーブル t に挿入されます。

```
BEGIN TRY
    DECLARE val INT;
    SET val = 0;
    INSERT INTO t VALUES( 1 / val );
    -- This statement will not be executed
    INSERT INTO t VALUES( val );
END TRY
BEGIN CATCH
    SET val = 6;
    INSERT INTO t VALUES( val );
END CATCH;
```

`CALL proc1(10);` を使用して次のプロシージャを実行すると、次の値がテーブル t に挿入されます。

-10
10

```
CREATE PROCEDURE DBA.proc1( v INTEGER )
BEGIN TRY
    DECLARE local_val INTEGER = 0;
    INSERT INTO t VALUES(-v);
    SET local_val = v / local_val;
    -- This statement will not be executed
    MESSAGE 'The value is ', v;
END TRY
BEGIN CATCH
    INSERT INTO t VALUES(v);
END CATCH;
```

関連情報

DECLARE 文 [1010 ページ]

DECLARE CURSOR 文 [ESQL] [SP] [1001 ページ]
DECLARE LOCAL TEMPORARY TABLE 文 [1007 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]
CONTINUE 文 [778 ページ]
SIGNAL 文 [SP] [1332 ページ]
RESIGNAL 文 [SP] [1268 ページ]
RAISERROR 文 [1246 ページ]
BEGIN 文 [TSQL] [748 ページ]

1.4.4.281 UNION 文

2 つ以上の SELECT 文またはクエリ式の結果を結合します。

構文

```
[ WITH temporary-views ] query-block
  UNION [ ALL | DISTINCT ] query-block
[ ORDER BY [ integer | select-list-expression-name ] [ ASC | DESC ], ... ]
[ FOR XML xml-mode ]
[ OPTION( query-hint, ... ) ]
```

```
temporary-views :
  regular-view, ...
| RECURSIVE { regular-view | recursive-view }, ...
```

```
regular-view :
  view-name [ ( column-name, ... ) ]
  AS ( query-block )
```

```
recursive-view :
  view-name ( column-name, ... )
  AS ( initial-query-block UNION ALL recursive-query-block )
```

query-block: SQL 構文の共通要素のマニュアルを参照

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

```
option-name : identifier
```

```
option-value :
hostvar（許可されたインジケータ）
| string
| identifier
| number
```

パラメータ

WITH or WITH RECURSIVE clause

1 つ以上の共通テーブル式を定義します。共通テーブル式はテンポラリビューでもあり、文の残りの部分に使用されます。これらの式は、非再帰的か自己再帰的のいずれかに指定できます。再帰的な共通テーブル式は、RECURSIVE キーワードが指定されている場合にのみ、単独で表示されるか、非再帰的なテーブル式と混合で表示されます。相互再帰的な共通テーブル式は、サポートされていません。

SELECT クエリブロックが次のいずれかの場所に表示されている場合のみ、この句を使用できます。

- ビュー定義の最上位レベルの SELECT クエリブロックを含む最上位レベルの SELECT クエリブロック内
- INSERT クエリブロック内の最上位レベルの SELECT 文内
- 任意のタイプの SQL 文で派生テーブルを定義しているネストされた SELECT クエリブロック内

再帰的な式は、初期のサブクエリと再帰的なサブクエリから構成されます。初期クエリは、ビューのスキーマを暗黙的に定義します。再帰的なサブクエリには、FROM 句内のビューへの参照を入れてください。それぞれの反復中に、この参照によって前の反復でビューに追加されたローだけが参照されます。参照は、外部ジョインの NULL 入力側では表示できません。再帰的な共通テーブル式は、集合関数を使用できません。また、GROUP BY、ORDER BY、DISTINCT の各句を含むことはできません。

リモートテーブルでは WITH 句はサポートされません。WITH 句は、INTERSECT、UNION、EXCEPT クエリブロック内でも使用されることがあります。

この機能は Watcom SQL ダイアレクトでのみ使用可能です。

```
WITH RECURSIVE
  manager ( EmployeeID, ManagerID,
            GivenName, Surname, mgmt_level ) AS
( ( SELECT EmployeeID, ManagerID,       -- initial subquery
           GivenName, Surname, 0
    FROM Employees AS e
    WHERE ManagerID = EmployeeID )
  UNION ALL
  ( SELECT e.EmployeeID, e.ManagerID,   -- recursive subquery
           e.GivenName, e.Surname, m.mgmt_level + 1
    FROM Employees AS e JOIN manager AS m
     ON   e.ManagerID =  m.EmployeeID
      AND e.ManagerID <> e.EmployeeID
      AND m.mgmt_level < 20 ) )
SELECT 'Manager', * FROM manager
WHERE mgmt_level > 0
UNION ALL
SELECT 'Employee', * FROM manager
WHERE mgmt_level = 0
ORDER BY mgmt_level, Surname, GivenName;
```

ORDER BY clause

結果の最終的な順序を指定します。UNION ORDER BY 句では、最初の `query-block` で定義されたとおりのカラム名を使用します。ただし、`query-block` に含まれるテーブル参照を ORDER BY 句で使用することはできません。

たとえば、以下の文を AS LastName を指定せずに実行すると、ORDER BY Surname という指定は可能ですが、ORDER BY t1.Surname という指定はできません。

```
SELECT t1.Surname AS LastName
FROM GROUPO.Employees t1
UNION
SELECT t1.Surname
FROM GROUPO.Customers t1
ORDER BY LastName;
```

FOR XML clause

FOR XML 句については、SELECT 文のマニュアルを参照してください。

OPTION clause

文を実行するためのヒントを指定します。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- `option-name` = `option-value`.クエリテキスト内の `OPTION( isolation_level = ... )` の指定は、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。

備考

UNION ALL は、2 つのクエリブロックの結果を連結して、単一の (より大きな) 結果セットにします。それぞれのクエリブロックはネストされることがあります。UNION DISTINCT は最終結果から重複ローを削除します。重複ローを削除するには余分な処理が必要なため、可能であれば UNION の代わりに UNION ALL を使用してください。UNION DISTINCT は UNION と同じです。

2 つの `query-block` の結果セットが UNION 対応になっている必要があります。つまり、それぞれの SELECT リストに同じ数の項目があり、個々の式のデータ型が比較可能となっている必要があります。2 つの SELECT リスト内の対応する項目が異なるデータ型の場合、データベースサーバは結果の中から対応するカラムのデータ型を選択し、各 `query-block` のカラムを自動的にそれぞれ変換します。

表示されるカラム名は最初の `query-block` に表示されるカラム名と同じであり、これらの名前は ORDER BY 句と一致する式の名前を判断するために使用されます。結果セットのカラム名をカスタマイズする別の方法として、共通のテーブル式 (WITH 句) を使用する方法があります。

権限

`query-block` で指定されているオブジェクトの所有者であるか、論理和にジョインするオブジェクトに対する SELECT 権限を持っている必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

コア機能。UNION とともに DISTINCT キーワードを指定する方法は、オプションの ANSI/ISO SQL 言語機能 T551 です。UNION とともに ORDER BY 句を指定する方法は、ANSI/ISO SQL 言語機能 F850 です。ORDER BY 句を含む `query-block` は、ANSI/ISO SQL 機能 F851 の構成要素です。ロー制限句 (SELECT TOP または LIMIT) を含むクエリブロックは、コンテキストに応じて、オプションの ANSI/ISO SQL 言語機能 F857 または F858 を構成します。FOR XML 句と OPTION 句は標準にありません。

Transact-SQL

UNION と UNION ALL は Adaptive Server Enterprise によってサポートされます。FOR XML 句と OPTION 句はサポートされません。

例

従業員と顧客のそれぞれの姓すべてをリストします。

```
SELECT Surname
FROM GROUPO.Employees
UNION
SELECT Surname
FROM GROUPO.Customers;
```

関連情報

一般的な SQL 構文要素 [608 ページ]
SELECT 文 [1291 ページ]
EXCEPT 文 [1088 ページ]
INTERSECT 文 [1177 ページ]

1.4.4.282 UNLOAD 文

データソースからファイルにデータをアンロードします。

構文

```
UNLOAD data-source
{ TO filename
```

```
    | INTO FILE filename
    | INTO CLIENT FILE client-filename
    | INTO VARIABLE variable-name }
[ unload-option ... ]
```

```
data-source :
[ FROM ] [ TABLE ] [ owner.]table-name
| [ FROM ] [ MATERIALIZED VIEW ] [ owner.]materialized-view-name
| select-statement
```

```
filename : string | variable
```

```
client-filename : string | variable
```

```
unload-option :
APPEND { ON | OFF }
| BYTE ORDER MARK { ON | OFF }
| { COMPRESSED | NOT COMPRESSED }
| COLUMN DELIMITED BY string
| DELIMITED BY string
| ENCODING encoding
| { ENCRYPTED KEY 'key' [ ALGORITHM 'algorithm' ] | NOT ENCRYPTED }
| ESCAPE CHARACTER character
| ESCAPES { ON | OFF }
| FORMAT { TEXT | BCP }
| HEXADECIMAL { ON | OFF }
| ORDER { ON | OFF }
| QUOTE string
| QUOTES { ON | OFF | ALL }
| ROW DELIMITED BY string
| WITH COLUMN NAMES
```

```
encoding : string
```

```
algorithm :
'AES' | 'AES256' | 'AES_FIPS' | 'AES256_FIPS'
```

パラメータ

TO clause

データのアンロード先ファイルの名前。`filename` パスは、データベースサーバの起動ディレクトリへの相対パスです。このファイルが存在しない場合は、作成されます。ファイルがすでに存在する場合は、APPEND ON が指定されていないかぎり、そのファイルが上書きされます。

INTO FILE clause

セマンティック上は TO `filename` と同じです。

INTO CLIENT FILE clause

データのアンロード先となるクライアントコンピュータ上のファイル。このファイルが存在しない場合は作成されます。ファイルがすでに存在する場合は、APPEND ON が指定されていないかぎり、そのファイルが上書きされます。パスは、クライアントコンピュータ上で、クライアントアプリケーションの現在の作業フォルダとの相対パスとして解決されます。

INTO VARIABLE clause

データのアンロード先変数。変数がすでに存在し、CHAR、NCHAR、または BINARY 型である必要があります。APPEND オプションを指定すると、アンロードされたデータが変数の現在の内容に連結されます。

APPEND clause

APPEND を ON にすると、アンロードされたデータは、指定されたファイルの最後に追加されます。APPEND を OFF にすると、指定されたファイルの内容がアンロードされたデータに置換されます。APPEND はデフォルトでは OFF になっています。COMPRESSED または ENCRYPTED 句を指定したときは、この句を指定できません。また、追加されるファイルが圧縮または暗号化されているときは使用できません。

BYTE ORDER MARK clause

この句は、バイトオーダーマーク (BOM) を書き込むどうかを制御するときに指定します。アンロードの送信先がローカルファイルまたはクライアントファイルの場合、このオプションはデフォルトで ON です。BYTE ORDER MARK オプションが ON で ENCODING が UTF-8 または UTF-16 の場合、BOM は書き込まれません。BYTE ORDER MARK が OFF の場合、BOM はアンロードされません。

COMPRESSED clause

データを圧縮するかどうかを指定します。デフォルトは NOT COMPRESSED です。データを追加する場合 (APPEND ON) は、データを圧縮できません。

COLUMN DELIMITED BY and DELIMITED BY clauses

カラム間で使用される文字列です。デフォルトのカラムデリミタはカンマです。最長で 255 バイトの文字列を指定して、代替のカラムデリミタを指定します。

ENCODING clause

すべてのデータベースデータは、データベースの文字エンコードから、指定した CHAR または NCHAR エンコードへと変換されます。ENCODING が指定されていない場合、データベースの CHAR エンコードが使用されます。

変換エラーがアンロード操作時に発生した場合、on_charset_conversion_failure オプションの設定に基づいてレポートされます。

データにバイトオーダーマークを含めるには、BYTE ORDER 句を指定します。

ENCRYPTED clause

データを暗号化するかどうかを指定します。NOT ENCRYPTED (デフォルト) を指定した場合、データは暗号化されません。キーは指定するがアルゴリズムは指定しないで ENCRYPTED KEY を使用すると、データは AES128 と指定したキーを使用して暗号化されます。このキーには文字列または変数名を使用できます。キーとアルゴリズムを指定して ENCRYPTED KEY を使用すると、データは指定したキーとアルゴリズムを使用して暗号化されます。CREATE DATABASE 文で許可されたアルゴリズムであれば、どのアルゴリズムでも使用できます。SIMPLE 難読化アルゴリズムは指定できません。

データを追加する場合 (APPEND ON)、データは暗号化できません。

ESCAPES clause

ESCAPES を ON (デフォルト) にすると、データベースサーバはエスケープ文字を書き込みます。改行文字は ¥n との組み合わせとして書き込まれ、他の文字はタブ文字の ¥x09 のような 16 進の ASCII のコードとしてデータにインクルードされます。2 つの円記号 (¥) は 1 つの円記号として書き込まれます。円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字として書き込まれます。たとえば、¥q であれば、円記号と q が挿入されます。エスケープ文字に指定する文字列は 1 マルチバイト文字を超えないようにすることをおすすめします。

FORMAT clause

データを TEXT フォーマットまたは BCP アウトフォーマットで出力します。TEXT を選択すると、出力行はテキスト文字として書き込まれ、1 行あたり 1 つのローで構成され、カラムデリミタ文字列によって値が区切られます。BCP を選択すると、

BLOB などのデータが、Adaptive Server Enterprise で使用する BCP 入力ファイルとしてエクスポートされます。デフォルトのフォーマットは TEXT です。

HEXADECIMAL clause

デフォルトでは、HEXADECIMAL は ON です。バイナリカラム値は 0x*nnnnnn*... の形式で書き込まれます。ここで、0x は 1 つのゼロの後に 1 つの x が続きます。*n* は 16 進数の数字です。マルチバイト文字セットを扱う場合は、HEXADECIMAL ON を使用することが重要です。

HEXADECIMAL 句は、FORMAT TEXT 句を指定した場合にのみ使用できます。

ESCAPE CHARACTER clause

この句は、データで使用するエスケープ文字を指定するときに使用します。16 進のコードおよび記号として書き込まれる文字に使用するデフォルトのエスケープ文字は、円記号 (¥) です。たとえば、¥x0A は改行文字です。エスケープ文字は、ESCAPE CHARACTER 句を使って変更することができます。

エスケープ文字に指定する文字列は 1 マルチバイト文字を超えないようにすることをおすすめします。

ORDER clause

ORDER を ON (デフォルト) に設定すると、エクスポートされたデータはクラスタドインデックス (存在する場合) の順に配列されます。クラスタドインデックスが存在しない場合、エクスポートされたデータはプライマリキー値で順序付けられます。ORDER を OFF に設定すると、ORDER BY 句を使わずにテーブルから選択したときと同じ順序でデータがエクスポートされます。ORDER を ON に設定すると、エクスポートは遅くなります。ただし、LOAD TABLE 文を使って再ロードする方が、インデックス手順が単純であるため速くなります。

UNLOAD `select-statement` の場合、ORDER 句は無視されます。ただし、SELECT 文で ORDER BY 句を指定すると、データを順序付けできます。

QUOTE clause

QUOTE 句は TEXT データ専用です。`string` は文字列値を囲みます。デフォルトは一重引用符 (') です。QUOTES ALL を使用する場合、引用符は一重引用符 (') または二重引用符 (") のみとなります。

QUOTES clause

QUOTES をオン (デフォルト) に設定すると、エクスポートされる文字列がすべて引用符文字列 (デフォルトでは、一重引用符 (アポストロフィ)) で囲まれます。QUOTES ALL を指定すると、引用符文字列は文字列だけでなく、すべての値の前後に置かれます。引用符を非表示にするには、QUOTES OFF を使用します。

ROW DELIMITED BY clause

この句は、レコードの末尾を示す文字列を指定するときに使用します。デフォルトのデリミタ文字列は、Windows の場合は '¥x0d¥x0a' (CR/LF) で、Unix の場合は '¥x0a' (LF) です。ただし、長さ 255 バイトまでの任意の文字列を使用できます。たとえば、`ROW DELIMITED BY '###'` のように指定します。タブで区切った行を指定する場合、タブ文字を表す 16 進のエスケープシーケンス (序数 9) を使用して、`ROW DELIMITED BY '¥x09'` のように指定します。デリミタ文字列に新規行文字 (¥n) が含まれる場合、結果として改行 (LF) 文字のみとなります。Windows システムの場合、'¥x0d¥x0a' を使用してテキストファイルにキャリッジリターンと改行 (CR/LF) を書き出すのが通例です。

WITH COLUMN NAMES clause

WITH COLUMN NAMES 句は、ファイルまたは変数の先頭行にカラム名を挿入します。

備考

UNLOAD `select-statement` 文では、SELECT 文からのデータをカンマ区切りのファイルにエクスポートできます。SELECT 文に ORDER BY 句が指定されていない場合は、結果セットを順序付けできません。

UNLOAD TABLE 文は、データベーステーブルまたはマテリアライズドビューからファイルへの効率的な大量エクスポートができます。UNLOAD TABLE 文は、Interactive SQL 文の OUTPUT より効率的で、すべてのクライアントアプリケーションから呼び出すことができます。

INTO FILE または INTO CLIENT FILE が指定されたかどうかに応じて、データベースサーバまたはクライアントアプリケーションはそれぞれ、指定されたファイルに対する書き込みのオペレーティングシステムパーミッションが付与されている必要があります。**Tabular Data Stream** (**TDS**) 接続に対しては、INTO CLIENT FILE はサポートされていません。

バイナリデータ型のテーブルカラムをアンロードする場合、UNLOAD TABLE は **¥x**`nnnn` の形式で 16 進文字列を書き込みます。`n` は 16 進数字です。

プロキシテーブルを持つデータベースをアンロードして再ロードする場合は、ユーザがローカルデータベースとリモートデータベースの両方に対して同じパスワードを持っていても、外部ログインを作成して、ローカルユーザをリモートユーザにマッピングしてください。外部ログインがない場合は、リモートサーバに接続できないため再ロードが失敗します。

変数をアンロードする場合 (INTO VARIABLE)、出力は次のように文字セットに変換されます。

CHAR

変数に CHAR エンコードで書き込みます。ENCODING 句は CHAR エンコードと一致する必要があります。

NCHAR

変数に NCHAR エンコードで書き込みます。ENCODING 句は NCHAR エンコードと一致する必要があります。

BINARY

変数に BINARY エンコードで書き込みます。ENCODING は BINARY エンコーディングと一致する必要があります。それ以外の場合、CHAR エンコーディングが使用されます。

アンロードされるデータの圧縮と暗号化を選択すると、最初にデータが圧縮されます。

UNLOAD TABLE は、テーブル全体またはマテリアライズドビュー全体に排他ロックをかけます。

この文の実行中に、進行状況メッセージの表示を要求できます。

また、Progress 接続プロパティを使用して、文がどの程度実行されたかを確認することもできます。

UNLOAD 文の実行時に、アンロードされるローの数が @@rowcount 変数に設定されます。

日付値の最大精度を保持するには、date_format を YYYY-MM-DD に設定します。

TIMESTAMP 値の最大精度を保持するには、timestamp_format を YYYY-MM-DD HH:NN:SS.SSSSSS に設定します。

TIMESTAMP WITH TIME ZONE 値の最大精度を保持するには、timestamp_with_time_zone_format を YYYY-MM-DD HH:NN:SS.SSSSSS+HH:NN に設定します。

ディスクサンドボックスが有効になっている場合、データベースの操作はメインデータベースファイルが格納されているディレクトリに制限されます。

権限

変数へのアンロード時には、権限は必要ありません。それ以外の場合、必要な権限は、データベースサーバの -gl オプションによって異なります。

- -gl オプションが ALL に設定されている場合は、そのテーブルの所有者であるか、そのテーブルに対する SELECT 権限を持っているか、または SELECT ANY TABLE システム権限を持っていることが必要です。
- -gl オプションが DBA に設定されている場合は、SELECT ANY TABLE システム権限が必要です。
- -gl オプションが NONE に設定されている場合、UNLOAD は使用できません。

クライアントコンピュータにあるファイルにアンロードする場合

- WRITE CLIENT FILE 権限が必要です。
- そのファイルが置かれているディレクトリに対する書き込みパーミッションが必要です。
- allow_write_client_file データベースオプションが有効になっている必要があります。
- WRITE_CLIENT_FILE 機能が有効になっている必要があります。

関連する動作

なし。このクエリは現在の独立性レベルで実行されます。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は、Products テーブルの内容を、UTF-8 でエンコードされたファイル `productsT.dat` にアンロードします。

```
UNLOAD TABLE GROUPO.Products TO 'c:¥¥temp¥¥productsT.dat' ENCODING 'UTF-8';
```

次の例は、変数 @myProducts を作成してから、Products.Name カラムの変数をアンロードします。

```
CREATE VARIABLE @myProducts LONG VARCHAR;
UNLOAD SELECT NAME FROM GROUPO.Products INTO VARIABLE @myProducts ESCAPE
CHARACTER '!';
```

関連情報

CREATE DATABASE 文 [781 ページ]
LOAD TABLE 文 [1182 ページ]

PASSTHROUGH 文 [SQL Remote] [1230 ページ]
OUTPUT 文 [Interactive SQL] [1222 ページ]

1.4.4.283 UNPIVOT 句

FROM 句 (FROM unpivoted-derived-table expression) のテーブル式の準拠型カラムを、派生テーブルのローに列行変換します。UNPIVOT はデータの正規化に使用されます。たとえば、テーブルの複数のカラムに格納された類似のデータを持つ場合、それらを 1 つのカラムに返したい場合があります。

構文

```
FROM unpivoted-derived-table
unpivoted-derived-table :
unpivot-source-table UNPIVOT [ { INCLUDE | EXCLUDE } NULLS ]  ( unpivot-clause )
[ AS ] correlation-name
unpivot-clause :
unpivot-value-clause unpivot-for-clause unpivot-in-clause
unpivot-value-clause :
value-column
| ( value-column [,...] )
unpivot-for-clause :
FOR unpivot-column
unpivot-in-clause :
IN( unpivot-old-column [[ AS ] unpivot-old-column-alias ] [,...] )
| IN(( unpivot-old-column [,...] ) [[ AS ] unpivot-old-column-alias ] [,...] )
```

パラメータ

{ INCLUDE | EXCLUDE } NULLS

value-column の結果に NULL 値を含めるか、除外するかを指定します。デフォルトの動作は EXCLUDE NULLS です。

unpivot-value-clause

UNPIVOT された値を持つ列行変換された派生テーブルの新しいカラムの名前を指定します。

unpivot-for-clause

データに対して UNPIVOT を実行するカラムを指定します。

unpivot-in-clause

データに対して UNPIVOT を実行する unpivot-for-clause の値を指定します。

備考

列行変換された派生テーブルには、unpivot-source-table のカラムのサブセットと、unpivot-value-clause および unpivot-column で指定した追加のカラムが含まれます。IN 句のカラムは、unpivot-source-table 内にある必要がありますが、列行変換された派生テーブルには存在しません。

IN 句に I 個の要素がある場合、unpivot-source-table の各ローが列行変換された派生テーブルの I 個のローに変換されます。つまり、IN 句の項目に対応するそれぞれの値のタプルは、列行変換された派生テーブルに新しいローを生成します。各ローの unpivot-column の値が、その項目の IN 句のエイリアスの値に設定されます。一方、unpivot-value-clause で指定した新しいカラムが、その項目の元となるローの値のタプルに設定されます。

IN 句の各項目は、IN 句の他の項目に準拠したドメインが必要です。unpivot-value-clause で指定した新しいカラムのデータ型は、これらの準拠ドメインのスーパータイプに対応します。unpivot-column の値のデータ型は NCHAR で、その値は IN 句で定義したエイリアスです。

権限

unpivot-source-table で参照されるオブジェクトに対する SELECT 権限が必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

取引先担当者の名前やさまざまな電話番号を含む MyCustomers というテーブルがあるとします。次の文は、そのようなテーブルを作成し、架空のデータを集計し、テーブルに対してクエリを実行します。

```
CREATE TABLE MyCustomers
( ContactID INT PRIMARY KEY,
  LastName VARCHAR(64),
  FirstName VARCHAR(64),
  Home VARCHAR(32),
  Mobile VARCHAR(32),
  AltPhone VARCHAR(32)
);
INSERT MyCustomers
  ( ContactID, LastName, FirstName, Home, Mobile, AltPhone )
VALUES
  ( 1, 'Bringle', 'Susan', '111-593-1451', '222-693-7620', NULL ),
```

```
  ( 2, 'Hoffsteter', 'Garth', '113-249-6622', NULL, NULL),
  ( 3, 'Zenibar', 'Austin', NULL, '171-765-8730', '888-536-5324' );
SELECT * FROM MyCustomers;
```

ContactID	LastName	FirstName	Home	Mobile	AltPhone
1	Bringle	Susan	111-593-1451	222-693-7620	(NULL)
2	Hoffsteter	Garth	113-249-6622	(NULL)	(NULL)
3	Zenibar	Austin	(NULL)	171-765-8730	888-555-5555

取引先担当者ごとに、連絡先番号が Home、Mobile、および AltPhone の 3 つのカラムに格納されます。

得意先営業員が呼び出しに使用した電話番号に関する統計を収集する必要があるとします。類似のバータ番号などの傾向を識別できるよう番号順にソートされた、呼び出しに使用されたすべての電話番号のリストを参照するとします。必要な結果を取得するには、Home、Mobile、および AltPhone のカラムを、電話番号別にソートされ、電話番号ごとに 1 つのローを含む単一のカラムに列行変換する必要があります。SELECT 文は次のようになります。

```
SELECT ContactID, PhoneNumber, PhoneType
FROM
(
  SELECT ContactID, Home, Mobile, AltPhone
  FROM MyCustomers
) AS MyUnpivotSource
UNPIVOT EXCLUDE NULLS
(
  PhoneNumber FOR PhoneType IN ( Home, Mobile, AltPhone )
)
AS MyUnpivotedTable
ORDER BY PhoneNumber;
```

ContactID	PhoneNumber	PhoneType
1	111-593-1451	Home
2	113-249-6622	Home
3	171-765-873	Mobile
1	222-693-7620	Mobile
3	888-536-5324	AltPhone

UNPIVOT 操作によって、MyCustomers テーブルの 3 つの電話番号データのカラムが、PhoneNumber という単一の電話番号のカラム (この例では `unpivot-value-clause`) に正規化されています。PhoneType カラム (この例では `unpivot-column`) の値は、`unpivot-source-table` で指定した電話番号値のカラムによって決定します。

次のクエリにより、取引担当者の電話番号と Customers テーブルの顧客情報から、電話番号がリストされた列行変換された派生テーブルが抽出されます。この派生テーブルでは、データの起源 (顧客の電話番号または取引担当者の電話番号) が People アンピボットカラムに記録されます。

```
SELECT DISTINCT *
FROM
 ( SELECT CO.CustomerID,
          C.City customer_city,
          C.State customer_state,
          c.Phone customer_phone,
          CO.City contact_city,
          CO.State contact_state,
          CO.Phone contact_phone
   FROM Customers C
```

```
   JOIN Contacts CO
   WHERE  C.State IN ( 'NJ', 'NY' ) ) UnpivotSourceTable
UNPIVOT
 ( ( City, State, Phone ) FOR People IN
     (
       ( customer_city, customer_state, customer_phone ) AS CustomerPhone,
       ( contact_city, contact_state, contact_phone ) AS ContactPhone
     )
 ) UnpivotedDerivedTable
ORDER BY CustomerID, People;
```

たとえば、UNPIVOT を実行するソーステーブル UnpivotSourceTable には、次のロー (およびその他) が含まれます。

CustomerID	customer_city	customer_state	customer_phone	contact_city	contact_state	contact_phone
101	Kingston	NJ	2015558966	Hespeler	NJ	6035550988

クエリにより、このデータが、以下の結果セットテーブルの CustomerID 101 で識別される新しいローに列行変換されます。

CustomerID	People	City	State	Phone
101	ContactPhone	Hespeler	NJ	6035550988
101	CustomerPhone	Kingston	NJ	2015558966
111	ContactPhone	Cornwall	NY	6175557956
111	ContactPhone	Cornwall	NY	6175558890
111	CustomerPhone	Hastings	NY	3155554486
116	ContactPhone	Cornwall	NY	6175552222
116	ContactPhone	Cornwall	NY	6175559877
116	CustomerPhone	Stayner	NY	9145553817
147	ContactPhone	Hespeler	NJ	6035551200
147	CustomerPhone	Campbellford	NJ	9085556021

関連情報

FROM 句 [1112 ページ]
PIVOT 句 [1232 ページ]

1.4.4.284 UPDATE (位置付け) 文 [ESQL] [SP]

カーソルの現在位置のデータを変更します。

構文

Embedded SQL

```
UPDATE WHERE CURRENT OF cursor-name
{ USING [ SQL ] DESCRIPTOR sqlda-name | { [ FROM ] | [ USING ] } hostvar-
list }
```

ストアドプロシージャ

```
UPDATE update-table, ...
SET set-item, ...
WHERE CURRENT OF cursor-name
```

```
hostvar-list : indicator variables allowed
```

```
update-table :
[owner-name.]object-name [ [ AS ] correlation-name ]
```

```
set-item :
[ correlation-name.]column-name = { expression | DEFAULT }
| [ owner-name. ]object-name.column-name = { expression | DEFAULT }
```

```
object-name: identifier （テーブルまたはビューの名前）
```

```
sqlda-name : identifier
```

パラメータ

UPDATE clause

各 update-table は、次のように、カーソルのクエリにあるテーブルとマッチングされます。

1. 相関名を指定した場合、同じ table-or-view-name と同じ correlation-name を持つカーソルのクエリにあるテーブルとマッチングされます。
2. 相関名がない同じ table-or-view-name を持つカーソルのクエリにテーブルがあるとき、または table-or-view-name と同じ相関名を持つとき、更新テーブルはカーソルのクエリのテーブルとマッチングされます。
3. 更新テーブルと同じ table-or-view-name を持つカーソルのクエリに単一のテーブルがある場合、更新テーブルはカーソルのクエリにあるテーブルとマッチングされます。

カラムにデフォルトが定義されている場合は、SET 句を使用してカラムをデフォルト値に設定してください。

USING DESCRIPTOR clause

変数に代入する場合、その変数がすでに宣言済みであることが必要です。また、変数名はアット記号 (@) で始めます。変数の代入とカラムの代入は混在させることができ、任意の数を使用できます。

SET clause

set-item のカラムは、更新対象となるテーブルまたはビューに配置してください。SET リストの割り当ての左側の名前が、更新されたテーブルのカラムと変数名に一致する場合、文はそのカラムを更新します。set-item カラムは、他のテーブルまたはビューからエイリアスまたはカラムを参照できません。更新するテーブルまたはビューにカーソル指定で相関名が与えられている場合は、SET 句に相関名を使用してください。

各 set-item は 1 つの update-table に関連付けられ、カーソルのクエリで一致するテーブルの対応するカラムは修正されます。expression は、UPDATE リストに指定されているテーブルのカラムを参照します。また、+、-、...、COALESCE、IF などの演算子を使用すると、定数、ホスト変数、変数、クエリの SELECT リストからの式、またはそれらを組み合わせて使用できます。expression は、カーソルのクエリまたは UPDATE リストにないカーソルのクエリのその他のテーブルのカラムから式のエイリアスを参照できません。subselect、サブクエリ述部、集合関数は、set-item に使用できません。

備考

この形式の UPDATE 文は、指定されたカーソルの現在のローを更新します。現在のローを、カーソルからフェッチされた最後のローとして定義します。カーソルに対する最後のオペレーションを位置付けられた DELETE 文にしないでください。

Embedded SQL 構文では、SQLDA からのカラム、またはホスト変数リストからの値は、指定したカーソルから返されるカラムに 1 対 1 で対応します。SQLDA の sqldata ポインタが NULL ポインタの場合、該当する SELECT リスト項目は更新されません。

ストアドプロシージャ構文では、要求されたカラムは指定されたクエリの現在のローに指定された値に設定されています。カラムは、指定した開いているカーソルの SELECT リストの中になくてもかまいません。このフォーマットは準備できます。

また、変数に代入する場合、その変数がすでに宣言済みであることが必要です。また、変数名はアット記号 (@) で始めます。変数の代入とカラムの代入は混在させることができ、任意の数を使用できます。SET リストの代入の左辺にある名前が、更新されたテーブルのカラムと変数名に一致する場合、UPDATE 文はカラムを更新します。

USING DESCRIPTOR、FROM hostvar-list、hostvar 形式を使用できるのは、ESQL の場合だけです。

権限

更新されるテーブルの所有者であるか、変更されるカラムに対する UPDATE 権限を持っているか、または UPDATE ANY TABLE システム権限を持っていることが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

Embedded SQL 構文は標準にありません。

ストアドプロシージャ構文は、コア機能です。Embedded SQL プログラムで使用する場合、この構文はオプションの SQL 言語機能 B031、"Basic dynamic SQL" の一部です。更新されるテーブルを複数指定できる機能は、標準にありません。

更新できるカーソルの範囲は、ansi_update_constraints オプションの設定によって異なります。順序付けされている (SQL クエリに ORDER BY 句が指定されている) カーソルの位置付け更新を実行する機能は、オプションの ANSI/ISO SQL 言語機能 F831、"Full cursor update" を構成します。より複雑な SQL 定数に対する位置付け更新の実行には、標準にない追加の拡張を伴う場合があります。

例

次は、架空のカーソル emp_cursor で実行される UPDATE 文の例です。

```
UPDATE GROUPO.Employees
SET Surname = 'Jones'
WHERE CURRENT OF emp_cursor;
```

関連情報

INSERT 文 [1167 ページ]
LOAD TABLE 文 [1182 ページ]
MERGE 文 [1204 ページ]
DELETE 文 [1016 ページ]
DELETE 文 (位置付け) [ESQL] [SP] [1014 ページ]
UPDATE 文 [1391 ページ]

1.4.4.285 UPDATE 文 [SQL Remote]

データベース内のデータを変更します。

構文

Message Agent によって実行される単一テーブルの単一ローの更新

```
UPDATE table-name
SET column-name = expression, ...
[ VERIFY ( column-name, ... ) VALUES ( expression, ... ) ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ], ... ]
[ OPTION( query-hint, ... ) ]
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| FORCE NO OPTIMIZATION
```

```
| option-name = option-value
```

```
option-name : identifier
```

```
option-value :
hostvar （許可されたインジケータ）
| string
| identifier
| number
```

特定の SQL Remote 機能の実装

```
UPDATE table-name
PUBLICATION publication-name
{ SUBSCRIBE BY subscription-expression |
   OLD SUBSCRIBE BY old-subscription-expression
   NEW SUBSCRIBE BY new-subscription-expression }
WHERE search-condition
```

```
expression : value | subquery
```

パラメータ

構文 - 特定の SQL Remote 機能の実装

table-name

table-name は、修正する必要がある、リモートデータベースのテーブルです。

publication-name

publication-name は、サブスクリプションを変更する必要のあるパブリケーションです。

subscription-expression

subscription-expression の値は、ローの新しい受信者と既存の受信者を決定するため、SQL Remote によって使用されます。subscription-expression は、値またはサブクエリのいずれかです。または、OLD と NEW の両方のサブスクリプション式を指定できます。

WHERE clause

WHERE 句は、サブスクリプションを作成したデータベース間で移動させるローを指定します。

備考

構文 - Message Agent によって実行される単一テーブルの単一ローの更新

SQL Remote Message Agent は、メッセージをデータベースに適用するときにこの構文を使用します。標準的な UPDATE 文で単一のテーブルに対するローの更新を行うパラメータやコメントはすべて、この変形版の UPDATE 文でも適用されます。

VERIFY 句には、更新するローの中にあると予想される値のセットを含めます。値が一致しない場合、UPDATE が処理される前に RESOLVE UPDATE トリガが起動されます。VERIFY 句の照合が失敗しても、UPDATE は失敗しません。VERIFY 句を指定すると、テーブルが一度に 1 つずつ更新されます。

構文 - 特定の SQL Remote 機能の実装

この構文は、リモートデータベースにあるデータの分割に影響を与える 1 つのテーブル内のローを変更する場合に使用します。

UPDATE `table-name` 構文を使うと、トランザクションログ内にエントリが作成されますが、データベーステーブルは変更されません。

UPDATE `table-name` 構文で OLD SUBSCRIBE BY 式と NEW SUBSCRIBE BY 式を付けない場合は BEFORE トリガで使用する必要があります。

UPDATE `table-name` 構文で OLD SUBSCRIBE BY 式と NEW SUBSCRIBE BY 式を付ける場合はどこでも使用できます。

構文 3 に OLD 式も NEW 式も使用しない場合は、関連する値にアクセスできるように、BEFORE トリガ内で使用します。この目的は、リストが変更されるたびに、SUBSCRIBE BY 値の完全なリストを提供することです。これは、SQL Remote トリガに配置され、データベースサーバが SUBSCRIBE BY 値の現在のリストを計算できるようにします。両方のリストがトランザクションログ内に記録されます。

Message Agent は 2 つのリストを使用して、ローが必要な、ローを保有しない任意のリモートデータベースにローが移動したことを確認します。また、Message Agent は、ローを保有し、そのローが不要になったリモートデータベースからローを削除します。そのローを保有し、依然そのローを必要とするリモートデータベースは、UPDATE 文の影響を受けません。

UPDATE `table-name` 構文を使用すると、古い SUBSCRIBE BY リストと新しい SUBSCRIBE BY リストを明示的に指定して、SQL Remote トリガの効率を高めることができます。このリストがないと、データベースサーバはパブリケーション定義から古い SUBSCRIBE BY リストを計算します。通常は、新しい SUBSCRIBE BY リストは古い SUBSCRIBE BY リストとほんのわずか異なるだけなので、古いリストの処理は 2 回行われます。古いリストと新しいリストの両方を指定すると、この無駄な作業を避けることができます。

OLD SUBSCRIBE BY 構文と NEW SUBSCRIBE BY 構文は、同じ SUBSCRIBE BY 式で同じトリガを使用し、たくさんのテーブルを更新する場合には特に便利です。これにより、パフォーマンスが大幅に向上します。

SUBSCRIBE BY 式は、値またはサブクエリのいずれかです。

SUBSCRIBE BY 句内のサブクエリを使って作成したパブリケーションについては、UPDATE `table-name` 構文を含むトリガを作成してローがそれぞれの正しいサブスクリプションに格納されていることを確認します。

UPDATE `table-name` 構文を使うと、トランザクションログ内にエントリが作成されますが、データベーステーブルは変更されません。

権限

更新されるテーブルの所有者であるか、変更されるカラムに対する UPDATE 権限を持っているか、または UPDATE ANY TABLE システム権限を持っていることが必要です。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、SQL Remote Message Agent 用の構文を使用して、従業員 Philip Chin (employeeID = 129) を販売部からマーケティング部に異動します。

```
UPDATE GROUPO.Employees
SET DepartmentID = 400
VERIFY ( DepartmentID )
VALUES ( 200 )
WHERE EmployeeID = 129;
```

関連情報

UPDATE 文 [1391 ページ]

1.4.4.286 UPDATE 文

データベーステーブルにあるローを変更します。

構文

ジョイン、ビュー、および派生テーブルを使用したローの更新

```
UPDATE [ row-limitation ] table-expression [, ...]
SET set-item[, ...]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ] , ...]
[ OPTION( query-hint, ... ) ]
```

```
row-limitation :
FIRST
| TOP { ALL | limit-expression } [ START AT startat-expression ]
limit-expression : simple-expression
startat-expression : simple-expression
```

```
simple-expression :
```

```
integer
| variable
| ( simple-expression )
| ( simple-expression { + | - | * } simple-expression )
```

`table-expression` :ジョイン、外部ジョイン、ビュー、派生テーブルを含むことのできるテーブル式。`FORM` 句の説明を参照。

```
set-item :
[ correlation-name.]column-name={ expression | DEFAULT }
| [ owner. ]table-name.column-name = { expression | DEFAULT }
| [ owner. ]@variable-name=expression
```

```
table-name :
[ owner.]base-table-name
| temporary-table-name
| derived-table-name
| [ owner.]view-name
```

```
query-hint :
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| FORCE NO OPTIMIZATION
| option-name = option-value
```

```
option-name : identifier
```

```
option-value :
hostvar (許可されたインジケータ)
| string
| identifier
| number
```

単一テーブルのローの更新

```
UPDATE [ row-limitation ] table-name
SET set-item[, ...]
FROM table-expression [, ...] ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ] , ...]
[ OPTION( query-hint, ... ) ]
```

```
table-name :
[ owner.]table-name [ [ AS ] correlation-name ]
| [ owner.]view-name [ [ AS ] correlation-name ]
| derived-table
```

```
derived-table :
( select-statement )
[ AS ] correlation-name [ ( column-name [, ...] ) ]
```

パラメータ

UPDATE clause

テーブルの更新では、table-expression にテンポラリテーブル、グローバルテンポラリテーブル、派生テーブル、またはビューを含めることができます。ビューと派生テーブルは、更新可能でないものを除き、更新することができます。

ジョイン、ビュー、および派生テーブルを使用して更新する場合、複数の table-expression のリストが、基本となるテーブル式でフォーマットされたローの直積になります。これは、WHERE 句を使用して制限することができます。

ビューを定義しているクエリ指定が更新可能な場合にかぎり、ビューに対して更新操作を実行できます。

row-limitation clause

ローを制限する句を使用することによって、アップデートされるローが WHERE 句を満たすローのサブセットのみに制限されます。TOP 引数と START AT 引数には、ホスト変数、整数定数、または整数変数を使用した簡単な算術演算を指定できます。TOP 引数は 0 以上にします。START AT 引数は 0 より大きい値にします。ローの順序を意味のあるものにするために ORDER BY 句も指定します。

SET clause

カラム名または変数を指定した expression に設定する場合は、SET 句を使用します。

SET 句は、このフォーマットを使用するカラムを計算カラム値に設定するときに使用します。

```
SET column-name = expression, ...
```

指定された各カラムは、expression の値に設定されます。expression には制限がありません。expression が column-name である場合は、そのカラムの古い値が使われます。

カラムにデフォルトが定義されている場合は、SET 句を使用してカラムをデフォルト値に設定してください。

また、SET 句で次のフォーマットを使用して変数に代入することもできます。

```
SET @variable-name = expression, ...
```

owner は、データベーススコープ変数でのみ指定できます。

変数に値を代入する場合、その変数がすでに宣言済みであることが必要です。また、変数名はアット記号 (@) で始めます。変数名が更新されるテーブルのカラム名と一致している場合、UPDATE 文はカラム値を更新し、変数は変更しないでそのままにします。変数の代入とカラムの代入は任意の順序で混在させることができます。

FROM clause

オプションの FROM table-expression 句で、ジョインに基づいてテーブルを更新できます。table-expression には、OUTER、CROSS、および NATURAL ジョインなどの任意の複雑なテーブル式を指定できます。

FROM 句が存在する場合、table-name では更新される単一のテーブルを指定します。この名前は FROM 句と同じように修飾されている必要があります。FROM 句で相関名が使用されている場合、同一の相関名を table-name として指定します。更新されるテーブル式が派生テーブルの場合、派生テーブルを table-name の指定で繰り返す必要があります。

ansi_update_constraints オプションが Strict に設定されている場合、指定したテーブルは更新できません。

FROM 句が指定されている場合、SET 句では更新される table-name のカラムのみを指定できます。それ以外の場合は、エラーが発生します。

次の文は、相関名を使用した特定のテーブルを更新する構文で、この構文を使用した UPDATE 文のテーブル名に、潜在的なあいまいさが存在することを示しています。

```
UPDATE table_1
SET column_1 = ...
FROM table_1 AS alias_1, table_1 AS alias_2
```

```
WHERE ...
```

次の例では、FROM 句の table_1 の各インスタンスには相関名があり、table_1 のそれ自体へのセルフジョインを示しています。ただし、この UPDATE 文はセルフジョインを構成するローのうち、どのローを更新対象とするかの指定に失敗します。この省略により、次のように、UPDATE 文の相関名を指定することで修正できます。

```
UPDATE table_1 as alias_1
SET column_1 = ...
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

WHERE clause

WHERE 句を指定すると、探索条件を満たすローだけが更新されます。WHERE 句を指定しない場合、すべてのローが更新されます。

ORDER BY clause

通常、ローを更新する順序は重要ではありません。ただし、FIRST 句または TOP 句と一緒に使用する場合は、順序が重要です。

ORDER BY 句ではカラムの順序数を使用できません。

ORDER BY 句を使用するときは、ansi_update_constraints オプションを Strict に設定しないでください。

ORDER BY 句に含まれるカラムを更新するには、ansi_update_constraints オプションを Off に設定します。

OPTION clause

この句は、文を実行するためのヒントを指定するときに使用します。指定する設定は、現在の文にのみ適用され、ODBC 実行可能アプリケーションによる設定など、パブリックオプションまたはテンポラリオプションの設定よりも優先されます。次のヒントがサポートされます。

- MATERIALIZED VIEW OPTIMIZATION `option-value`
- FORCE OPTIMIZATION
- FORCE NO OPTIMIZATION
- `option-name` = `option-value`.
  クエリテキスト内の `OPTION( isolation_level = ... )` の指定を使用すると、クエリの独立性レベルを指定する他のいずれの手段よりも優先されます。
  クエリテキスト内の `OPTION( parameterization_level = ... )` の指定を使用すると、クエリのパラメータ化レベルよりも優先されます。

備考

UPDATE 文を使って、1 つまたは複数のテーブルのローを修正します。指定した各カラムに、等号の右側の式の値を設定します。`expression` には制限がありません。`column-name` も式の中で使用できます。この場合、古い値を使用します。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字の状態のままで格納されます。文字列 Street で更新した CHAR データ型カラムは、常に大文字の S と残りの文字が小文字でデータベースに格納されます。SELECT 文は、文字列を Street として返します。ただし、データベースで大文字と小文字が区別されない場合は、すべての比較において Street は street、STREET などと同じと見なされます。さらに、単一カラムのプライマリキーにエントリ Street がある場合、プライマリキーがユニークでなくなるため、別のローのプライマリキーの street への UPDATE は拒否されます。

新しい値が元の値と同じ場合、データは変更されません。ただし、BEFORE UPDATE トリガは、ローを対象に UPDATE が実行されるたび、新しい値が古い値と異なるかどうかに関係なく起動されます。一方、AFTER UPDATE トリガは、新しい値が古い値と異なる場合にのみ起動されます。

UPDATE 文を使用してデータを大量に更新する場合も、カラム統計は更新されます。

WHERE 句を指定しない場合、すべてのローが更新されます。WHERE 句を指定する場合、探索条件を満たすローだけが更新されます。

通常、ローを更新する順序は問題ではありません。ただし、NUMBER(*) 関数とともに使用して、指定された順序でローの中の数字を増加させる場合に、順序付けが役に立ちます。また、テーブルのプライマリキー値に 1 を追加するような操作を行う場合は、操作中にプライマリキーの重複が起こらないように、プライマリキーの降順でその操作を行う必要があります。

ビューを定義する SELECT 文の中に GROUP BY 句や集合関数がないか、または UNION 句を伴わない場合は、ビューを更新できます。

オプションの FROM 句で、ジョインに基づいてテーブルを更新できます。FROM 句がある場合、WHERE 句は FROM 句のローが条件を満たしているかどうかを調べます。FROM 句を使用する場合、更新されるテーブル名をその文の両方の部分で同じように修飾することが重要です。一方の場所で相関名が使用されている場合は、もう一方の場所でも同じ相関名を使用します。それ以外の場合は、エラーが発生します。

SET 句を使用してデータベーススコープ変数を更新する場合、データベースの再起動間で変数は保持されますが設定は保持されません。データベースが再起動すると、データベーススコープ変数の値は NULL またはデフォルト (定義されている場合) に戻されます。SYSDATABASEVARIABLE システムビューには、すべてのデータベーススコープ変数のリストとそれらのデフォルト値が含まれます。

別のユーザが所有するデータベーススコープ変数は更新できません。

権限

更新されるテーブルの所有者であるか、変更されるカラムに対する UPDATE 権限を持っているか、または UPDATE ANY TABLE システム権限を持っていることが必要です。

自分が所有するデータベーススコープ変数の更新には、権限は不要です。PUBLIC が所有するデータベーススコープ変数の更新には、UPDATE PUBLIC DATABASE VARIABLE システム権限が必要です。

関連する動作

カラム統計は変更された値を反映するように更新されます。

テーブルにプライマリキー、UNIQUE 制約、または UNIQUE インデックスがある場合、一意性制約に違反しなければテーブル操作を実行できないときには、UPDATE 文の処理にテンポラリテーブルを使用する必要がある場合があります。テンポラリテーブルには、1 つ以上の一意性制約に違反している UPDATE 文で変更されたローが格納されます。これらのローは、UPDATE 文の実行時に一時的にベーステーブルから削除され、後で再挿入されます。この動作は、AFTER トリガと他の同時接続に影響することがあります。

標準

ANSI/ISO SQL 標準

UPDATE...table-expression 構文は、ANSI/ISO SQL 標準のコア機能です。ただし、次の拡張は標準にありません。

- FROM 句と ORDER BY 句。
- row-limitation 句。
- 複数の table-expression を指定する機能。
- SET 句を使用して変数を更新する機能。
- OPTION 句。

UPDATE...table-expression 構文には、以下の 2 つのオプションの ANSI/ISO SQL 言語機能も含まれます。

- 1 つ以上の派生テーブルを含む可能性のあるジョインの更新に対するサポート。これは、オプションの ANSI/ISO SQL 言語機能 T111、"Updatable joins, unions, and columns" の一部を構成します。
- UPDATE 文の検索条件部分を形成するネストされたサブクエリで参照されるテーブルの変更に対するサポート。これは、オプションの ANSI/ISO SQL 言語機能 F781 "Self-referencing operations" を構成します。

UPDATE...table-expression 構文では、ansi_update_constraints オプションの設定によって、変更できるテーブル式の形式が制御されます。ANSI/ISO コア機能との互換性を確保するために、ansi_update_constraints オプションは必ず Strict に設定してください。

UPDATE...table-name 構文は標準にありません。

例

次の例では、サンプルデータベースを使用して、従業員 Philip Chin (employeeID = 129) を販売部からマーケティング部に異動する例を示します。

```
UPDATE GROUPO.Employees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

次の例では、サンプルデータベースを使用して、既存するすべての受注の ID から 2000 を引いて番号をふり直します。

```
UPDATE GROUPO.SalesOrders AS orders
SET orders.ID = orders.ID - 2000
ORDER BY orders.ID ASC;
```

この更新が可能となるのは、SalesOrderItems テーブル (プライマリキー SalesOrders.ID を参照する) の外部キーがアクション ON UPDATE CASCADE を使用して定義されている場合だけです。SalesOrderItems テーブルも更新されます。文で ORDER BY 句が指定されており、SET 句でも順序属性が指定されているため、ansi_update_constraints オプションを Off に設定しないとエラーが返されます。

この例は、サンプルデータベースを使用し、データベースの現在の独立性レベル設定ではなく、独立性レベル 2 で製品の価格を変更します。

```
UPDATE GROUPO.Products
SET UnitPrice = 7.00
WHERE ID = 501
OPTION( isolation_level = 2 );
```

この例では、ansi_update_constraints オプションを Strict 以外の値に設定する必要があります。この例では再度サンプルデータベースを使用し、UPDATE...table-name 構文を使用して、現在の在庫を上回る量の注文が 1 つ以上存在する Tee Shirt の在庫をリセットします。

```
UPDATE GROUPO.Products AS a
SET Quantity = 0
FROM GROUPO.Products a JOIN GROUPO.SalesOrderItems b ON a.ID = b.ProductID
WHERE a.Name = 'Tee Shirt' AND b.Quantity > a.Quantity;
```

この例では、ansi_update_constraints オプションを Strict 以外の値に設定する必要があります。この例では、UPDATE...table-expression 構文を使用して、これらの Tee Shirts の在庫をリセットし、Tee Shirt の注文の ShipDate を今日の日付にリセットします。

```
UPDATE GROUPO.Products a JOIN GROUPO.SalesOrderItems b on a.ID = b.ProductID
SET a.Quantity = 0, b.ShipDate = CAST( NOW() AS DATE)
WHERE a.Name = 'Tee Shirt' AND b.Quantity > a.Quantity
```

この例では、テーブルを更新して、カラムをデフォルト値に設定する方法を示します。この例では、テーブル MyTable を作成し、このテーブルにデータを移植します。次に、SET 句を指定して UPDATE 文を実行し、いくつかのカラム値を更新します。

```
CREATE OR REPLACE TABLE MyTable(
   PK INT PRIMARY KEY  DEFAULT AUTOINCREMENT,
   TableName CHAR(128) NOT NULL,
   TableNameLen INT DEFAULT 20,
   LastUser CHAR(10) DEFAULT LAST USER,
   LastTime TIMESTAMP DEFAULT TIMESTAMP,
   LastTimestamp TIMESTAMP DEFAULT CURRENT TIMESTAMP );
INSERT INTO MyTable WITH AUTO NAME
   SELECT
      LENGTH(t.table_name) AS TableNameLen,
      t.table_name AS TableName
   FROM SYS.SYSTAB t
   WHERE table_id <= 10;
WAITFOR DELAY '00:00:05';
UPDATE MyTable
   SET TableName = TableName || '*',
       LastTimestamp = DEFAULT
   WHERE TableName LIKE '%idx%';
SELECT * FROM MyTable;
```

この例では、LastTime カラムが自動的に更新されます。このカラムのデフォルトが TIMESTAMP であるためです。LastTimestamp カラムはデフォルトが CURRENT TIMESTAMP であるため、普通は更新されません。SET 句によって、デフォルトを使用したカラムの値の更新が行われています。

関連情報

FROM 句 [1112 ページ]
INSERT 文 [1167 ページ]
UPDATE 文 [SQL Remote] [1388 ページ]
UPDATE (位置付け) 文 [ESQL] [SP] [1386 ページ]

1.4.4.287 VALIDATE LDAP SERVER 文

LDAP サーバ設定オブジェクトを検証します。

構文

```
VALIDATE LDAP SERVER { ldapua-server-name | ldapua-server-attribs }
[ CHECK user-id [ user-dn-string ] ]
```

パラメータ

ldapua-server-name

検証する LDAP サーバ構成オブジェクトの名前。この句の詳細については、CREATE LDAP SERVER 文を参照してください。

ldapua-server-attribs

`ldapua-server-attribs` を使用して LDAP サーバ構成オブジェクトを検証する場合は、指定された属性が検証されます。URL が解析されて構文エラーが識別されます。構文エラーが発生した場合は、検証が停止してエラーが返されます。

この句の詳細については、CREATE LDAP SERVER 文を参照してください。

CHECK clause

LDAP サーバで検索するユーザ ID を指定する場合は、この句を使用します。

備考

VALIDATE LDAP SERVER 文が実行されると、LDAP サーバへの接続が試行されます。ACCESS ACCOUNT とパスワードが指定されている場合、その値を使用して SEARCH DN URL への接続を確立して、SEARCH DN URL、ACCESS ACCOUNT、ACCESS ACCOUNT のパスワードを検証します。

LDAP ユーザ認証を使用して新しいサーバを設定する場合、この文は LDAP サーバ構成の変更をその適用前に検証し、データベースサーバと LDAP サーバ間の問題を診断するのに役立ちます。

プロシージャの定義は SYSPROCEDURE システムビューに表示されるため、この文をプロシージャ内で使用し、属性に IDENTIFIED BY 句を含める場合は、文字列リテラルとしてパスワードを指定しないでください。セキュリティ保護のため、プロシージャ定義の外部で宣言される変数を使用してパスワードを指定してください。

権限

MANAGE ANY LDAP SERVER システム権限が必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、LDAP サーバ構成オブジェクトを作成して、ホスト名 voyager とポート番号 389 で、apps_primary で指定された ACCESS ACCOUNT とパスワードを使用して LDAP サーバに接続します。

```
CREATE LDAP SERVER apps_primary2
    SEARCH DN
        URL  'ldap://voyager:389/dc=MyCompany,dc=com??sub?cn=*'
        ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
        IDENTIFIED BY 'Secret99Password'
    AUTHENTICATION URL 'ldap://voyager:389/'
    CONNECTION TIMEOUT 3000
    WITH ACTIVATE;
VALIDATE LDAP SERVER apps_primary2;
```

次の例では、ホスト名 voyager とポート番号 389 で、apps_primary2 で指定された ACCESS ACCOUNT とパスワードを使用して LDAP サーバに接続します。ユーザ ID myusername が有効で、想定されるユーザの識別名に一致していることもチェックします。

```
VALIDATE LDAP SERVER apps_primary2
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com';
```

LDAP サーバ構成オブジェクトが定義されていない場合は、指定されている属性によって同じチェックが実行されます。

```
VALIDATE LDAP SERVER
SEARCH DN
URL 'ldap://voyager:389/dc=MyCompany,dc=com??sub?cn=*'
ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldap://voyager:389/'
CONNECTION TIMEOUT 3000
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com';
```

関連情報

CREATE LDAP SERVER 文 [847 ページ]
ALTER LDAP SERVER 文 [654 ページ]
DROP LDAP SERVER 文 [1042 ページ]

1.4.4.288 VALIDATE 文

現在のデータベース、または現在のデータベース内にある単一または複数のテーブル、マテリアライズドビュー、またはインデックスを検証します。

構文

データベースの検証

```
VALIDATE { CHECKSUM | DATABASE }
```

テーブルまたはマテリアライズドビューの検証

```
VALIDATE {
TABLE [ owner.]table-name
| MATERIALIZED VIEW [ owner.]materialized-view-name }
[ WITH EXPRESS CHECK ]
[ WITH DATA LOCK | WITH SNAPSHOT ]
```

インデックスの検証

```
 VALIDATE {
INDEX index-name
| [ INDEX ] FOREIGN KEY role-name
| [ INDEX ] PRIMARY KEY }
[ WITH DATA LOCK | WITH SNAPSHOT ]
ON [ owner.]object-name
```

```
object-name :
table-name
| materialized-view-name
```

テキストインデックスの検証

```
VALIDATE TEXT INDEX index-name
ON [ owner.]table-name
```

パラメータ

CHECKSUM clause

この句は、データベース上で各ページのチェックサムを検証するときに使用します。データベースでチェックサムが有効になっている必要があります。CHECKSUM 句によって、データベースページがディスク上で変更されていないことが確認されます。

チェックサムを有効にしてデータベースを作成すると、各データベースページがディスクに書き込まれる前に、そのページのチェックサムが計算されます。CHECKSUM は、各データベースページを (データベースサーバのキャッシュからではなく) ディスクから直接読み込み、各ページのチェックサムを計算します。

ページに対して計算されたチェックサムが、そのページについて格納されているチェックサムと一致しない場合は、エラーが発生し、無効なページに関する情報がデータベースサーバメッセージウィンドウに表示されます。

DATABASE clause

VALIDATE DATABASE 文では、フリーマップでページが割り当て済みかフリーかが正しく識別され、孤立している BLOB がないことを確認できます。VALIDATE DATABASE は、チェックサム検証を実行して、各データベースページが正しいオブジェクトに所属することを検証します。たとえば、テーブルページ上でテーブル ID によって、定義がテーブルページのセットに最新のテーブルを含む有効なテーブルが識別されなければなりません。

VALIDATE DATABASE 文はページをサーバのキャッシュに順番に転送します。データベースサーバは、キャッシュに転送されたページの内容とチェックサムを常に検証します。データベースクリーナーが実行している間にデータベースの検証を開始すると、データベースクリーナーの実行が終了するまで検証は実行されません。

[INDEX] PRIMARY KEY | FOREIGN KEY

VALIDATE INDEX 文は VALIDATE TABLE 文と同じ操作を実行しますが、指定されたインデックスのみ検証して、基になるテーブルやその他のインデックスは検証しない点が異なります。

外部キーインデックスの場合、WITH EXPRESS CHECK 句が指定されていなければ、プライマリキーテーブルの各値が検索されて、参照整合性制約がそのまま残っていることが確認されます。指定されたインデックスが外部キーインデックスでない場合、WITH EXPRESS CHECK は効果がありません。

MATERIALIZED VIEW

指定されたマテリアライズドビューを検証します。

TABLE

VALIDATE TABLE 文は、ベーステーブル内の一連のローと値が各インデックスに含まれる一連のローと値に一致するかどうかを確認して、指定されたテーブルとそのすべてのインデックスを検証します。

また、すべてのテーブルの BLOB を移動して、BLOB のアロケーションマップを検証し、孤立した BLOB を検出します。また、VALIDATE TABLE 文は、すべてのテーブルのインデックスページの物理構造を確認し、インデックスハッシュ値の順序、インデックスの一意性要件も (指定されている場合は) 検査します。

VALIDATE TABLE 文は、VALIDATE DATABASE と同じようにデータベースサーバのキャッシュを使用するため、データベースサーバもテーブルとそのインデックスで使用されるすべてのページのチェックサムと基本的な妥当性を検査します。

TEXT INDEX

VALIDATE TEXT INDEX 文は、インデックス内の単語の位置情報が正常であることを確認できます。位置情報が維持されていない場合は、エラーが生成され、テキストインデックスを再作成する必要があります。テキストインデックスが自動または手動のいずれかの場合、REFRESH TEXT INDEX 文を実行して、テキストインデックスを再作成できます。

生成されたエラーが即時テキストインデックスに関わる場合、即時インデックスを削除して、新しいインデックスを作成します。

WITH EXPRESS CHECK

WITH EXPRESS CHECK 句を指定すると、参照整合性検査が無効になるため、パフォーマンスが大幅に向上する場合があります。

WITH DATA LOCK | WITH SNAPSHOT

アクティブなトランザクションを持つテーブルを検証する場合は、破損したテーブルに関する誤ったエラーを受信することを避けるため、次のいずれかのオプションを選択します。

WITH DATA LOCK

指定したテーブルに排他データロックを適用することで、トランザクションによってテーブルスキーマまたはデータが修正されることを回避します。同時実行トランザクションは、テーブルデータまたはスキーマを読み込むことができますが、修正することはできません。

FOREIGN KEY 句を指定すると、排他データロックはプライマリキーテーブルにも適用されます。

WITH SNAPSHOT

スナップショットアイソレーションを適用して、コミットされたデータのみがチェックされるようにします。トランザクションは、データを読み込むことも修正することもできます。この句を使用するには、データベースで (allow_snapshot_isolation データベースオプションを使用して) スナップショットアイソレーションが有効になっている必要があります。この句ではスナップショットアイソレーションを使用するため、パフォーマンスに影響することがよくあります。

備考

警告

WITH DATA LOCK または WITH SNAPSHOT を指定しない場合、データベースを変更している接続がない状態のときに検証を実行してください。それ以外の場合は、データベースの何らかの破損を示す誤ったエラーが報告される場合があります。

権限

VALIDATE ANY OBJECT システム権限が必要です。

関連する動作

WITH DATA LOCK 句を指定すると、指定したテーブルおよびビューに排他データロックが適用されます。

WITH DATA LOCK 句とともに FOREIGN KEY 句を指定すると、排他データロックはプライマリキーテーブルにも適用されます。

WITH DATA LOCK 句および WITH SNAPSHOT 句のオートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例では、Products テーブルを検証します。

```
VALIDATE TABLE GROUPO.Products;
```

関連情報

REFRESH TEXT INDEX 文 [1255 ページ]
DROP TEXT INDEX 文 [1077 ページ]
CREATE TEXT INDEX 文 [976 ページ]
sa_validate システムプロシージャ [1653 ページ]
sa_clean_database システムプロシージャ [1455 ページ]
CREATE DATABASE 文 [781 ページ]
CREATE INDEX 文 [842 ページ]

1.4.4.289 WAITFOR 文

指定された時間の間、または指定の時間になるまで現在の接続処理を遅らせます。

構文

```
WAITFOR {
DELAY time
| TIME time }
[ CHECK EVERY integer ]
[ AFTER MESSAGE BREAK ]
```

```
time : string
```

パラメータ

DELAY clause

DELAY を使用すると、処理は特定の期間だけ中断されます。

TIME clause

TIME を指定すると、データベースサーバ時刻が指定の時刻に達するまで、処理が中断されます。現在のサーバ時刻が指定の時刻より後の場合は、翌日のその時刻になるまで処理が中断されます。

CHECK EVERY clause

このオプション句は、WAITFOR 文が起動する頻度を制御します。デフォルトでは、5 秒ごとに起動します。値はミリ秒単位であり、最小値は 250 ミリ秒です。

AFTER MESSAGE BREAK clause

WAITFOR 文を使用して、別の接続からのメッセージを待つことができます。メッセージが受信されると、通常このメッセージは WAITFOR 文を実行したアプリケーションに転送され、WAITFOR 文は待機を継続します。AFTER MESSAGE BREAK 句が指定されている場合、別の接続からのメッセージが受信されると、WAITFOR 文が完了します。メッセージテキストはアプリケーションに転送されませんが、MessageReceived 接続プロパティの値を取得してアクセスできます。

備考

WAITFOR 文は、定期的 (デフォルトでは 5 秒おき) に起動して、キャンセルされたかどうか、またはメッセージが受信されたかどうかをチェックします。どちらも発生していない場合、文は待機を継続します。

スケジュールされたイベントはそれ専用の接続で実行されるため、通常は WAITFOR TIME よりスケジュールされたイベントを使用することをお奨めします。

権限

なし

関連する動作

WAITFOR 文の実装によって、文を処理しているワーカーは、文が待機している間ブロックされます。これによって、ワーカプール内の使用可能なワーカーの数が減少します。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の例は 3 秒だけ待機します。

```
WAITFOR DELAY '00:00:03';
```

次の例は 0.5 秒 (500 ミリ秒) だけ待機します。

```
WAITFOR DELAY '00:00:00.500';
```

次の例は午後 8 時まで待機します。

```
WAITFOR TIME '20:00';
```

次の例では、接続 1 の WAITFOR 文は、接続 2 からメッセージを受信すると完了します。

```
// connection 1:
BEGIN
  DECLARE msg LONG VARCHAR;
  LOOP  // forever
    WAITFOR DELAY '00:05:00' AFTER MESSAGE BREAK;
    SET msg = CONNECTION_PROPERTY('MessageReceived');
    IF msg != '' THEN
```

```
      MESSAGE 'Msg: ' || msg TO CONSOLE;
    END IF;
  END LOOP
END;
// connection 2:
MESSAGE 'here it is' FOR connection 1
```

関連情報

CREATE EVENT 文 [805 ページ]
MESSAGE 文 [1211 ページ]

1.4.4.290 WAITFOR SEMAPHORE 文

セフォマに関連付けられたカウンタを減分します。

構文

```
WAITFOR SEMAPHORE [ owner.]semaphore-name
[ TIMEOUT number-milliseconds ]
```

パラメータ

owner

セマフォの所有者。owner は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

semaphore-name

セマフォの名前。semaphore-name は、間接識別子 (`[@variable-name]` など) を使用することでも指定できます。

TIMEOUT clause

セフォマに関連付けられたカウンタの減分を待機する期間を、ミリ秒単位で指定します。この句を指定しない場合、接続はカウントを減分できるようになるか、エラーが返されるまで無制限に待機します。

number-milliseconds の指定には、変数 (TIMEOUT @timeout-value など) を使用できます。number-milliseconds に変数が設定され、その変数が NULL の場合、句を指定しないのと同じ動作になります。

備考

WAITFOR SEMAPHORE 文は、セマフォに関連付けられたカウンタを減分します。カウンタが正の整数の場合、カウントが減分されて文が完了します。カウンタが 0 の場合、接続はカウンタが正の整数になるか、TIMEOUT によって指定された期間が経過した時点でタイムアウトを示すエラーが返されるまで待機します。

セマフォの待機中のデッドロック検出で現在の接続が識別されると、エラーが返されます。セマフォが削除された場合もエラーが返されます。

セフォマを通知した接続が削除またはキャンセルされてもカウンタの減分は存続するため、ご使用のアプリケーションでこの状況に対処する必要があります。

権限

UPDATE ANY MUTEX SEMAPHORE システム権限を持っているか、セマフォの所有者である必要があります。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文では、license_counter セフォマのカウンタを 1 減分します。セマフォカウントが 0 の場合、文はカウンタが増分されるまで無制限に待機します。

```
WAITFOR SEMAPHORE license_counter;
```

関連情報

DROP SEMAPHORE 文 [1059 ページ]
NOTIFY SEMAPHORE 文 [1215 ページ]
CREATE SEMAPHORE 文 [909 ページ]
SYSMUTEXSEMAPHORE システムビュー [1819 ページ]

1.4.4.291 WHENEVER 文 [ESQL]

Embedded SQL プログラム内でのエラー処理を指定します。

構文

```
WHENEVER { SQLERROR | SQLWARNING | NOTFOUND }
GOTO
    label
    | STOP
    | CONTINUE
    | { C-code; }
```

```
label : identifier
```

備考

WHENEVER 文を使って、SQL 文を処理するときにデータベースに発生するエラー、警告、例外条件をトラップします。Embedded SQL C プログラム内のどこにでもこの文を入れることができ、この文はコードを生成しません。プリプロセッサは、それぞれの継続する SQL 文の後にコードを生成します。すべての Embedded SQL 文では、WHENEVER 文のソース行から同じエラー条件の次の WHENEVER 文、またはソースファイルの最後まで、エラーアクションは有効です。

注記

エラー条件は、文がいつ実行されたかではなく、C 言語ソースファイル内での位置に基づいて有効になります。

デフォルトアクションは CONTINUE です。

この文は、単純なプログラムの中で便利に使用できます。たいていの場合、SQLCA (SQLCODE) の sqlcode フィールドを直接チェックするのがエラー条件をチェックする最も簡単な方法です。この場合、WHENEVER 文を使用しません。WHENEVER 文が実行されると、プリプロセッサはそれぞれの文の後に `if (SQLCODE)` テストを生成します。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

WHENEVER 文で実行される例外条件宣言は、コア機能です。標準では、キーワード SQLERROR ではなく SQLEXCEPTION を使用します。WHENEVER 文に単なる文ラベルではなく直接 C コードを含める機能は、標準にありません。STOP アクションも標準にありません。

例

```
EXEC SQL WHENEVER NOTFOUND GOTO done;
EXEC SQL WHENEVER SQLERROR
   {
      PrintError( &sqlca );
      return( FALSE );
   };
```

1.4.4.292 WHILE 文 [T-SQL]

文または複合文を繰り返し実行します。

構文

```
WHILE search-condition statement
```

備考

WHILE は、単一の SQL 文とキーワード BEGIN と END で囲まれた複合文を制御します。

複合文中での文の実行は、BREAK 文と CONTINUE 文で制御できます。BREAK 文はループを終了し、ループの最後を示す END の後から実行が再開されます。CONTINUE 文は、その後の文をすべて省略して WHILE ループを再開します。

権限

なし。

関連する動作

なし。

標準

ANSI/ISO SQL 標準

WHILE 文は、オプションの ANSI/ISO SQL 言語機能 P002、"Computational completeness" の一部です。WHILE 文の Transact-SQL 変形には、END WHILE は含まれていません。

例

次のコードは、WHILE の使い方を示します。

```
WHILE ( SELECT AVG(UnitPrice) FROM GROUPO.Products ) < $30
BEGIN
   UPDATE GROUPO.Products
   SET UnitPrice = UnitPrice + 2
   IF ( SELECT MAX(UnitPrice) FROM GROUPO.Products ) > $50
      BREAK
END
```

BREAK 文は、最も高い製品の価格が 50 ドルを超える場合、WHILE ループをブレークします。そうでない場合、ループは平均価格が 30 ドル以上になるまで継続します。

関連情報

LOOP 文 [1203 ページ]
CONTINUE 文 [778 ページ]

1.4.4.293 WINDOW 句

SELECT 文の AVG や RANK などの Window 関数を使用するウィンドウのすべてまたは一部を定義します。

構文

```
WINDOW window-expression, ...
```

```
window-expression : new-window-name AS ( window-spec )
```

```
window-spec :
[ existing-window-name ]
[ PARTITION BY expression, ... ]
[ ORDER BY expression [ ASC | DESC ], ... ]
[ { ROWS | RANGE } { window-frame-start | window-frame-between } ]
```

```
window-frame-start :
{ UNBOUNDED PRECEDING
 | unsigned-integer PRECEDING
 | CURRENT ROW }
```

```
window-frame-between :
BETWEEN window-frame-bound1 AND window-frame-bound2
```

```
window-frame-bound :
window-frame-start
| UNBOUNDED FOLLOWING
| unsigned-integer FOLLOWING
```

パラメータ

PARTITION BY clause

PARTITION BY 句は、指定した式のユニークな値に基づいて、結果セットを論理グループに構築します。Window 関数を使用してこの句を使用する場合、関数は独立して各パーティションに適用されます。たとえば、カラム名が指定された PARTITION BY に従う場合、結果セットはカラムの個別の値によってパーティション化されます。

この句を省略すると、全体の結果セットはパーティションとして扱われます。

PARTITION BY `expression` には、整数リテラルを使用できません。

ORDER BY clause

ORDER BY 句は、結果セットの各パーティションのローをソートする方法を定義します。さらに、昇順の場合は ASC (デフォルト)、降順の場合は DESC を指定して順序を制御することができます。

ORDER BY `expression` には、整数リテラルを使用できません。

この句が省略されると、データベースサーバは最も効率的な順序でローを返します。そのため、ローに最後にアクセスした日時によって、結果セットでの表示順序が異なることがあります。

ROWS clause and RANGE clause

ROWS 句または RANGE 句を使用して、ウィンドウのサイズを表します。ウィンドウサイズは、パーティションの 1 つのロー、複数のロー、またはすべてのローに設定できます。ウィンドウサイズは、現在のローの値からのオフセットとしてデータ値の範囲を指定するか (RANGE)、現在のローからの物理的なロー数としてオフセットを指定する (ROWS) ことで表すことができます。

RANGE 句を使用する場合は、範囲計算で値の格納が必要になるため、ORDER BY も指定する必要があります。範囲のための ORDER BY 句には式が 1 つ含まれていること、およびその式の結果が日付または数値になることが必要です。

ROWS 句または RANGE 句を指定しなかった場合、データベースサーバは、ORDER BY 句が指定されているかどうかに基づいて、デフォルトのウィンドウサイズを使用します。

PRECEDING clause

PRECEDING 句は、現在のローを参照ポイントとして使用して、ウィンドウの最初のローを定義するときに使用します。開始ローは、現在のローの前にあるロー数で表します。たとえば、`5 PRECEDING` と設定すると、ウィンドウは現在のローの前にある 5 番目のローから開始されます。

UNBOUNDED PRECEDING は、ウィンドウの最初のローを設定し、パーティションの最初のローにするときに使用します。

BETWEEN clause

BETWEEN 句は、現在のローを参照ポイントとして使用して、ウィンドウの最初のローと最後のローを定義するときに使用します。最初のローと最後のローは、それぞれ現在のローの前にあるロー数と後にあるロー数で表します。たとえば、`BETWEEN 3 PRECEDING AND 5 FOLLOWING` と設定すると、ウィンドウは現在のローの前にある 3 番目のローから開始され、現在のローの後の 5 番目のローで終了します。

BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING は、ウィンドウの最初のローと最後のローを設定することで、それぞれパーティションの最初のローと最後のローにするときに使用します。これは ROW 句や RANGE 句を指定する場合のデフォルト動作と同じです。

FOLLOWING clause

FOLLOWING 句は、現在のローを参照ポイントとして使用して、ウィンドウの最後のローを定義するときに使用します。最後のローは、現在のローの後にあるロー数で表します。

UNBOUNDED FOLLOWING は、ウィンドウの最後のローを設定し、パーティションの最後のローにするときに使用します。

備考

WINDOW 句は、SELECT 文の ORDER BY 句の前に指定する必要があります。

LIST 関数には例外がありますが、すべての集合関数を Window 関数として使用できます。ただし、ランキング集合関数 (RANK、DENSE_RANK、PERCENT_RANK、CUME_DIST、ROW_NUMBER) には ORDER BY 句が必要です。また、WINDOW 句やインライン定義には ROW 句と RANGE 句は使用できません。それ以外の Window 関数では、任意の句を使用できます。

権限

なし。

標準

ANSI/ISO SQL 標準

WINDOW 句と Window 集合関数は、オプションの SQL 言語機能 T611、"Elementary OLAP operations" と T612、"Advanced OLAP operations" を構成します。Window 関数 FIRST_VALUE と LAST_VALUE は、標準にありません。

例

次の例は、選択した州に在住する全従業員について、各従業員の給料と平均給料を返します。結果は State、Surname の順にソートされます。

```
SELECT EmployeeID, Surname, Salary, State,
  AVG( Salary ) OVER Salary_Window
FROM GROUPO.Employees
WINDOW Salary_Window AS ( PARTITION BY State )
```

```
ORDER BY State, Surname;
```

関連情報

SELECT 文 [1291 ページ]

1.4.4.294 WRITETEXT 文 [T-SQL]

CHAR、NCHAR、または BINARY カラムのログなしの更新を許可します。この機能は Transact-SQL との互換性のためにのみ提供されているものであり、使用しないことをお奨めします。

構文

```
WRITETEXT table-name.column-name
text-pointer [ WITH LOG ] data
```

備考

既存のカラム値を更新します。更新は、WITH LOG オプションが提供されないかぎり、トランザクションログに記録されません。ビューに対して WRITETEXT 操作を実行することはできません。

権限

なし。

関連する動作

WRITETEXT はトリガを起動しません。また、デフォルトでは、作業内容はトランザクションログに記録されません。

標準

ANSI/ISO SQL 標準

Transact-SQL 拡張。

例

次の Embedded SQL コードフラグメントは、WRITETEXT 文の使い方を示しています。この例での SELECT 文は、単一のローを返します。次の例は、指定されたローの Description カラムの内容を新しい値で置き換えます。

```
EXEC SQL create variable txtptr binary(16);
EXEC SQL set txtptr =
    ( SELECT textptr(Description)
        FROM MarketingInformation
        WHERE ProductID = '500' );
EXEC SQL writetext MarketingInformation.Description
   txtptr 'newdata';
```

関連情報

READTEXT 文 [T-SQL] [1250 ページ]
TEXTPTR 関数 [テキストとイメージ] [561 ページ]

1.5 テーブル

このソフトウェアでは、複数のテーブルの種類がサポートされています。

このセクションの内容:

システムテーブル [1413 ページ]
すべてのデータベースの構造は、システムテーブルに記述されています。

1.5.1 システムテーブル

すべてのデータベースの構造は、システムテーブルに記述されています。

システムテーブルは、ユーザ SYS によって所有されます。これらのテーブルの内容は、データベースサーバでのみ変更できます。したがって、テーブルの内容の変更に UPDATE、DELETE、INSERT 文は使用できません。また、ALTER TABLE 文と DROP 文を使って、これらのテーブルの構造を変更することもできません。チェックポイントが発生すると、システムビューが更新されます。

一部の例外を除き、システムテーブル内のデータは、対応するシステムビュー経由で公開されます。システムテーブル名はそれぞれ大文字の i (I) で始まります。対応するそれぞれのシステムビューも同じ名前になりますが、最初の i は削除されます。システムテーブルに関する情報を表示するには、対応するシステムビューのマニュアルを検索してください。

このセクションの内容:

アクセス可能な例外テーブル [1414 ページ]

直接アクセスできるシステムテーブルがいくつかあります。

関連情報

システムビュー [1784 ページ]

1.5.1.1 アクセス可能な例外テーブル

直接アクセスできるシステムテーブルがいくつかあります。

このセクションの内容:

1.5.1.1.1 DUMMY システムテーブル

DUMMY テーブルは、常に 1 つだけのローを持つ、読み込み専用のテーブルとして提供されています。

カラム名	カラム型	カラム制約	テーブル制約
dummy_col	INTEGER	NOT NULL	

これはデータベースから情報を抽出するのに役立ちます。次に、データベースから現在のユーザ ID と今日の日付を取り出す例を示します。

```
SELECT USER, today(*) FROM SYS.DUMMY;
```

dummy_col

このカラムは使用されません。テーブルはカラムなしでは作成できないので、このカラムが存在します。

DUMMY テーブルからの読み取りコストは、同様のユーザ作成テーブルからの読み取りコストよりも低いコストです。これは、DUMMY のテーブルページにはロックがないためです。

アクセスプランは、DUMMY テーブルのスキャンによって構築されるわけではありません。そうではなく、DUMMY への参照がローコンストラクタアルゴリズムに置き換えられ、これがテーブル参照を仮想化します。これにより、DUMMY の使用に伴う競合を排除できます。DUMMY は、テーブル名か相関名またはその両方として、短いプラン、長いプラン、グラフィカルプランに引き続き表示されます。

1.5.1.1.2 診断トレーシングテーブル (廃止予定)

次は、診断トレーシングに使用されるメインテーブルです。このテーブルは dbo ユーザが所有し、システムテーブルとはみなされません。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

システムテーブルとは異なり、診断テーブルにはデータを表示するための対応するビューがありません。テーブルを直接クエリできます。

これらテーブルの大部分には、似た名前と似たスキーマを持つグローバルで共有のテンポラリテーブルが存在します。たとえば、sa_diagnostic_blocking テーブルには、対応するグローバルなテンポラリテーブルとして sa_tmp_diagnostic_blocking テーブルがあり、スキーマは同じです。トレースセッション中に、診断データはこれらのテンポラリテーブルに書き込まれます。テンポラリテーブルはロギングされないため、トレースセッション中のパフォーマンスは優れています。サーバに与える影響を最小限に抑えるには重要です。

このセクションの内容:

sa_diagnostic_request テーブル (廃止予定) [1424 ページ]
sa_diagnostic_request テーブルは、dbo ユーザが所有します。また、全要求のマスタテーブルです。

sa_diagnostic_statement テーブル (廃止予定) [1425 ページ]
sa_diagnostic_statement テーブルは dbo ユーザが所有します。また、文のテキストを格納します。

sa_diagnostic_statistics テーブル (廃止予定) [1426 ページ]
sa_diagnostic_statistics テーブルは、dbo ユーザが所有します。また、サーバで維持されているパフォーマンスカウンタの履歴が含まれます。各ローは、特定時点の特定パフォーマンスカウンタの値を表します。

sa_diagnostic_tracing_level テーブル (廃止予定) [1426 ページ]
sa_diagnostic_tracing_level テーブルは、dbo ユーザが所有します。また、このテーブルの各ローは、トレーシングデータベースに送信する診断情報の種類を決定する条件です。

1.5.1.1.2.1 sa_diagnostic_auxiliary_catalog テーブル (廃止予定)

sa_diagnostic_auxiliary_catalog テーブルは、dbo ユーザが所有し、運用データベースとトレーシングデータベース間のデータベースオブジェクトのマッピングに使用されます。オブジェクトには、テーブル、プロシージャ、関数などがあります。このテーブルは、主にインデックスコンサルタントや TRACED_PLAN 関数に使用されます。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

カラム

カラム名	カラム型	説明
original_object_id	UNSIGNED BIGINT	メイントレーシングデータベースにある、このオブジェクトのオブジェクト ID。
local_object_id	UNSIGNED BIGINT	補助トレーシングデータベースにある、このオブジェクトのオブジェクト ID。
pages_if_table	UNSIGNED INT	オブジェクトがテーブルの場合、これはテーブルのページ数です。オブジェクトがテーブル以外の場合、この値は NULL です。
rows_if_table	UNSIGNED BIGINT	オブジェクトがテーブルの場合、これはテーブルのロー数です。オブジェクトがテーブル以外の場合、この値は NULL です。

関連情報

TRACED_PLAN 関数 [その他] (廃止予定) [568 ページ]

1.5.1.1.2.2 sa_diagnostic_blocking テーブル (廃止予定)

sa_diagnostic_blocking テーブルは、dbo ユーザが所有し、ブロッキングイベントを記録します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

ブロッキングイベントのロギングが有効な場合、リソースにアクセスを試みている間、接続がブロックされるたびにローがこのテーブルに挿入されます。一般的に、この問題はテーブルまたはローのロックによって発生します。大量にブロックがある場合、テーブルとローの競合を軽減するために、アプリケーションの同時実行性を確認する必要があります。

このテーブルには sa_diagnostic_blocking と sa_tmp_diagnostic_blocking という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
lock_id	UNSIGNED BIGINT	ローまたはテーブルのロックがブロックを引き起こした場合、ブロックを引き起こしたロックの ID。それ以外の場合は NULL。
request_id	UNSIGNED BIGINT	ブロックがカーソルのために発生しなかった場合、ブロックされた要求の ID。それ以外の場合は NULL。この値は、sa_diagnostic_request の要求に割り当てられている ID に対応します。
cursor_id	UNSIGNED BIGINT	ブロックがカーソルのために派生した場合はカーソルの ID。それ以外の場合は NULL。この値は、sa_diagnostic_cursor のカーソルに割り当てられている ID に対応します。
original_table_object_id	UNSIGNED BIGINT	テーブルのロックのためにブロックが発生した場合、ブロックが発生したテーブルの ID。それ以外の場合は NULL。
rowid	UNSIGNED BIGINT	テーブルのロックのためにブロックが発生した場合、ブロックが発生したテーブルの ID。それ以外の場合は NULL。
block_time	TIMESTAMP	ブロックが発生した時間。
unblock_time	TIMESTAMP	ブロックが終了した時間。
blocked_by	UNSIGNED INT	ロックを保持し、ブロックを引き起こした接続の ID。

1.5.1.1.2.3 sa_diagnostic_cachecontents テーブル (廃止予定)

sa_diagnostic_cachecontents テーブルは、dbo ユーザが所有します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

診断トレーシングが有効なときは、キャッシュコンテンツのスナップショットが定期的に撮られます。sa_diagnostic_cachecontents テーブルは、スナップショットを撮ったときのキャッシュ内の各テーブルのテーブルページ数、および各テーブルのロー数を記録します。オプティマイザはこの情報を使用して、クエリで最初に最適化した条件を再作成し、最適化の決定を下します。

sa_diagnostic_cachecontents テーブルのデータは、クエリアクティビティがあれば、20 秒ごとに更新されます。

このテーブルには sa_diagnostic_cachecontents と sa_tmp_diagnostic_cachecontents という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
"time"	TIMESTAMP	キャッシュのスナップショットを撮った時間。
original_table_object_id	UNSIGNED BIGINT	スナップショットで表される各テーブルのオブジェクト ID。
pages_in_cache	UNSIGNED INT	スナップショットに指定したテーブルの場合、スナップショット時のキャッシュの総ページ数。
num_table_pages	UNSIGNED INT	スナップショットで指定したテーブルの場合、テーブルの総ページ数。
num_table_rows	UNSIGNED BIGINT	スナップショットで指定したテーブルの場合、テーブルの総ロー数。

1.5.1.1.2.4 sa_diagnostic_connection テーブル (廃止予定)

sa_diagnostic_connection テーブルは、dbo ユーザが所有しています。また、ロギングセッション中にアクティブな各データベース接続に 1 つのローがあります。接続と接続解除がロギングセッション中に発生した場合、その発生時間は sa_diagnostic_request テーブルから派生します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルのほとんどの値は、接続プロパティの値を反映しています。

このテーブルには sa_diagnostic_connection と sa_tmp_diagnostic_connection という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
connection_number	UNSIGNED INT	データベースへの特定のユーザの接続を識別するためにデータベースサーバが割り当てる番号。この値は、Number 接続プロパティの値を反映しています。
connection_name	LONG VARCHAR	オプションの接続名プロパティ。この値は、Name 接続プロパティの値を反映しています。
user_name	LONG VARCHAR	データベースに接続するユーザ名。
comm_link	CHAR(40)	クライアント側のネットワークプロトコルオプションを指定します。この値は、CommLinks 接続プロパティの値を反映しています。
node_address	LONG VARCHAR	クライアント/サーバ接続のクライアント側に対応するノード。この値は、NodeAddress 接続プロパティの値を反映しています。
appinfo	LONG VARCHAR	クライアントコンピュータの IP アドレス、稼働しているオペレーティングシステムなど、クライアント処理に関する情報。この値は、AppInfo 接続プロパティの値を反映しています。

1.5.1.1.2.5 sa_diagnostic_cursor テーブル (廃止予定)

sa_diagnostic_cursor テーブルは、dbo ユーザが所有します。各ローは、ロギングセッション中に開かれた内部カーソルまたは外部カーソルを示します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルには sa_diagnostic_cursor と sa_tmp_diagnostic_cursor という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
cursor_id	UNSIGNED BIGINT	カーソルを識別するユニークな番号。

カラム名	カラム型	説明
query_id	UNSIGNED BIGINT	このカーソルの範囲にあるクエリを識別します。
isolation_level	TINYINT	このカーソルを開いた独立性レベル。
flags	UNSIGNED INT	内部的に使用。
forward_fetches	UNSIGNED INT	カーソルで実行された、プリフェッチなどの転送フェッチの数。
reverse_fetches	UNSIGNED INT	カーソルで実行された、プリフェッチなどのリバースフェッチの数。
absolute_fetches	UNSIGNED INT	カーソルで実行された絶対フェッチの数。
first_fetch_time_ms	UNSIGNED INT	最初のローをフェッチするときにかかる時間。
total_fetch_time_ms	UNSIGNED INT	フェッチにかかる時間。この値には、実際のフェッチ間 (シンクタイム) のアプリケーション処理時間を含みません。
plan_xml	LONG VARCHAR	カーソルを閉じたときにダンプされるカーソルの詳細プラン。このプランには、必要に応じて詳細な統計情報が含まれます。

1.5.1.1.2.6 sa_diagnostic_deadlock テーブル (廃止予定)

sa_diagnostic_deadlock テーブルは、dbo ユーザが所有します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

診断トレーシングが有効で、デッドロックイベントのトレーシングを含めるように設定されている場合、デッドロックが発生するたびに、ローセットがこのテーブルに挿入されます (デッドロックの一部である各接続の 1 ローが挿入されます)。1 のデッドロックイベントを構成するすべてのローセットは、snapshot_id でユニークに識別されます。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
snapshot_id	UNSIGNED BIGINT	このローが含まれるデッドロックイベントを識別する番号。このカラムは、スナップショットアイソレーションとは関係ありません。
snapshot_at	TIMESTAMP	デッドロックが発生した時刻。
waiter	UNSIGNED INT	このローが表す接続の接続数。

カラム名	カラム型	説明
request_id	UNSIGNED BIGINT	デッドロックが発生したときにこの接続が処理していた要求 ID。
original_table_object_id	UNSIGNED BIGINT	この接続がブロックされたときのテーブルのオブジェクト ID。
rowid	UNSIGNED BIGINT	この接続がブロックされたときのローのレコード ID。
owner	UNSIGNED INT	このローをロックした接続の接続数。
rollback_operation_count	UNSIGNED INT	コミットされていないオペレーションの数。

1.5.1.1.2.7 sa_diagnostic_hostvariable テーブル (廃止予定)

sa_diagnostic_hostvariable テーブルは dbo ユーザが所有しています。また、指定したカーソルが使用するホスト変数値が含まれます。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルには sa_diagnostic_hostvariable と sa_tmp_diagnostic_hostvariable という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
request_id	UNSIGNED BIGINT	ホスト変数が所属する要求の ID。
cursor_id	UNSIGNED BIGINT	ホスト変数が保持するカーソルの ID。
hostvar_num	UNSIGNED SMALLINT	SQL 文のホスト変数の順序位置。
hostvar_type	UNSIGNED TINYINT	内部でのみ使用。
hostvar_value	LONG NVARCHAR	ホスト変数の値を表す文字列。ホスト変数が整数または浮動の場合でも、この値は文字列として表されます。

1.5.1.1.2.8 sa_diagnostic_internalvariable テーブル (廃止予定)

sa_diagnostic_internalvariable テーブルは dbo ユーザが所有しています。また、指定した文が使用する内部 (ローカル) 変数値が含まれます。このテーブルは、主にインデックスコンサルタントや traced_plan 関数に使用されます。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルには sa_diagnostic_internalvariable と sa_tmp_diagnostic_internalvariable という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
request_id	UNSIGNED BIGINT	内部変数を含む要求 ID。
rowvariable_id	UNSIGNED INT	この値のロー変数のカラム番号。
variable_domain	UNSIGNED SMALLINT	内部でのみ使用。
variable_name	CHAR(128)	内部変数の名前。
variable_value	LONG NVARCHAR	内部変数の値を表す文字列。

関連情報

SQL 変数 [118 ページ]

1.5.1.1.2.9 sa_diagnostic_query テーブル (廃止予定)

sa_diagnostic_query テーブルは dbo ユーザが所有しています。また、特に最適化したコンテキストなど、クエリの最適化情報を格納しています。このテーブルのローは、クエリのオプティマイザの呼び出しを表します。最適化時にキャプチャされたプランはここに格納されます。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルの値の一部は、データベースオプション値を反映します。

このテーブルには sa_diagnostic_query と sa_tmp_diagnostic_query という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	クエリまたは要求が発生したときのロギングセッションの ID。
query_id	UNSIGNED BIGINT	クエリをユニークに識別する番号。

カラム名	カラム型	説明
statement_id	UNSIGNED BIGINT	クエリで文をユニークに識別する番号。
user_object_id	UNSIGNED BIGINT	このクエリが実行されたときのユーザのオブジェクト ID。クエリをプロシージャから実行した場合、これはプロシージャ所有者のユーザ ID になります。
start_time	TIMESTAMP	クエリを最適化した時刻。
cache_size_bytes	UNSIGNED BIGINT	クエリを最適化した時点のキャッシュサイズ (バイト単位)。
optimization_goal	TINYINT	クエリ処理の最適化の対象を、最初のローを迅速に返すこと、または完全な結果セットを返すコストを最小限に抑えることのどちらかに指定します。この値は、optimization_goal データベースオプションの値を反映しています。
optimization_level	TINYINT	SQL Anywhere クエリオプティマイザが SQL 文のアクセスプランの検索に費やす作業量を制御します。この値は、optimization_level データベースオプションの値を反映しています。
user_estimates	TINYINT	クエリの述部に含まれるユーザ選択性推定をクエリオプティマイザが尊重するか無視するかを制御します。この値は、user_estimates データベースオプションの値を反映しています。
optimization_workload	TINYINT	クエリ処理において、更新と読み込みを組み合わせた負荷に対して最適化するか、または大部分が読み込みベースの負荷に対して最適化するかを決定します。この値は、optimization_workload データベースオプションの値を反映しています。
available_requests	TINYINT	クエリ内並列処理のレベルを計算するときに内部的に使用されます。
active_requests	TINYINT	クエリ内並列処理のレベルを計算するときに内部的に使用されます。
max_tasks	TINYINT	クエリ内並列処理のレベルを計算するときに内部的に使用されます。
used_bypass	TINYINT	単純なクエリの回避を使用したかどうか。1 の値は、回避を使用したことを示します。0 の値は、クエリを完全に最適化したことを示します。
estimated_cost_ms	TINYINT	推定コスト (ミリ秒)。

カラム名	カラム型	説明
plan_explain	LONG VARCHAR	このクエリのテキストプラン表記。
plan_xml	LONG VARCHAR	クエリのグラフィカルプラン表記 (記録された場合)。
sql_rewritten	LONG VARCHAR	最適化を適用した後のクエリのテキスト。最適化のロギングが有効な場合、値はこのカラムにのみ表示されます。

1.5.1.1.2.10 sa_diagnostic_request テーブル (廃止予定)

sa_diagnostic_request テーブルは、dbo ユーザが所有します。また、全要求のマスタテーブルです。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラを使用して、データベースの問題を診断します。

要求はクエリ処理に関連するイベントであり、一般的に次のイベントが含まれます

。

- イベントの接続または接続解除
- 文の実行
- 文の準備
- カーソルイベントを開く、または削除

このテーブルには sa_diagnostic_request と sa_tmp_diagnostic_request という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	要求が発生したときのロギングセッション。
request_id	UNSIGNED BIGINT	要求をユニークに識別する番号。
start_time	TIMESTAMP	イベントが開始した時間。
finish_time	TIMESTAMP	文の実行の場合、文が完了した時間。それ以外の場合は NULL。
duration_ms	UNSIGNED INT	イベントの継続時間 (ミリ秒)。
connection_number	UNSIGNED INT	イベントをトリガさせた接続の ID。
request_type	UNSIGNED SMALLINT	要求の型。次のような値があります。開始された新しい診断トレースセッション、実行された SQL 文、開かれたカーソル、閉じられたカーソル、クライアントの接続、クライアントの切断、チェックポイントです。
statement_id	UNSIGNED BIGINT	イベントが文関連だった場合、トレーシング用途で文に割り当てられた ID。

カラム名	カラム型	説明
query_id	UNSIGNED BIGINT	イベントがクエリ関連だった場合、トレーシング用途でクエリに割り当てられた ID。
cursor_id	UNSIGNED BIGINT	イベントがカーソル関連だった場合、トレーシング用途でカーソルに割り当てられた ID。
sql_code	SMALLINT	このテーブルのローは、文、カーソル、クエリに関する操作を表すため、ほとんどの場合、SQL コードを返します。このカラムには返された SQL コードが含まれます。0 の SQL コードが返される場合、カラムには NULL が含まれます。

1.5.1.1.2.11 sa_diagnostic_statement テーブル (廃止予定)

sa_diagnostic_statement テーブルは dbo ユーザが所有します。また、文のテキストを格納します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルのローは、サーバ側で実行された SQL 文を示します。クライアント要求などの外部ソース、またはプロシージャ、トリガ、ユーザ定義関数などの内部ソースによって、この文が発行された可能性があります。ここには内部的な文のみが 1 セッションにつき 1 つのみ表示されます。

このテーブルには sa_diagnostic_statement と sa_tmp_diagnostic_statement という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	文が送信されたときのロギングセッション。
statement_id	UNSIGNED BIGINT	トレーシング処理用に文に割り当てたユニークな番号。
database_object	UNSIGNED BIGINT	文がプロシージャ、トリガ、または関数に由来する場合、これは ISYSOBJECT システムテーブルに指定されている ID です。
line_number	UNSIGNED SMALLINT	文が複合文の一部である場合、これは複合文内にその文が占める順序位置を反映します。
signature	UNSIGNED INT	内部的に同様のクエリをグループ化するときに使用します。
statement_text	LONG VARCHAR	文のテキスト。

1.5.1.1.2.12 sa_diagnostic_statistics テーブル (廃止予定)

sa_diagnostic_statistics テーブルは、dbo ユーザが所有します。また、サーバで維持されているパフォーマンスカウンタの履歴が含まれます。各ローは、特定時点の特定パフォーマンスカウンタの値を表します。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

このテーブルには sa_diagnostic_statistics と sa_tmp_diagnostic_statistics という 2 つのバージョンがあります。

カラム

カラム名	カラム型	説明
logging_session_id	UNSIGNED INT	診断情報が収集されたロギングセッションを一意に識別する番号。
"time"	TIMESTAMP	パフォーマンスカウンタ値をキャプチャした時刻。
counter_id	UNSIGNED SMALLINT	パフォーマンスカウンタをユニークに識別する番号。この counter_id が PROPERTY_NAME 関数を使用して表すプロパティの名前を取得できます。
type	TINYINT	データベース、サーバ、または接続の統計情報のいずれかを示します。サーバの場合は 0、データベースの場合は 1、接続の場合は 2、外部データベースの場合は 4 を使用できます。
connection_number	UNSIGNED INT	接続の統計情報の場合、このプロパティをキャプチャする接続番号。拡張データベース統計情報の場合、このプロパティをキャプチャしたファイルのファイル番号。それ以外の場合、値は 0 です。
counter_value	UNSIGNED INT	パフォーマンスカウンタの値。

関連情報

PROPERTY_NAME 関数 [システム] [473 ページ]

1.5.1.1.2.13 sa_diagnostic_tracing_level テーブル (廃止予定)

sa_diagnostic_tracing_level テーブルは、dbo ユーザが所有します。また、このテーブルの各ローは、トレーシングデータベースに送信する診断情報の種類を決定する条件です。

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。

ロギングデータがこのテーブルの 1 つ以上のローの条件に適合する場合、対応するデータがロギングされます。

このテーブルのデータは、CONNECT TRACING 文または REFRESH TRACING LEVEL 文を使用して作成されます。

カラム

カラム名	カラム型	説明
id	UNSIGNED INT	内部でのみ使用。
scope	CHAR(32)	以下に示す診断トレーシングの範囲。 • DATABASE • ORIGIN • USER • CONNECTION_NAME • CONNECTION_NUMBER • FUNCTION • PROCEDURE • EVENT • TRIGGER • TABLE
identifier	CHAR(128)	スコープの識別子。この値は、指定したスコープに応じて変化します。次に例を示します。 • `scope` が DATABASE の場合、`identifier` が提示されない場合があります。 • `scope` が ORIGIN の場合、`identifier` は Internal または External にする必要があります。 • `scope` が USER の場合、`identifier` はユーザの ID です。 • `scope` が CONNECTION_NAME または CONNECTION_NUMBER の場合、`identifier` はそれぞれ接続の名前または番号です。 • `scope` が FUNCTION、PROCEDURE、EVENT、TRIGGER、または TABLE の場合、`identifier` はオブジェクトの完全修飾識別子です。

カラム名	カラム型	説明
trace_type	CHAR(32)	指定したスコープについてトレースするデータ型を以下に示します。 • VOLATILE_STATISTICS • NONVOLATILE_STATISTICS • CONNECTION_STATISTICS • BLOCKING • PLANS • PLANS_WITH_STATISTICS • STATEMENTS • STATEMENTS_WITH_VARIABLES
trace_condition	CHAR(32)	プランにのみ適用されます。また、大規模でコストが高いクエリ、またはオプティマイザが最適化を選択しなかったクエリのどちらをトレースするかを制御します。使用できる値を以下に示します。 • NONE または NULL • SAMPLE_EVERY • ABSOLUTE_COST • RELATIVE_COST_DIFFERENCE
value	UNSIGNED INT	`condition` に関連付けられている値。たとえば、`condition` が SAMPLE_EVERY の場合、`condition_value` は時間 (ミリ秒) を反映した正の整数です。追加の規則は次のとおりです。 • `condition` が NULL または NONE の場合、`condition_value` はありません。 • `condition` が ABSOLUTE_COST の場合、`condition_value` は文の実際の合計実行コスト (ミリ秒) を反映します。 • `condition` が RELATIVE_COST_DIFFERENCE の場合、`condition_value` は推定コストに占める割合として実行コストを反映します。
enabled	BIT	ローを有効にするかどうか。つまり、ローのトレーシング設定を有効にするかどうか。1 の場合は有効にします。0 の場合は無効にします。

関連情報

ATTACH TRACING 文 (廃止) [737 ページ]
REFRESH TRACING LEVEL 文 (廃止予定) [1258 ページ]

1.5.1.1.3 RowGenerator テーブル (dbo)

dbo.RowGenerator テーブルは、255 のローを持つ読み込み専用のテーブルとして提供されています。このテーブルは、小規模な結果セットを作成するクエリと一連の数値を必要とするクエリに有用です。

RowGenerator テーブルは、システムプロシージャとシステムビューによって使用されます。このテーブルは、どのような方法でも修正できません。

sa_rowgenerator システムプロシージャでも一定範囲の数値を生成できます。

カラム名	カラム型
row_num	SMALLINT

row_num

1 ～ 255 の値

関連情報

sa_rowgenerator システムプロシージャ [1598 ページ]

1.6 システムプロシージャ

ソフトウェアには数百ものシステムがあり、その多くは内部でのみ使用されます。マニュアルでは、外部で使用されるシステムプロシージャについて説明します。

すべてのシステムプロシージャはセキュリティ機能として有効にしたり無効にすることができます。

このセクションの内容:

システムプロシージャとファンクションの詳細の表示 [1430 ページ]
SQL Central からシステムプロシージャと関数の定義にアクセスします。

Web サービスシステムプロシージャ [1431 ページ]
次のシステムプロシージャは、Web サービスの操作時に使用します。

ロールと権限のシステムプロシージャ [1431 ページ]
これらのプロシージャは、ロールと権限を管理するときに使用します。

MAPI と SMTP のシステムプロシージャ [1432 ページ]
Microsoft Messaging API 標準 (MAPI) または Internet 標準のメール転送プロトコル (SMTP) を使用して電子メールを送信するためのシステムプロシージャが含まれています。これらのシステムプロシージャは、拡張システムプロシージャとして実装されます。各プロシージャは、外部 DLL の関数を呼び出します。

ディレクトリとファイルのシステムプロシージャ [1434 ページ]
ディレクトリアクセスサーバを使用するか、ファイルや、sp_create_directory システムプロシージャなどのディレクトリシステムプロシージャを使用して、データベースサーバを実行しているコンピュータのローカルファイル構造にアクセスすることができます。

セキュリティ保護された機能のシステムプロシージャ [1436 ページ]
セキュリティ保護可能な機能の管理のため、いくつかのシステムプロシージャが提供されています。

Adaptive Server Enterprise のシステムプロシージャとカタログプロシージャ [1436 ページ]
SQL Anywhere には、Adaptive Server Enterprise システムおよびカタログプロシージャのサブセットのサポートが実装されました。ただし、このプロシージャの使用の詳細については、Adaptive Server Enterprise のマニュアルを参照してください。

システムプロシージャのアルファベット順リスト [1438 ページ]
データを返したり、データで操作を実行するために使用できるシステムプロシージャがあります。

1.6.1 システムプロシージャとファンクションの詳細の表示

SQL Central からシステムプロシージャと関数の定義にアクセスします。

手順

1. SQL Anywhere17 プラグインを使用してデータベースに接続します。
2. データベースを右クリックし、*[所有者フィルタの設定]* をクリックします。
3. *[DBO]* をクリックしてから、*[OK]* をクリックします。
4. 左ウィンドウ枠で、*[プロシージャとファンクション]* をダブルクリックします。
5. 左ウィンドウ枠でプロシージャを選択し、右ウィンドウ枠で *[SQL]* タブをクリックします。

結果

プロシージャまたはファンクションの定義が *[SQL]* タブに表示されます。

1.6.2 Web サービスシステムプロシージャ

次のシステムプロシージャは、Web サービスの操作時に使用します。

- sa_http_header_info システムプロシージャ
- sa_http_php_page システムプロシージャ
- sa_http_php_page_interpreted システムプロシージャ
- sa_http_variable_info システムプロシージャ
- sa_set_http_header システムプロシージャ
- sa_set_http_option システムプロシージャ
- sa_set_soap_header システムプロシージャ

関連情報

Web サービス関数 [212 ページ]
sa_http_header_info システムプロシージャ [1526 ページ]
sa_http_php_page システムプロシージャ [1528 ページ]
sa_http_php_page_interpreted システムプロシージャ [1530 ページ]
sa_http_variable_info システムプロシージャ [1532 ページ]
sa_set_http_header システムプロシージャ [1619 ページ]
sa_set_http_option システムプロシージャ [1621 ページ]
sa_set_soap_header システムプロシージャ [1625 ページ]

1.6.3 ロールと権限のシステムプロシージャ

これらのプロシージャは、ロールと権限を管理するときに使用します。

- sp_displayroles システムプロシージャ
- sp_has_role システムプロシージャ
- sp_objectpermission システムプロシージャ
- sp_proc_priv システムプロシージャ
- sp_sys_priv_role_info システムプロシージャ

関連情報

sp_displayroles システムプロシージャ [1672 ページ]
sp_has_role システムプロシージャ [1685 ページ]
sp_objectpermission システムプロシージャ [1698 ページ]
sp_proc_priv システムプロシージャ [1705 ページ]

sp_sys_priv_role_info システムプロシージャ [1734 ページ]

1.6.4 MAPI と SMTP のシステムプロシージャ

Microsoft Messaging API 標準 (MAPI) または Internet 標準のメール転送プロトコル (SMTP) を使用して電子メールを送信するためのシステムプロシージャが含まれています。これらのシステムプロシージャは、拡張システムプロシージャとして実装されます。各プロシージャは、外部 DLL の関数を呼び出します。

これらのプロシージャは dbo ロールに所有されます。ユーザがこれらのシステムプロシージャを使用するには、それらの EXECUTE 権限が必要です。

MAPI または SMTP システムプロシージャを使用するには、データベースサーバコンピュータから MAPI または SMTP E-mail システムにアクセスできるようにします。

MAPI システムプロシージャと SMTP システムプロシージャは、次のとおりです。

xp_startmail

MAPI メッセージシステムにログオンして、指定したメールアカウントでメールセッションを開始します。

xp_startsmtp

SMTP メッセージシステムにログオンして、指定したメールアカウントでメールセッションを開始します。

xp_sendmail

指定したユーザにメールメッセージを送信します。

xp_stopmail

MAPI メールセッションを閉じます。

xp_stopsmtp

SMTP メールセッションを閉じます。

xp_get_mail_error_code

最後の SMTP エラーまたは MAPI エラーに関する情報を返します。

xp_get_mail_error_text

最後の SMTP エラーのテキストを返します。

例

次のプロシージャは、バックアップが完了したことをユーザグループに通知します。リターンコードはチェックしません。

```
CREATE PROCEDURE notify_backup( )
BEGIN
    CALL xp_startmail(  mail_user='ServerAccount',
                        mail_password='ServerPassword' );
    CALL xp_sendmail( recipient='IS Group',
                      subject='Backup',
                      "message"='Backup completed' );
    CALL xp_stopmail( );
END;
```

次のプロシージャは、バックアップが完了してエラーチェックが含まれていることをユーザグループに通知します。

```
CREATE OR REPLACE PROCEDURE notify_backup_with_error_check( )
```

```
BEGIN
    DECLARE result_code INTEGER;
    DECLARE error_code INTEGER;
    DECLARE error_text LONG VARCHAR;
    SET result_code = xp_startmail(  'ServerAccount', -- mail_user
                                     'ServerPassword' -- mail_password
                                  );
    IF result_code = 0 THEN
         SET result_code = xp_sendmail( 'IS Group', --recipient
                                        'Backup', -- subject
                                        NULL, -- cc_recipient
                                        NULL, -- bcc_recipient
                                        NULL, -- query
                                        'Backup completed' -- message
                                      );
    ENDIF;
    IF result_code = 0 THEN
        SET result_code = xp_stopmail( );
    ENDIF;
    IF result_code = 0 THEN
        MESSAGE 'Backup completed message successfully sent';
    ELSE
        SET error_code = xp_get_mail_error_code();
        SET error_text = xp_get_mail_error_text();
        MESSAGE 'Error:' || result_code ||
                ' Code:' || error_code ||
                ' Text:' || error_text;
    ENDIF;
END;;
```

このセクションの内容:

MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]

MAPI システムプロシージャまたは SMTP システムプロシージャによって、次のコードが返される場合があります。

関連情報

xp_startmail システムプロシージャ [1775 ページ]

xp_startsmtp システムプロシージャ [1777 ページ]

xp_sendmail システムプロシージャ [1770 ページ]

xp_stopmail システムプロシージャ [1780 ページ]

xp_stopsmtp システムプロシージャ [1781 ページ]

xp_get_mail_error_code システムプロシージャ [1761 ページ]

xp_get_mail_error_text システムプロシージャ [1762 ページ]

1.6.4.1 MAPI システムプロシージャと SMTP システムプロシージャのリターンコード

MAPI システムプロシージャまたは SMTP システムプロシージャによって、次のコードが返される場合があります。

ステータスコード	意味
-1	未知のエラー [1]
0	成功
1	無効なパラメータが指定されました
2	メモリ不足です
3	xp_startmail または xp_startsmtp が呼び出されませんでした
4	ホスト名が不正です
5	接続エラー [1]
6	セキュリティで保護された接続のエラー [1]
7	MAPI 関数を使用できません [1]

[1] リターンコードに関する追加情報を取得するには、xp_get_mail_error_code および xp_get_mail_error_text システムプロシージャを使用します。

関連情報

xp_startmail システムプロシージャ [1775 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]
xp_get_mail_error_code システムプロシージャ [1761 ページ]
xp_get_mail_error_text システムプロシージャ [1762 ページ]

1.6.5 ディレクトリとファイルのシステムプロシージャ

ディレクトリアクセスサーバを使用するか、ファイルや、sp_create_directory システムプロシージャなどのディレクトリシステムプロシージャを使用して、データベースサーバを実行しているコンピュータのローカルファイル構造にアクセスすることができます。

ファイルとディレクトリのシステムプロシージャはディレクトリアクセスサーバよりも簡単に使用できますが、ディレクトリアクセスのプロキシテーブルおよびサーバほど柔軟でも強力でもありません。

サーバが存在するコンピュータのディレクトリやファイルを操作するには、次のシステムプロシージャを使用します。

sp_list_directory システムプロシージャ

指定されたディレクトリの内容を一覧表示します。
sp_create_directory システムプロシージャ

指定されたディレクトリを作成します。
sp_copy_directory システムプロシージャ

指定されたディレクトリを別の場所にコピーします。
sp_move_directory システムプロシージャ

指定されたディレクトリを移動します。
sp_delete_directory システムプロシージャ

指定されたディレクトリを削除します。
sp_copy_file システムプロシージャ

指定されたファイルをコピーします。
sp_move_file システムプロシージャ

指定されたファイルを移動します。
sp_delete_file システムプロシージャ

指定されたファイルを削除します。

次のシステムプロシージャも、ファイルの読み書きに使用できます。

xp_read_file システムプロシージャ

ファイルを読み込み、そのファイルの内容を LONG BINARY 変数として返します。
xp_write_file システムプロシージャ

SQL 文からファイルにデータを書き込みます。
sp_disk_info システムプロシージャ 指定されたファイルまたはディレクトリのディスクドライブ情報を返します。

> **注記**
>
> ディスクサンドボックスが有効な場合、プロシージャが機能するには、システムプロシージャにより要求される場所がサンドボックスの内部にある必要があります。

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]
xp_read_file システムプロシージャ [1767 ページ]

xp_write_file システムプロシージャ [1782 ページ]

1.6.6 セキュリティ保護された機能のシステムプロシージャ

セキュリティ保護可能な機能の管理のため、いくつかのシステムプロシージャが提供されています。

- sa_server_option システムプロシージャ
- sp_alter_secure_feature_key システムプロシージャ
- sp_create_secure_feature_key システムプロシージャ
- sp_drop_secure_feature_key システムプロシージャ
- sp_list_secure_feature_key システムプロシージャ
- sp_use_secure_feature_key システムプロシージャ

関連情報

sa_server_option システムプロシージャ [1606 ページ]
sp_alter_secure_feature_key システムプロシージャ [1657 ページ]
sp_create_secure_feature_key システムプロシージャ [1664 ページ]
sp_drop_secure_feature_key システムプロシージャ [1675 ページ]
sp_list_secure_feature_key システムプロシージャ [1693 ページ]
sp_use_secure_feature_key システムプロシージャ [1751 ページ]

1.6.7 Adaptive Server Enterprise のシステムプロシージャとカタログプロシージャ

SQL Anywhere には、Adaptive Server Enterprise システムおよびカタログプロシージャのサブセットのサポートが実装されました。ただし、このプロシージャの使用の詳細については、Adaptive Server Enterprise のマニュアルを参照してください。

このセクションの内容:

Adaptive Server Enterprise システムプロシージャ [1437 ページ]
SQL Anywhere には Adaptive Server Enterprise のシステムプロシージャがいくつか含まれています。

Adaptive Server Enterprise カタログプロシージャ [1437 ページ]
SQL Anywhere は、Adaptive Server Enterprise カタログプロシージャのサブセットを実装します。

1.6.7.1 Adaptive Server Enterprise システムプロシージャ

SQL Anywhere には Adaptive Server Enterprise のシステムプロシージャがいくつか含まれています。

これらのプロシージャは、Adaptive Server Enterprise と同じファンクションを実行しますが、まったく同じではありません。これらのプロシージャを使用する既存のスクリプトがある場合は、プロシージャコードを調べてください。ストアドプロシージャのテキストを表示するには、次のコマンドを実行します。

```
sp_helptext 'dbo.procedure_name'
```

次の表に、実装されたシステムプロシージャを示します。

システムプロシージャの名前／パラメータ	説明
sp_addgroup @grpname	データベースにグループを追加します
sp_addlogin @login_name , @passwd [, @defaultdb [, @deflanguage [, @fullname]]]	データベースに新規ログイン ID を追加します
sp_addmessage @message_num , @message_text [, @language]	ストアドプロシージャの PRINT と RAISERROR の呼び出しで使用され、ISYSUSERMESSAGE にユーザ定義メッセージを追加します
sp_addtype @typename , @phystype [, @ident_null]	ユーザ定義データ型を作成します
sp_adduser @login_name [, @name_in_db [, @grpname]]	データベースに新規ユーザ ID を追加します
sp_changegroup @grpname , @name_in_db	ユーザグループを変更、またはグループにユーザを追加します
sp_dropgroup @grpname	データベースからグループを削除します
sp_droplogin @login_name	データベースからログイン ID を削除します
sp_dropmessage @message_number [, @language]	ユーザ定義メッセージを削除します
sp_droptype @typename	ユーザ定義データ型を削除します
sp_dropuser @name_in_db	データベースからユーザ ID を削除します
sp_getmessage @message_num , @msg_var [, @language]	PRINT 文と RAISERROR 文で使用され、ISYSUSERMESSAGE から格納されたメッセージ文字列を取り出します
sp_helptext [@objname]	システムプロシージャ、トリガ、またはビューのテキストを表示します
sp_password @caller_pswd , @new_pswd [, @login_name]	ユーザ ID のパスワードを追加または変更します

1.6.7.2 Adaptive Server Enterprise カタログプロシージャ

SQL Anywhere は、Adaptive Server Enterprise カタログプロシージャのサブセットを実装します。

これらのプロシージャは、Adaptive Server Enterprise と同じファンクションを実行しますが、まったく同じではありません。これらのプロシージャを使用する既存のスクリプトがある場合は、プロシージャコードを調べてください。ストアドプロシージャのテキストを表示するには、次のコマンドを実行します。

```
sp_helptext 'dbo.procedure_name'
```

次に、実装されたカタログプロシージャを示します。

カタログプロシージャの名前／パラメータ	説明
sp_column_privileges	サポート対象外
sp_columns [@table_name [, @table_owner [, @table_qualifier [, @column_name]]]]	指定したカラムのデータ型を返します
sp_fkeys [@pktable_name [, @pktable_owner [, @pktable_qualifier [, @fktable_name [, @fktable_owner [, @fktable_qualifier]]]]]]	指定したテーブルの外部キー情報を返します
sp_pkeys @table_name [, @table_owner [, @table_qualifier]]	指定したテーブルのプライマリキー情報を返します
sp_special_columns @table_name [, @table_owner [, @table_qualifier [, @col_type]]]	指定したテーブルのローをユニークに識別するのに最適な一連のカラムを返します
sp_sproc_columns @sp_name [, @sp_owner [, @sp_qualifier [, @column_name]]]	ストアドプロシージャの入力パラメータとリターンパラメータの情報を返します
sp_statistics [@table_name [, @table_owner [, @table_qualifier [, @index_name [, @is_unique]]]]	テーブルとそのインデックスの情報を返します
sp_stored_procedures [@sp_name [, @sp_owner [, @sp_qualifier]]]	1 つ以上のストアドプロシージャの情報を返します
sp_tables [@table_name [, @table_owner [, @table_qualifier [, @table_type]]]	FROM 句で表示できるオブジェクトのリストを返します

1.6.8 システムプロシージャのアルファベット順リスト

データを返したり、データで操作を実行するために使用できるシステムプロシージャがあります。

sa_get_table_definition など、一部のシステムプロシージャは関数として実装されています。しかし、それらもシステムプロシージャと同じコンテキストと方法で使用されるため、システムプロシージャに含められており、システムプロシージャと同じような名前が付けられています (sa_xxx)。

このセクションの内容:

sa_column_stats システムプロシージャ [1457 ページ]
指定したカラムについて多様な統計情報を返します。この統計情報は、オプティマイザが使用するために管理されているカラムの統計情報とは関係がありません。

sa_conn_activity システムプロシージャ [1460 ページ]
サーバ上に示されている指定したデータベース接続に関して、最後に作成された SQL 文を返します。

sa_conn_compression_info システムプロシージャ [1462 ページ]
通信の圧縮率を要約します。

sa_conn_info システムプロシージャ [1464 ページ]
接続プロパティ情報をレポートします。

sa_conn_list システムプロシージャ [1468 ページ]
接続 ID を含む結果セットを返します。

sa_conn_options システムプロシージャ [1469 ページ]
データベースオプションに対応する接続プロパティのプロパティ情報を返します。

sa_conn_properties システムプロシージャ [1471 ページ]
接続プロパティ情報をレポートします。

sa_convert_ml_progress_to_timestamp システムプロシージャ [1473 ページ]
Mobile Link のスクリプト化されたアップロードの場合のみ。スクリプト化されたアップロードの進行状況値を UNSIGNED BIGINT から TIMESTAMP に変換します。

sa_convert_timestamp_to_ml_progress システムプロシージャ [1474 ページ]
Mobile Link のスクリプト化されたアップロードの場合のみ。スクリプト化されたアップロードの進行状況値を TIMESTAMP から UNSIGNED BIGINT に変換します。

sa_copy_cursor_to_temp_table システムプロシージャ [1475 ページ]
テンポラリテーブルを作成し、そのテーブルにオープンカーソルの結果セットをコピーします。

sa_cpu_topology システムプロシージャ [1477 ページ]
データベースサーバを実行しているコンピュータのプロセッサのトポロジを返します。

sa_db_info システムプロシージャ [1479 ページ]
データベースのプロパティ情報をレポートします。

sa_db_list システムプロシージャ [1481 ページ]
データベース ID を返します。

sa_db_option システムプロシージャ [1482 ページ]
データベースの実行中に、データベースオプションを上書きします。

sa_db_properties システムプロシージャ [1485 ページ]
データベースのプロパティ情報をレポートします。

sa_dependent_views システムプロシージャ [1487 ページ]
指定したテーブルまたはビューのすべての従属ビューリストを返します。

sa_describe_cursor システムプロシージャ [1488 ページ]
カーソルのカラムの名前と型に関する情報を記述します。

sa_describe_query システムプロシージャ [1491 ページ]
1 つのローにクエリの各出力カラムを記述したクエリの結果セットを記述します。

sa_describe_shapefile システムプロシージャ [1495 ページ]

ESRI シェイプファイルに含まれるカラムの名前と型を記述します。このシステム機能は、空間データ機能とともに使用します。

sa_disable_auditing_type システムプロシージャ [1497 ページ]
特定のイベントの監査を無効にします。

sa_disk_free_space システムプロシージャ [1499 ページ]
DB 領域、トランザクションログ、トランザクションログミラー、テンポラリファイルに使用可能な領域に関する情報を報告します。

sa_enable_auditing_type システムプロシージャ [1500 ページ]
監査に含めるイベントを指定します。

sa_eng_properties システムプロシージャ [1502 ページ]
データベースサーバのプロパティ情報をレポートします。

sa_error_stack_trace システムプロシージャ [1504 ページ]
エラーハンドラを呼び出したエラーのスタックトレースを返します。

sa_event_schedules システムプロシージャ [1506 ページ]
イベントについてのスケジュール情報を表示します。

sa_external_library_unload システムプロシージャ [1507 ページ]
外部ライブラリをアンロードします。

sa_flush_cache システムプロシージャ [1508 ページ]
データベースサーバキャッシュ内の現在のデータベースに対するすべてのページを空にします。

sa_flush_statistics システムプロシージャ [1509 ページ]
すべてのコストモデル統計をデータベースサーバキャッシュに保存します。

sa_get_bits システムプロシージャ [1510 ページ]
ビット文字列を取得し、文字列内の各ビットのローを返します。デフォルトでは、1 のビット値を持つローのみが返されます。

sa_get_dtt システムプロシージャ [1512 ページ]
コストモデルの一部であるディスク転送時間 (DTT: Disk Transfer Time) モデルの現在の値をレポートします。

sa_get_dtt_groupreads システムプロシージャ [1514 ページ]
データベースサーバでグループ読み込みを発行するコストを評価し、レポートします。

sa_get_histogram システムプロシージャ [1515 ページ]
カラムのヒストグラムを取得します。

sa_get_ldapserver_status システムプロシージャ [1517 ページ]
LDAP サーバの現在のステータスを特定できます。

sa_get_request_profile システムプロシージャ [1518 ページ]
要求ログを分析し、同様の文の実行時間を判断します。

sa_get_request_times システムプロシージャ [1520 ページ]
要求ログを分析し、文の実行時間を判別します。

sa_get_server_messages システムプロシージャ [旧式] [1522 ページ]
データベースサーバメッセージウィンドウの定数を結果セットとして返すことができます。このシステムプロシージャは廃止される予定です。代わりに sa_server_messages システムプロシージャを使用します。

sa_get_table_definition システムプロシージャ [1523 ページ]

指定されたテーブルとそのインデックス、外部キー、トリガ、付与された権限の作成に必要な SQL 文を含む LONG VARCHAR 文字列を返します。

sa_get_user_status システムプロシージャ [1524 ページ]
ユーザの現在のステータスを特定できます。

sa_http_header_info システムプロシージャ [1526 ページ]
HTTP 要求ヘッダ名と値を返します。

sa_http_php_page システムプロシージャ [1528 ページ]
ヘッダ、GET または POST データ、プロトコルバージョン、要求 URL、メソッドなど、現在の HTTP 要求のコンテキスト情報を使用して、PHP インタプリタによって解釈される PHP コードを渡した結果を返します。

sa_http_php_page_interpreted システムプロシージャ [1530 ページ]
ヘッダ、GET または POST データ、プロトコルバージョン、要求 URL、メソッドなど、指定したコンテキスト情報のパラメータを使用して、PHP インタプリタによって解釈される PHP コードを渡した結果を返します。

sa_http_variable_info システムプロシージャ [1532 ページ]
HTTP 変数名と値を返します。

sa_index_density システムプロシージャ [1534 ページ]
インデックス内の断片化とスキューの量に関する情報をレポートします。

sa_index_levels システムプロシージャ [1536 ページ]
インデックス内のレベル数が報告され、パフォーマンスのチューニングに役立てることができます。

sa_install_feature システムプロシージャ [1539 ページ]
空間機能の追加のように、追加機能をインストールします。

sa_java_loaded_classes システムプロシージャ [1540 ページ]
データベースサーバによって Java VM に現在ロードされているクラスをリストします。

sa_list_cursors システムプロシージャ [1541 ページ]
現在の接続で使用されているカーソルのリストを返します。

sa_list_statements システムプロシージャ [1543 ページ]
現在の接続で使用されている文のリストを返します。

sa_load_cost_model システムプロシージャ [1544 ページ]
現在のコストモデルを、指定したファイルに格納されているコストモデルで置換します。

sa_locks システムプロシージャ [1546 ページ]
データベース内のすべてのロック (ニューテックスを含む) を表示します。

sa_make_object システムプロシージャ (廃止予定) [1550 ページ]
ALTER 文を実行する前に、オブジェクトのスケルトンインスタンスが存在することを確認します。

sa_materialized_view_can_be_immediate システムプロシージャ [1552 ページ]
指定されたマテリアライズドビューを即時ビューとして定義できるかどうかを返します。

sa_materialized_view_info システムプロシージャ [1554 ページ]
指定されたマテリアライズドビューに関する情報を返します。

sa_migrate システムプロシージャ [1560 ページ]
SQL Anywhere データベースにリモートテーブルセットをマイグレートします。

sa_migrate_create_fks システムプロシージャ [1563 ページ]
dbo.migrate_remote_fks_list テーブルにリストされている各テーブルの外部キーを作成します。

sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
dbo.migrate_remote_fks_list テーブルに移植します。

sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
dbo.migrate_remote_table_list テーブルに移植します。

sa_migrate_create_tables システムプロシージャ [1568 ページ]
dbo.migrate_remote_table_list テーブルにリストされている各テーブルのプロキシテーブルとベーステーブルを作成します。

sa_migrate_data システムプロシージャ [1569 ページ]
リモートデータベーステーブルから SQL Anywhere ターゲットデータベースにデータをマイグレートします。

sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]
マイグレーションの目的で作成されたプロキシテーブルを削除します。

sa_mirror_server_status システムプロシージャ [1572 ページ]
現在のサーバと現在のサーバから直接または間接的にログページを受信するすべてのサーバの接続ステータスを返します。プライマリサーバでは、接続されているすべてのサーバのステータスを返します。

sa_nchar_terms システムプロシージャ [1574 ページ]
NCHAR 文字列を単語に分割し、各単語とその位置を 1 つのローとして返します。

sa_performance_diagnostics システムプロシージャ [1576 ページ]
データベースサーバが要求のタイミングの記録を有効にしたとき、すべての接続について要求のタイミング情報の概要を返します。

sa_performance_statistics システムプロシージャ [1581 ページ]
サーバ、データベース、および接続のパフォーマンス統計を返します。

sa_post_login_procedure システムプロシージャ [1582 ページ]
ユーザのパスワードの有効期限切れが近づいているかどうかを判断します。

sa_procedure_profile システムプロシージャ [1584 ページ]
データベースで実行されたプロシージャ、関数、イベント、またはトリガ内の各行について、実行時間に関する情報をレポートします。

sa_procedure_profile_summary システムプロシージャ [1587 ページ]
データベースで実行されたすべてのプロシージャ、ファンクション、イベント、またはトリガの実行時間に関する概要情報をレポートします。

sa_recompile_views システムプロシージャ [1589 ページ]
カタログに格納された、カラム定義を持たないビュー定義を見つけ、カラム定義を作成します。

sa_refresh_materialized_views システムプロシージャ [1591 ページ]
初期化されていないステータスであるすべてのマテリアライズドビューを初期化します。

sa_refresh_text_indexes システムプロシージャ [1592 ページ]
MANUAL REFRESH または AUTO REFRESH と定義されているすべてのテキストインデックスを再表示します。

sa_remove_tracing_data システムプロシージャ (廃止予定) [1593 ページ]
診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。sa_remove_tracing_data システムプロシージャは、診断トレーシングテーブルから、指定したロギング (トレーシング) セッション ID に関連するすべてのレコードを永久的に削除します。

sa_report_deadlocks システムプロシージャ [1594 ページ]
データベースサーバによって作成された内部バッファから、デッドロックに関する情報を取り出します。

sa_reserved_words システムプロシージャ [1596 ページ]
予約語のリストを返します。SQL 文で使用するキーワードの多くは予約語です。

sa_reset_identity システムプロシージャ [1597 ページ]
次の identity の値をテーブルに設定できます。このシステムプロシージャを使用して、次に挿入されるローの AUTOINCREMENT 値を変更します。

sa_rowgenerator システムプロシージャ [1598 ページ]
指定された開始値と終了値の間のローを格納した結果セットを返します。

sa_save_trace_data システムプロシージャ [1601 ページ]
トレーシングデータを基本テーブルに保存します。

sa_send_udp システム関数 [1602 ページ]
指定されたアドレスに UDP パケットを送信します。

sa_server_messages システムプロシージャ [1603 ページ]
データベースサーバメッセージウィンドウのメッセージを結果セットとして返すことができます。

sa_server_option システムプロシージャ [1606 ページ]
データベースサーバの実行中に、データベースサーバオプションを上書きします。

sa_set_http_header システムプロシージャ [1619 ページ]
Web サービスが HTTP 応答ヘッダを設定できるようにします。

sa_set_http_option システムプロシージャ [1621 ページ]
プロセス制御のため、Web サービスが HTTP オプションを設定できるようにします。

sa_set_soap_header システムプロシージャ [1625 ページ]
SOAP 応答の SOAP ヘッダの設定を許可します。このプロシージャは、SOAP Web サービスから呼び出されたストアドプロシージャ内で使用されます。

sa_set_tracing_level システムプロシージャ (廃止予定) [1627 ページ]
診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。診断トレーシングテーブルに格納するトレーシング情報のレベルを初期化します。

sa_snapshots システムプロシージャ [1629 ページ]
現在アクティブなスナップショットのリストを返します。

sa_split_list システムプロシージャ [1630 ページ]
デリミタで区切った値の文字列を使用して、ローのセットを返します (各値に 1 つのロー)。

sa_stack_trace システムプロシージャ [1633 ページ]
現在の呼び出しの場所につながるスタックトレースを返します。

sa_statement_text システムプロシージャ [1636 ページ]
各行に項目が 1 つずつ表示されるように、SELECT 文をフォーマットします。これは、すべての改行文字が削除されている要求ログから、長い文を表示する場合に便利です。

sa_table_fragmentation システムプロシージャ [1637 ページ]
データベーステーブルの断片化に関する情報をレポートします。

sa_table_page_usage システムプロシージャ [1639 ページ]
データベーステーブルのページの使用状況に関する情報をレポートします。

sa_table_stats システムプロシージャ [1640 ページ]
各テーブルから読み込まれたページ数に関する情報をレポートします。

sa_text_index_stats システムプロシージャ [1641 ページ]
データベース内のテキストインデックスに関する統計情報を返します。

sa_text_index_vocab システムプロシージャ [1643 ページ]
CHAR テキストインデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数のリストを返します。

sa_text_index_vocab_nchar システムプロシージャ [1645 ページ]
NCHAR テキストインデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数のリストを返します。

sa_transactions システムプロシージャ [1647 ページ]
現在アクティブなトランザクションのリストを返します。

sa_unload_cost_model システムプロシージャ [1649 ページ]
現在のコストモデルを特定のファイルにアンロードします。

sa_user_defined_counter_add システムプロシージャ [1650 ページ]
指定した量でユーザ定義カウンタの値を調整します。

sa_user_defined_counter_set システムプロシージャ [1652 ページ]
指定した値にユーザ定義カウンタを設定します。

sa_validate システムプロシージャ [1653 ページ]
データベースの全部または一部を検証します。

sa_verify_password システムプロシージャ [1656 ページ]
現在のユーザのパスワードを検証します。

sp_alter_secure_feature_key システムプロシージャ [1657 ページ]
認証コード、機能リスト、またはその両方を変更して、以前に定義されたセキュリティ保護済み機能キーを変更します。

sp_auth_sys_role_info システムプロシージャ [1659 ページ]
以前のバージョンの SQL Anywhere から、それに対応する互換ロールへの、権限のマッピングを返します。

sp_copy_directory システムプロシージャ [1660 ページ]
ディレクトリを指定された場所にコピーします。

sp_copy_file システムプロシージャ [1662 ページ]
ファイルを指定された場所にコピーします。

sp_create_directory システムプロシージャ [1663 ページ]
コンピュータ上にディレクトリを作成します。

sp_create_secure_feature_key システムプロシージャ [1664 ページ]
新しいセキュリティ保護済み機能キーを作成します。

sp_db_cache_contents システムプロシージャ [1666 ページ]
現在キャッシュにあるページを示すビットマップとともに、DB 領域ごとにローを 1 つ返します。

sp_delete_directory システムプロシージャ [1668 ページ]
指定されたディレクトリを削除します。

sp_delete_file システムプロシージャ [1669 ページ]
指定されたファイルをコンピュータから削除します。

sp_disk_info システムプロシージャ [1671 ページ]
特定のパスのディスクドライブ情報を返します。

sp_displayroles システムプロシージャ [1672 ページ]
指定されたシステム権限、システムロール、ユーザ定義ロール、またはユーザ名に付与されたすべてのロールを返すか、またはロールの階層ツリー全体を表示します。

sp_drop_secure_feature_key システムプロシージャ [1675 ページ]
セキュリティ保護済み機能キーを削除します。

sp_find_top_statements システムプロシージャ [1677 ページ]
ログに記録された文とプランの各組み合わせに対するパフォーマンス統計をレポートします。

sp_forward_to_remote_server システムプロシージャ [1679 ページ]
アプリケーションがリモートサーバ上で SQL 文を実行し、その文によって生成される結果セットを取得できます。

sp_generate_key_pair システムプロシージャ [1681 ページ]
新しい RSA 暗号化キーペアを作成します。

sp_get_last_synchronize_result システムプロシージャ [1682 ページ]
SYNCHRONIZE 文によって最後に開始された同期に関する情報を返します。

sp_has_role システムプロシージャ [1685 ページ]
プロシージャのインボーカに、指定されたシステム権限またはユーザ定義ロールが付与されているかどうかを返します。

sp_http_listeners システムプロシージャ [1687 ページ]
指定されたデータベースに使用される HTTP および HTTPS 接続リスナーをリストします。

sp_list_directory システムプロシージャ [1688 ページ]
指定されたディレクトリのファイルとサブディレクトリに関する情報を返します。

sp_list_mutexes_semaphores システムプロシージャ [1691 ページ]
すべてのテンポラリ/永続ミューテックスおよびセマフォに関する情報を返します (各ミューテックスを保持している接続や、セマフォが待機しているかどうかなど)。

sp_list_secure_feature_key システムプロシージャ [1693 ページ]
定義されたセキュリティ保護済み機能キーのリストを返します。

sp_login_environment システムプロシージャ [1694 ページ]
ユーザがログインするときの接続オプションを設定します。

sp_move_directory システムプロシージャ [1695 ページ]
この関数は、source_path がポイントするディレクトリを、destination_path がポイントする宛先に移動します。

sp_move_file システムプロシージャ [1696 ページ]
指定されたファイルをコンピュータ上の新しいディレクトリに移動します。

sp_objectpermission システムプロシージャ [1698 ページ]
指定されたロールまたはユーザ ID に付与されているオブジェクト権限のレポート、または指定されたオブジェクトまたは DB 領域に付与されているオブジェクト権限のレポートを生成します。

sa_parse_json システムプロシージャ [1701 ページ]
SQL データ型を使用した JSON データの表現を返します。

sp_plancache_contents システムプロシージャ [1702 ページ]
接続用のプランキャッシュのコンテンツを返します。これには、平均実行時間、最小実行時間、最大実行時間、文のカーソルが構築された回数を示す統計情報が含まれます。

sp_proc_priv システムプロシージャ [1705 ページ]
システムプロシージャの実行に必要なシステム権限のリストを返します。

sp_property_history システムプロシージャ [1707 ページ]
データベースによって追跡されたすべてのデータベースサーバプロパティの値を返します。

sp_remote_columns システムプロシージャ [1710 ページ]
リモートテーブルにあるカラムとそれらのデータ型の記述のリストを生成します。

sp_read_db_pages システムプロシージャ [1712 ページ]
指定されたページをキャッシュに読み込みます。

sp_read_etd システムプロシージャ [1714 ページ]
指定されたイベントトレースデータ (ETD) ファイルを読み込み、ファイルのコンテンツをローのセットとして返します。

sp_remote_exported_keys システムプロシージャ [1717 ページ]
指定されたプライマリテーブルに外部キーを持つテーブルに関する情報を表示します。

sp_remote_imported_keys システムプロシージャ [1719 ページ]
指定された外部キーに対応するプライマリキーを持つリモートテーブルに関する情報を提供します。

sp_remote_pcols システムプロシージャ [1721 ページ]
リモートテーブルにあるカラムとそれらのデータ型の記述のリストを生成します。

sp_remote_primary_keys システムプロシージャ [1723 ページ]
リモートデータアクセスを使用してリモートテーブルについてのプライマリキー情報を提供します。

sp_remote_procedures システムプロシージャ [1725 ページ]
リモートサーバ上のプロシージャのリストを返します。

sp_remote_tables システムプロシージャ [1727 ページ]
サーバ上のテーブルのリストを返します。

sp_servercaps システムプロシージャ [1729 ページ]
リモートサーバの機能についての情報を表示します。

sp_start_listener システムプロシージャ [1731 ページ]
新しい接続リスナーを起動します。

sp_stop_listener システムプロシージャ [1733 ページ]
既存の接続リスナーを停止します。

sp_sys_priv_role_info システムプロシージャ [1734 ページ]
システム権限とシステムロール間の 1 対 1 のマッピングを返します。

sp_top_k_statements システムプロシージャ [1736 ページ]
最大ランタイムを持つ文とプランの組み合わせを、上位から指定した数だけ返します。

sp_trace_event_fields システムプロシージャ [1739 ページ]
指定されたトレースイベントのフィールドに関する情報を返します。

sp_trace_event_session_events システムプロシージャ [1740 ページ]
特定のトレースイベントセッションの一部であるトレースイベントをリストします。

sp_trace_event_session_target_options システムプロシージャ [1742 ページ]
トレースイベントセッションのターゲットオプションをリストします。

sp_trace_event_session_targets システムプロシージャ [1744 ページ]
トレースセッションのターゲットをリストします。

sp_trace_event_sessions システムプロシージャ [1746 ページ]
データベースで定義されているトレースイベントセッションのリストを返します。

sp_trace_events システムプロシージャ [1747 ページ]
データベース内のトレースイベントに関する情報を返します。

sp_tsql_environment システムプロシージャ [1749 ページ]
ユーザが jConnect または Open Client アプリケーションから接続するときの接続オプションを設定します。

sp_use_secure_feature_key システムプロシージャ [1751 ページ]
現在の接続で指定されている保護された機能キーに含まれる機能を有効にします。

st_geometry_dump システムプロシージャ [1752 ページ]
ジオメトリを最低レベルのコンポーネントジオメトリに分解します。

st_geometry_load_shapefile システムプロシージャ [1757 ページ]
テーブルを作成し、そこに ESRI シェイプファイルをロードします。

xp_cmdshell システムプロシージャ [1759 ページ]
プロシージャからオペレーティングシステムコマンドを実行します。

xp_get_mail_error_code システムプロシージャ [1761 ページ]
最後の SMTP エラーまたは MAPI エラーに関する情報を返します。

xp_get_mail_error_text システムプロシージャ [1762 ページ]
最後の SMTP エラーまたはステータスメッセージテキストを返します。

xp_getenv システムプロシージャ [1764 ページ]
環境変数の値を返します。

xp_msver システムプロシージャ [1765 ページ]
データベースサーバのバージョンと名前についての情報を取り出します。

xp_read_file システムプロシージャ [1767 ページ]
ファイルを読み込み、そのファイルの内容を LONG BINARY 変数として返します。

xp_scanf システムプロシージャ [1768 ページ]
入力文字列とフォーマット文字列から部分文字列を抽出します。

xp_sendmail システムプロシージャ [1770 ページ]
xp_startmail または xp_startsmtp でセッションが開始されると、指定された受信者に電子メールメッセージを送信します。このプロシージャには、任意の長さのメッセージを指定できます。

xp_sprintf システムプロシージャ [1774 ページ]
入力文字列セットから結果文字列を構築します。

xp_startmail システムプロシージャ [1775 ページ]
MAPI で電子メールセッションを開始します。

xp_startsmtp システムプロシージャ [1777 ページ]
SMTP で電子メールセッションを開始します。

xp_stopmail システムプロシージャ [1780 ページ]
MAPI 電子メールセッションを閉じます。

xp_stopsmtp システムプロシージャ [1781 ページ]
SMTP 電子メールセッションを閉じます。

xp_write_file システムプロシージャ [1782 ページ]
SQL 文からファイルにデータを書き込みます。

1.6.8.1 sa_ansi_standard_packages システムプロシージャ

SQL 文で使用されている、コア以外の SQL 拡張機能に関する情報を返します。

構文

```
sa_ansi_standard_packages(
standard
, statement
)
```

パラメータ

standard

コア拡張機能で使用する標準を指定する LONG VARCHAR パラメータです。SQL:1999 または SQL:2003。

statement

評価する SQL 文を指定する LONG VARCHAR パラメータです。

結果セット

カラム名	データ型	説明
package_id	VARCHAR(10)	機能識別子。
package_name	LONG VARCHAR	機能名。

備考

文に使用される、コア以外の拡張機能がない場合、結果セットは空です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次に、sa_ansi_standard_packages システムプロシージャを呼び出す例を示します。

```
CALL sa_ansi_standard_packages( 'SQL:2003',
'SELECT *
   FROM ( SELECT o.SalesRepresentative,
   o.Region,
                SUM( s.Quantity * p.UnitPrice ) AS total_sales,
                DENSE_RANK() OVER ( PARTITION BY o.Region,
                                    GROUPING( o.SalesRepresentative )
                                    ORDER BY total_sales DESC ) AS sales_rank
            FROM Product p, SalesOrderItems s, SalesOrders o
            WHERE p.ID = s.ProductID AND s.ID = o.ID
            GROUP BY GROUPING SETS( ( o.SalesRepresentative, o.Region ),
o.Region ) ) AS DT
   WHERE sales_rank <= 3
   ORDER BY Region, sales_rank');
```

この例では、次の結果セットが生成されます。

package_id	package_name
T612	Advanced OLAP operations
T611	Elementary OLAP operations
F591	Derived tables
T431	Extended grouping capabilities

関連情報

SQLFLAGGER 関数 [その他] [537 ページ]

1.6.8.2 sa_audit_string システムプロシージャ

文字列をトランザクションログに追加します。

構文

```
sa_audit_string( string )
```

パラメータ

string

トランザクションログに追加する VARCHAR(128) 文字列。

備考

このシステムプロシージャでは、監査が有効な場合、トランザクションログに格納されている監査情報にコメントが追加されます。この文字列は最大 128 文字までです。

権限

MANAGE AUDITING システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、sa_audit_string を使用してコメントをトランザクションログに追加します。

```
CALL sa_audit_string( 'Auditing test' );
```

1.6.8.3 sa_certificate_info システムプロシージャ

データベースに保存される指定の証明書に関する情報を表示します。

構文

```
sa_certificate_info( cert_name )
```

パラメータ

cert_name

CREATE CERTIFICATE 文で使用した CHAR(128) 証明書名。

結果セット

カラム名	データ型	説明
name	CHAR(128)	属性の名前。
value	LONG VARCHAR	属性の値。

備考

cert_name が NULL の場合または ISYSCERTIFICATE テーブルにその名前の証明書がない場合は、行が返されません。それ以外の場合は、常に次のキーが返されます。

name	value
Name	ISYSCERTIFICATE の cert_name カラムの値。
ID	ISYSCERTIFICATE の object_id カラムの値。

返される他の行には、通称、国コード、地方、組織などの証明書の属性に対応するキーがあります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、証明書 mycert に関する情報を返します。

```
CALL sa_certificate_info( 'mycert' );
```

関連情報

CREATE CERTIFICATE 文 [779 ページ]
SYSCERTIFICATE システムビュー [1795 ページ]

1.6.8.4 sa_char_terms システムプロシージャ

CHAR 文字列を単語に分割し、各単語とその位置を 1 つのローとして返します。

構文

```
sa_char_terms(
text
[, config_name
[, owner ] ]
)
```

パラメータ

text

解析する LONG VARCHAR 文字列。

config_name

このオプションの CHAR(128) パラメータを使用して、文字列の処理時に適用するテキスト設定オブジェクトを指定します。デフォルト値は 'default_char' です。

owner

オプションの CHAR(128) パラメータを使用して、テキスト設定オブジェクトの所有者を指定します。デフォルト値は NULL です。所有者が指定されない場合、または NULL が指定されている場合には、現在のユーザが使用されます。

備考

このシステムプロシージャを使用すると、テキスト設定オブジェクトの設定が適用されたときに文字列がどのように解釈されるかを確認できます。インデックス処理で、またはクエリ文字列から、どの単語が削除されるかを確かめる場合に便利です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の文は、デフォルトの CHAR テキスト設定オブジェクト default_char を使用して、CHAR 文字列 "It's a work-at-home day!" 内の単語を返します。

```
CALL sa_char_terms ('It's a work-at-home day!', 'default_char', 'sys');
```

term	position
It	1
s	2
a	3
work	4
at	5
home	6
day	7

関連情報

sa_nchar_terms システムプロシージャ [1574 ページ]

1.6.8.5 sa_check_commit システムプロシージャ

コミット前に未処理の参照整合性違反がないか確認します。

構文

```
sa_check_commit(
tname
, keyname
)
```

パラメータ

tname

参照整合性に現在違反しているローのあるテーブルの名前を含む VARCHAR(128) パラメータ

keyname

対応する外部キーインデックスの名前を含む VARCHAR(128) パラメータ

備考

データベースオプション wait_for_commit が ON に設定されている場合、または外部キーが CREATE TABLE 文の CHECK ON COMMIT 句を使用して定義されている場合、データベースを更新すると、変更がコミットされる前に違反が解決されていれば、参照整合性違反が発生します。

変更をコミットする前に、sa_check_commit システムプロシージャを使用して未処理の参照整合性違反があるかどうかを確認できます。

返されたパラメータは、現在、参照整合性に違反しているローが含まれているテーブルの名前と、対応する外部キーインデックスを表しています。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の一連の文を Interactive SQL から実行できます。サンプルデータベースの Departments テーブルからローが削除され、参照整合性違反が発生します。sa_check_commit システムプロシージャを呼び出して、どのテーブルとキーに未処理の違反があるかを確認します。ROLLBACK で変更をキャンセルします。

```
SET TEMPORARY OPTION wait_for_commit='On';
DELETE FROM Departments;
CREATE OR REPLACE VARIABLE tname VARCHAR( 128 );
CREATE OR REPLACE VARIABLE keyname VARCHAR( 128 );
CALL sa_check_commit( tname, keyname );
SELECT tname, keyname;
ROLLBACK;
SET TEMPORARY OPTION wait_for_commit='Off';
```

関連情報

CREATE TABLE 文 [952 ページ]

1.6.8.6 sa_clean_database システムプロシージャ

データベースクリーナーを起動し、実行できる期間の最大値を設定します。

構文

```
sa_clean_database( [ duration ] )
```

パラメータ

duration

任意の UNSIGNED INTEGER パラメータを使用して、クリーン操作の実行が許可される時間 (秒単位) を指定します。デフォルトは 0 で、クリーナーの実行期間に制限がないと解釈されます。

備考

データベースクリーナーは、デフォルトのスケジュールで実行される内部タスクです。このシステムプロシージャを使用すると、データベースクリーナーを強制的にただちに実行したり、クリーナーの起動のたびにその実行時間を指定したりできます。期間が 0 の場合は、すべての DB 領域のすべてのページがクリーンアップされるまでデータベースクリーナーが実行されます。

データベースの検証中にこのシステムプロシージャを使用してデータベースクリーナーを起動した場合、検証が完了するまでデータベースクリーナーは実行されません。

要求の一部を後で実行すると、スナップショットアイソレーショントランザクション、インデックス管理、ローの削除などのデータベースタスクをより効率的に実行できます。このような後回しにできるアクティビティには、一般的に、削除済みエントリ、履歴エントリなどの不要なエントリをデータベースページから削除することによるクリーンアップや、アクセス効率を上げるためのデータベースページの再編成などがあります。

こうしたアクティビティを後で実行することにより、現在の要求がより高速に処理されるだけでなく、データベースサーバの負荷が軽いときにクリーンアップを実行する余地が生まれます。不要なエントリが区別されて、他のトランザクションから認識できなくなりますが、ページ上の領域は使用しているので、いつかは削除する必要があります。

データベースクリーナーは、後回しにされたクリーンアップアクティビティを実行します。データベースクリーナーは、20 秒ごとに実行されるようスケジュールされています。起動されると、データベースを DB 領域全体にわたって順次読み出し、クリーンアップ可能なページを検証してクリーニングし、次のページへ移ります。データベースクリーナーは、データベースサーバによって自動的に起動された場合は、セルフチューニングプロセスになります。データベースクリーナーによる作業量と実行時間は、DB 領域内の未処理のクリーンアップ可能ページの割合、データベースサーバの現在のアクティビティ量、データベースクリーナーがすでにクリーニングに費やした時間など、いくつかの要因に左右されます。0.5 秒間実行した後に、クリーナーがサーバでアクティブな要求を検出した場合、クリーナーは停止し、通常の間隔で実行するよう自らをスケジュールし直します。データベースクリーナーは、サーバで実行されている要求が他にないときにページ処理を試行することで、サーバの休止時間を活用します。

データベースクリーナーの統計情報は、次の 4 つのデータベースプロパティから取得できます。

CleanablePagesAdded

クリーンアップが必要なページ数を返します。

CleanablePagesCleaned

クリーンアップ済みのページ数を返します。

CleanableRowsAdded

クリーンアップが必要なロー数を返します。

CleanableRowsCleaned

クリーンアップ済みのロー数を返します。

CleanablePagesAdded 値と CleanablePagesCleaned 値の差は、クリーニングが必要な残りのデータベースページ数を示します。

sa_clean_database システムプロシージャを使用すると、データベースクリーナーをデータベース内のすべてのページがクリーンアップされるまで実行したり、データベースクリーナーの実行時間の上限を指定したりできます。

データベースクリーナーの動作をさらにカスタマイズするには、クリーンアップが必要なページ数やロー数が指定したしきい値を超えたらデータベースクリーナーを起動するイベントを設定します。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例では、データベースクリーナーの実行時間を 10 秒に設定します。

```
CALL sa_clean_database( 10 );
```

次の例では、データベースクリーナーを毎日実行してデータベース内のすべてのページをクリーンアップするようスケジュールされたイベントを作成します。

```
CREATE EVENT DailyDatabaseCleanup
SCHEDULE
   START TIME '6:00 pm'
   ON ( 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday' )
   HANDLER
      BEGIN
         CALL sa_clean_database( );
      END;
```

次の例では、データベース内のページの 20% 以上でクリーンアップが必要になるとデータベースクリーナーを強制的に実行します。

```
CREATE EVENT PeriodicCleaner
SCHEDULE
```

```
BETWEEN '9:00 am' and '5:00 pm'
EVERY 1 HOURS
HANDLER
BEGIN
     DECLARE @num_db_pages INTEGER;
     DECLARE @num_dirty_pages INTEGER;
     -- Get the number of database pages
     SELECT (SUM( DB_EXTENDED_PROPERTY( 'FileSize', t.dbspace_id ) -
                    DB_EXTENDED_PROPERTY( 'FreePages', t.dbspace_id ) ))
     INTO @num_db_pages
     FROM (SELECT dbspace_id FROM SYSDBSPACE) AS t;
     -- Get the number of pages to be cleaned
     SELECT (DB_PROPERTY( 'CleanablePagesAdded' ) -
                    DB_PROPERTY( 'CleanablePagesCleaned' ))
     INTO @num_dirty_pages;
     -- Check whether the number of dirty pages exceeds 20% of
     -- the size of the database
     IF @num_dirty_pages > @num_db_pages * 0.20 THEN
       -- Start cleaning the database for a maximum of 60 seconds
       CALL sa_clean_database( 60 );
     END IF;
END;
```

関連情報

CREATE EVENT 文 [805 ページ]

1.6.8.7 sa_column_stats システムプロシージャ

指定したカラムについて多様な統計情報を返します。この統計情報は、オプティマイザが使用するために管理されているカラムの統計情報とは関係がありません。

構文

```
sa_column_stats(
[ tab_name
[, col_name
[, tab_owner
[, max_rows ] ] ] ]
)
```

パラメータ

tab_name

任意の CHAR(128) パラメータが、テーブルの名前を指定します。このパラメータが指定されていない場合、すべてのテーブルのすべてのカラムについて統計情報が計算されます。デフォルトでは "%" です。

col_name

任意の CHAR(128) パラメータが、統計情報を計算するカラムを指定します。このパラメータが指定されていない場合、指定したテーブルのすべてのカラムについて統計情報が計算されます。デフォルトでは "%" です。

tab_owner

任意の CHAR(128) パラメータが、テーブルの所有者を指定します。このパラメータが指定されていない場合、データベースサーバは、指定された `tab_name` と一致する最初のテーブルの所有者を使用します。デフォルトでは "%" です。

max_rows

任意の INTEGER パラメータが、計算に使用するロー数を指定します。デフォルトは 1000 です。0 を指定すると、データベースサーバはテーブル内のすべてのローに基づいて比率を計算します。

結果セット

table_owner、table_name、column_name という例外はありますが、文字列以外のカラムで結果セットのすべての値は NULL です。また、空のテーブルの場合、num_rows_processed と num_values_compressed は 0 ですが、その他の値はすべて NULL です。

カラム名	データ型	説明
table_owner	CHAR(128)	テーブルの所有者。
table_name	CHAR(128)	テーブル名。
column_name	CHAR(128)	カラムの名前。
num_rows_processed	INTEGER	統計情報を計算するときに読み取るローの総数。
num_values_compressed	INTEGER	圧縮されるカラムの値の数。カラムが圧縮されない場合、この値は 0 です。
avg_compression_ratio	DOUBLE	カラムの圧縮済み値の平均圧縮率 (サイズの縮小率で表示)。カラムが圧縮されない場合、この値は NULL です。
avg_length	DOUBLE	カラムのすべての NOT NULL 文字列の平均長。
stddev_length	DOUBLE	カラムのすべての NOT NULL 文字列の長さの標準偏差。
min_length	INTEGER	カラムのすべての NOT NULL 文字列の最短長。
max_length	INTEGER	カラムの文字列の最大長。
avg_uncompressed_length	DOUBLE	カラムの圧縮されていないすべての NOT NULL 文字列の平均長。
stddev_uncompressed_length	DOUBLE	カラムの圧縮されていないすべての NOT NULL 文字列の長さの標準偏差。
min_uncompressed_length	INTEGER	カラムの圧縮されていないすべての NOT NULL 文字列の最短長。

カラム名	データ型	説明
max_uncompressed_length	INTEGER	カラムの圧縮されていないすべての NOT NULL 文字列の最大長。

備考

データベースサーバは、指定した所有者、テーブル、カラムの各名前と一致するカラムを決定します。次に、各指定したカラムのデータについて統計情報を計算します。デフォルトで、データベースサーバはデータの先頭 1000 ローのみを使用します。

avg_compression_ratio の場合は、値を 100 以上にすることはできません。ただし、極めて圧縮しづらいデータが圧縮カラムに挿入された場合は (たとえば、圧縮済みデータなど)、0 未満になることがあります。値が高いほど、高い圧縮率であることを示します。たとえば、戻り値が 80 の場合、圧縮データのサイズは圧縮していないデータのサイズよりも 80% 小さくなっています。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、テーブルの所有者であるか、カラムの SELECT 権限を持っているか、または MONITOR または MANAGE ANY STATISTICS システム権限を持っている必要があります。

関連する動作

なし

例

次の例では、SELECT 文で sa_column_stats システムプロシージャを使用して、カラムの圧縮で最も圧縮されるデータベースのカラムを決定します。

```
SELECT * FROM sa_column_stats()
  WHERE num_values_compressed > 0
  ORDER BY avg_compression_ratio desc;
```

次の例では、前の例から選択を絞り込み、bsmith が所有するテーブルのみを検索します。

```
SELECT * FROM sa_column_stats( tab_owner='GROUPO' )
  WHERE num_values_compressed > 0
  ORDER BY avg_compression_ratio desc;
```

1.6.8.8 sa_conn_activity システムプロシージャ

サーバ上に示されている指定したデータベース接続に関して、最後に作成された SQL 文を返します。

構文

```
sa_conn_activity( [ connidparm ] )
```

パラメータ

connidparm

任意の INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。
Name	VARCHAR(255)	現在の接続の名前を返します。 テンポラリ接続名の場合、接続名の先頭に `INT:` が追加されます。
Userid	VARCHAR(255)	接続のユーザ ID を返します。
DBNumber	INTEGER	データベースの ID 番号を返します。
LastReqTime	VARCHAR(255)	指定された接続において最後の要求が開始された時刻を返します。このプロパティは、イベントなどの内部接続の場合は空の文字列を返すことがあります。
LastStatement	LONG VARCHAR	現在の接続で最後に準備された SQL 文を返します。

備考

`connidparm` が 0 未満の場合、現在の接続についての情報が返されます。`connidparm` が指定されていない場合または NULL の場合、データベースサーバ上で実行されているすべてのデータベースに対するすべての接続に関する情報が返されます。

sa_conn_activity システムプロシージャは、接続について最後に作成された SQL 文で構成される一連の結果を返します。sa_conn_activity を呼び出す前に、データベースサーバで文の記録をオンにします。文の記録をオンにするには、データベースサーバの起動時に -zl オプションを指定するか、次の文を実行します。

```
CALL sa_server_option('RememberLastStatement','ON');
```

このプロシージャは、データベースサーバがビジー状態のときに、各接続について作成された最後の SQL 文の情報を取得するのに便利です。この機能は、要求ロギングの代わりとして使用できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

すべての接続 ID のリストを取得するには、SERVER OPERATOR、MONITOR、または DROP CONNECTION システム権限も必要です。

関連する動作

なし。

例

次の文は、sa_conn_activity システムプロシージャを使用して、各接続に関して最後に作成された SQL 文を表示します。

```
CALL sa_conn_activity( );
```

Number	Name	Userid	DBNumber	...
1,949	SQL_DBC_117acc40	DBA	0	...
1,948	setup	User1	0	...
...	...	...	...	...

関連情報

sa_server_option システムプロシージャ [1606 ページ]

1.6.8.9 sa_conn_compression_info システムプロシージャ

通信の圧縮率を要約します。

構文

```
sa_conn_compression_info( [ connidparm ] )
```

パラメータ

connidparm

任意の INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
タイプ	VARCHAR(20)	この後に続く圧縮統計が 1 つの接続 (Connection) の圧縮統計か、サーバへの全接続 (Server) の圧縮統計かを識別する文字列を返します。
ConnNumber	INTEGER	接続 ID 番号を表す INTEGER を返します。Type が Server の場合は NULL が返されます。
Compression	VARCHAR(10)	この接続で通信圧縮が有効であるかどうかを示す値として、On または Off を返します。 Type が Server の場合は NULL が返され、Type が Connection の場合は ON/OFF が返されます。
TotalBytes	INTEGER	送受信された実際のバイト数の合計を示す INTEGER を返します。
TotalBytesUnComp	INTEGER	圧縮を無効にした場合に送受信されると推測されるバイト数を示す INTEGER を返します。
CompRate	NUMERIC(5,2)	全体的な圧縮率を示す NUMERIC(5,2) の値を返します。たとえば、0 は圧縮が実行されなかったことを意味します。値が 75 の場合、データの圧縮率は 75% であり、元のサイズの 1/4 に圧縮されていることを意味します。

カラム名	データ型	説明
CompRateSent	NUMERIC(5,2)	クライアントに送信されたデータの圧縮率を示す NUMERIC(5,2) の値を返します。
CompRateReceived	NUMERIC(5,2)	クライアントから受信したデータの圧縮率を示す NUMERIC(5,2) の値を返します。
TotalPackets	INTEGER	送受信された実際のパケット数の合計を示す INTEGER を返します。
TotalPacketsUnComp	INTEGER	圧縮を無効にした場合に送受信されると推測されるパケットの総数を示す INTEGER を返します。
CompPktRate	NUMERIC(5,2)	パケットの全体的な圧縮率を示す NUMERIC(5,2) の値を返します。
CompPktRateSent	NUMERIC(5,2)	クライアントに送信されたパケットの圧縮率を示す NUMERIC(5,2) の値を返します。
CompPktRateReceived	NUMERIC(5,2)	クライアントから受信したパケットの圧縮率を示す NUMERIC(5,2) の値を返します。

備考

`connidparm` が 0 未満の場合、現在の接続について、圧縮プロパティで構成される結果セットが返されます。`connidparm` が指定されていない場合または NULL の場合、データベースサーバ上で実行されているすべてのデータベースに対するすべての接続について、圧縮プロパティが返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

すべての接続 ID のリストを取得するには、SERVER OPERATOR、MONITOR、または DROP CONNECTION システム権限も必要です。

関連する動作

なし。

例

次の例では、sa_conn_compression_info システムプロシージャを使用して、サーバに対する全接続の圧縮プロパティをまとめた結果セットを返します。

```
CALL sa_conn_compression_info( );
```

Type	ConnNumber	Compression	TotalBytes	...
Connection	79	Off	7841	...
Server	(NULL)	(NULL)	2737761	...
...	...	...	...	...

1.6.8.10 sa_conn_info システムプロシージャ

接続プロパティ情報をレポートします。

構文

```
sa_conn_info( [ connidparm ] )
```

パラメータ

connidparm

接続 ID 番号を指定する任意の INTEGER パラメータです。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。
Name	VARCHAR(255)	現在の接続の識別子 (文字列) を返します。 テンポラリ接続名の場合、接続名の先頭に **INT:** が追加されます。
Userid	VARCHAR(255)	接続のユーザ ID を返します。
DBNumber	INTEGER	データベースの ID 番号を返します。
LastReqTime	VARCHAR(255)	指定された接続において最後の要求が開始された時刻を、データベースのタイムゾーンで返します。このプロパティは、イベントなどの内部接続の場合は空の文字列を返すことがあります。

カラム名	データ型	説明
ReqType	VARCHAR(255)	最後の要求のタイプを返します。接続が接続プールによってキャッシュされる場合、その ReqType 値は CONNECT_POOL_CACHE になります。
CommLink	VARCHAR(255)	接続の通信リンクを返します。これは、SQL Anywhere がサポートするネットワークプロトコルであり、同一マシン接続用の "ローカル" なものです。
NodeAddr	VARCHAR(255)	クライアント/サーバ接続のクライアントアドレスを返します。
ClientPort	INTEGER	クライアントの TCP/IP ポート番号を返します。接続の種類が TCP/IP でない場合は、0 を返します。
ServerPort	INTEGER	データベースサーバの TCP/IP ポート番号または 0 を返します。
BlockedOn	INTEGER	現在の接続がブロックされていない場合は 0 を返し、ブロックされている場合はロック競合によってブロックされた接続の接続番号を返します。
LockRowID	UNSIGNED BIGINT	ロックされたローの識別子を返します。 ローに関連付けられているロックで接続が待機していない場合 (つまり、ロックで待機していない場合、または関連付けられているローがないロックで待機している場合)、LockRowID は NULL です。
LockIndexID	INTEGER	ロックされたインデックスの識別子を返します。 LockTable のテーブルのすべてのインデックスにロックが関連付けられている場合、LockIndexID は -1 です。インデックスに関連付けられているロックで接続が待機していない場合 (つまり、ロックで待機していない場合、または関連付けられているインデックスがないロックで待機している場合)、LockIndexID は NULL です。

カラム名	データ型	説明
LockTable	VARCHAR(255)	接続が現在ロックを待機している場合、そのロックに関連付けられているテーブルの名前を返します。ロックがない場合、またはロックに関連付けられているオブジェクトがテーブルでない場合、LockTable 値は空の文字列です。
UncommitOps	INTEGER	コミットされていないオペレーションの数を返します。
ParentConnection	INTEGER	データベースオペレーション (データベースのバックアップや作成など) を実行するためにテンポラリ接続を作成した接続の接続 ID を返します。その他のタイプの接続について、このプロパティは NULL を返します。
LockObject	VARCHAR(255)	接続が待機しているロックに関連付けられたオブジェクトの名前を返します (ある場合)。オブジェクトがテーブルである場合、この値は LockTable 値と同じです。接続がロックを待機していない場合、LockObject は NULL です。
LockObjectType	CHAR(20)	接続が待機しているロックに関連付けられたオブジェクトの型を返します (ある場合)。接続がロックを待機していない場合、LockObjectType は NULL です。

備考

`connidparm` が 0 未満の場合、現在の接続について、接続プロパティで構成される結果セットが返されます。`connidparm` が指定されていない場合または NULL の場合、データベースサーバ上で実行されているすべてのデータベースに対するすべての接続について、接続プロパティが返されます。

ブロックの場合には、このプロシージャが返す BlockedOn 値によって、どのユーザがどのユーザによってブロックされているかを調べることができます。sa_locks システムプロシージャを使用して、ブロックされた接続で保持されているロックを表示できます。

これらプロパティに関する詳細については、次のような例を実行できます。

```
SELECT *, DB_NAME( DBNumber ),
   CONNECTION_PROPERTY( 'LastStatement', Number )
   FROM sa_conn_info( );
```

LockRowID の値は、sa_locks プロシージャの出力でロックを検索するときに使用できます。

LockIndexID の値は、sa_locks プロシージャの出力でロックを検索するときに使用できます。また、LockIndexID の値は、SYSIDX システムビューを使用して表示できる ISYSIDX システムテーブルのプライマリキーに対応します。

ロックにはそれぞれ関連付けられたテーブルがあるため、LockTable の値を使用して、ロックで接続が待機しているかどうかを明確に判断できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

すべての接続 ID のリストを取得するには、SERVER OPERATOR、MONITOR、または DROP CONNECTION システム権限も必要です。

関連する動作

なし。

例

次の例では、sa_conn_info システムプロシージャを使用して、サーバに対する全接続の接続プロパティをまとめた結果セットを返します。

```
CALL sa_conn_info( );
```

Number	Name	Userid	DBNumber	...
79	SQL_DBC_10dcf810	DBA	0	...
46	setup	User1	0	...
...	...	...	...	...

次の例では、sa_conn_info システムプロシージャを使用して、どの接続がテンポラリ接続を作成したかを示す結果セットを返します。

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

接続 8 がテンポラリ接続を作成し、そのテンポラリ接続によって CREATE DATABASE 文が実行されました。

```
Number      Name                ParentConnection
------------------------------------------------
1000000048  INT: CreateDB       8
9           SQL_DBC_14675af8    (NULL)
8           SQL_DBA_152d5ac0    (NULL)
```

次の例では、sa_conn_info システムを使用して、ブロックされた接続の数を返します。

```
SELECT COUNT(*) FROM sa_conn_info()
WHERE blockedOn = connection_property('number');
```

関連情報

sa_locks システムプロシージャ [1546 ページ]
SYSIDX システムビュー [1810 ページ]

1.6.8.11 sa_conn_list システムプロシージャ

接続 ID を含む結果セットを返します。

構文

```
sa_conn_list(
[ connidparm
[, dbidparm ] ]
)
```

パラメータ

connidparm

任意の INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは NULL です。

dbidparm

任意の INTEGER パラメータを使用して、データベース ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。

備考

`connidparm` が 0 より大きい場合、指定した接続についての情報が返されます。`connidparm` が 0 未満の場合、現在の接続についての情報が返されます。`connidparm` と `dbidparm` を指定しない場合、またはこれらのパラメータが NULL の場合、データベースサーバ上で実行されているすべてのデータベースに対するすべての接続の接続 ID が返されます。

`connidparm` が NULL で `dbidparm` が 0 以上の場合、そのデータベースの接続 ID のみが返されます。`connidparm` が NULL で `dbidparm` が 0 未満の場合、現在のデータベースの接続 ID のみが返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

すべての接続 ID のリストを取得するには、SERVER OPERATOR、MONITOR、または DROP CONNECTION システム権限も必要です。

関連する動作

なし。

例

次の例は、sa_conn_list システムプロシージャを使用して接続 ID のリストを表示します。

```
CALL sa_conn_list( );
```

Number
1,949
1,948
...

関連情報

sa_db_list システムプロシージャ [1481 ページ]
sa_conn_options システムプロシージャ [1469 ページ]

1.6.8.12 sa_conn_options システムプロシージャ

データベースオプションに対応する接続プロパティのプロパティ情報を返します。

構文

```
sa_conn_options( [ connidparm ] )
```

パラメータ

connidparm

任意の INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。
PropNum	INTEGER	接続プロパティの番号が返されます。
OptionName	VARCHAR(255)	オプション名を返します。
OptionDescription	VARCHAR(255)	オプションの説明を返します。
Value	LONG VARCHAR	オプションの値を返します。

備考

接続 ID を Number として返し、データベースオプションに対応する使用可能な各接続プロパティについて PropNum、OptionName、OptionDescription、Value を返します。

`connidparm` が 0 未満の場合、現在の接続のオプション値が返されます。`connidparm` が指定されていないか NULL のとき、必要なシステム権限を有している場合に限り、現在のデータベースに対するすべての接続のオプション値が返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

すべての接続に対するオプションを取得するには、SERVER OPERATOR、MONITOR、または DROP CONNECTION システム権限も必要です。これらのシステム権限のどれも有していない場合、現在の接続に対する結果のみが返されます。

関連する動作

なし。

例

次の例は、sa_conn_options システムプロシージャを使用して、データベースオプションに対応する接続プロパティのプロパティ情報を表示します。

```
SELECT * FROM sa_conn_options()
    ORDER BY OptionName;
```

Number	PropNum	OptionName	OptionDescription	Value
1,952	502	ブロッキング	ロック競合への応答を制御します。	On

Number	PropNum	OptionName	OptionDescription	Value
1,952	503	blocking_timeout	トランザクションがロックを獲得するまでの待機時間を制御します。	0
...	...	...	...	...

関連情報

sa_db_list システムプロシージャ [1481 ページ]
sa_conn_list システムプロシージャ [1468 ページ]

1.6.8.13 sa_conn_properties システムプロシージャ

接続プロパティ情報をレポートします。

構文

```
sa_conn_properties( [ connidparm ] )
```

パラメータ

connidparm

任意の INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。
PropNum	INTEGER	接続プロパティの番号が返されます。
PropName	VARCHAR(255)	接続プロパティの名前が返されます。
PropDescription	VARCHAR(255)	接続プロパティの説明が返されます。
Value	LONG VARCHAR	接続プロパティの値が返されます。

備考

接続 ID を Number として返し、使用可能な各接続プロパティについて PropNum、PropName、PropDescription、Value を返します。すべての接続プロパティ、接続に関するデータベースオプション設定、接続に関連する統計情報の値が返されます。NULL 値を持つ有効なプロパティも返されます。

`connidparm` が 0 未満の場合、現在の接続のプロパティ値が返されます。`connidparm` が指定されていないか NULL の場合、現在のデータベースに対するすべての接続のプロパティ値が返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

すべての接続 ID のリストを取得するには、SERVER OPERATOR、MONITOR、または DROP CONNECTION システム権限も必要です。

関連する動作

なし。

例

次の例では、sa_conn_properties システムプロシージャを使用して、全接続の接続プロパティ情報をまとめた結果セットを返します。

```
CALL sa_conn_properties( );
```

Number	PropNum	PropName	...
79	37	ClientStmtCacheHits	...
79	38	ClientStmtCacheMisses	...
...	...	...	...

次の例では、sa_conn_properties システムプロシージャを使用して、CPU 時間の降順で、すべての接続のリストを返します*。

```
SELECT Number AS connection_number,
    CONNECTION_PROPERTY ( 'Name', Number ) AS connection_name,
    CONNECTION_PROPERTY ( 'Userid', Number ) AS user_id,
  CAST ( Value AS NUMERIC ( 30, 2 ) ) AS approx_cpu_time
  FROM sa_conn_properties( )
  WHERE PropName = 'ApproximateCPUTime'
  ORDER BY approx_cpu_time DESC;
```

関連情報

システム関数 [215 ページ]
sa_conn_list システムプロシージャ [1468 ページ]
sa_conn_options システムプロシージャ [1469 ページ]

1.6.8.14 sa_convert_ml_progress_to_timestamp システムプロシージャ

Mobile Link のスクリプト化されたアップロードの場合のみ。スクリプト化されたアップロードの進行状況値を UNSIGNED BIGINT から TIMESTAMP に変換します。

構文

```
sa_convert_ml_progress_to_timestamp( progress )
```

パラメータ

progress

この UNSIGNED BIGINT パラメータを使用して、TIMESTAMP に変換する進捗値を指定します。

戻り値

この関数は、渡される値で表現する TIMESTAMP を返します。

備考

この関数は、sa_convert_timestamp_to_ml_progress の反対です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、UNSIGNED BIGINT 値をタイムスタンプ値 (2009-10-20 13:36:51.199) に変換します。

```
SELECT sa_convert_ml_progress_to_timestamp( 3465034611199 );
```

関連情報

sa_convert_timestamp_to_ml_progress システムプロシージャ [1474 ページ]

1.6.8.15 sa_convert_timestamp_to_ml_progress システムプロシージャ

Mobile Link のスクリプト化されたアップロードの場合のみ。スクリプト化されたアップロードの進行状況値を TIMESTAMP から UNSIGNED BIGINT に変換します。

構文

```
sa_convert_timestamp_to_ml_progress( t1 )
```

パラメータ

t1

この TIMESTAMP パラメータを使用して、UNSIGNED BIGINT に変換する進行状況値を指定します。

戻り値

この関数は、パラメータとして渡されるタイムスタンプを表す UNSIGNED BIGINT を返します。

備考

このプロシージャは、sa_convert_ml_progress_to_timestamp の反対です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、タイムスタンプ値を UNSIGNED BIGINT 値に変換します。

```
SELECT sa_convert_timestamp_to_ml_progress( CURRENT TIMESTAMP );
```

```
SELECT sa_convert_timestamp_to_ml_progress( '2009-10-20 13:36:51.199' );
```

関連情報

sa_convert_ml_progress_to_timestamp システムプロシージャ [1473 ページ]

1.6.8.16 sa_copy_cursor_to_temp_table システムプロシージャ

テンポラリテーブルを作成し、そのテーブルにオープンカーソルの結果セットをコピーします。

構文

```
sa_copy_cursor_to_temp_table(
cursor_name
, table_name
[, first_row
[, max_rows ] ]
)
```

パラメータ

cursor_name

この VARCHAR(256) パラメータを使用して、オープンカーソルの名前を指定します。

table_name

この VARCHAR(256) パラメータを使用して、テンポラリテーブルの名前を指定します。

first_row

この BIGINT パラメータを使用して、テンポラリテーブルにコピーする最初の行の番号を指定します。デフォルトは 1 です。

max_rows

この BIGINT パラメータを使用して、テンポラリテーブルにコピーする行の最大数を指定します。デフォルトは 9223372036854775807 (すべてのロー) です。

備考

いくつかの整数カラムのカーソルが存在している場合を想定します。sa_copy_cursor_to_temp_table では、次の形式の文を使用して、テンポラリテーブルを作成します。

```
BEGIN
  CREATE LOCAL TEMPORARY TABLE TempTab (
      col1 INT,
      col2 INT,
  ...
rownum bigint primary key )
END;
```

sa_copy_cursor_to_temp_table では、カラムに col1、col2 などの名前を付け、名前の重複や、カーソルカラムの名前が適切に定義されていない場合 (たとえば、複雑な式の場合) の問題を回避します。

テンポラリテーブルが作成されると、オープンカーソルの内容が挿入されます。このとき、first_row に指定したロー番号まで移動し、max_rows に指定した数のローが挿入されます。テンポラリテーブルに内容が挿入された後、カーソルは元のロケーションに再配置されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

カーソルからコピーすると、カーソルの独立性設定を使用してローがフェッチされます。これにより、ローがロックされ、カーソルからフェッチする場合と同じ、他の影響が発生する場合があります。

現在の接続の外部で同時に変更が加えられ、カーソルが実体化または独立性設定によって保護されていない場合、プロシージャが完了した後でカーソルが別のローに配置される可能性があります。たとえば、直前に最新のローとなっていたローが削除された場合は、カーソルが元の位置の後のローに再配置されることがあります。

カーソルからのコピー中にエラーが発生した場合、カーソルは無効な状態になり、カーソルに対するそれ以降の操作はエラーで失敗します。

例

次のバッチでは、myCursor というカーソルを作成し、Products テーブルからデータをロードします。その後、カーソルが開かれます (OPEN 文)。DROP 文では、myTempTable (すでに存在する場合) を削除します。sa_copy_cursor_to_temp_table を呼び出すと、myTempTable というテンポラリテーブルが作成され、そのテーブルに myCursor の内容がコピーされます。最後に、SELECT 文によって、カーソルからテンポラリテーブルにコピーされたデータが返されます。

```
BEGIN
  DECLARE myCursor CURSOR FOR
      SELECT ID, Name, Description, Color, Quantity FROM Products;
  OPEN myCursor;
  DROP TABLE IF EXISTS myTempTable;
  CALL sa_copy_cursor_to_temp_table( 'myCursor', 'myTempTable' );
  CLOSE myCursor;
  SELECT * FROM myTempTable;
END
```

関連情報

sa_list_cursors システムプロシージャ [1541 ページ]
sa_describe_cursor システムプロシージャ [1488 ページ]

1.6.8.17 sa_cpu_topology システムプロシージャ

データベースサーバを実行しているコンピュータのプロセッサのトポロジを返します。

構文

sa_cpu_topology( )

結果セット

カラム名	データ型	説明
os_id	UNSIGNED INTEGER	基盤となっているオペレーティングシステムが使用する論理プロセッサを返します。たとえば、Windows ではこの値は Windows タスクマネージャーに表示されるプロセッサ ID と一致します。
socket	UNSIGNED INTEGER	CPU ソケットまたはパッケージ識別子を返します。
core	UNSIGNED INTEGER	特定のソケット内のコアの識別子を返します。
thread	UNSIGNED INTEGER	特定のソケットとコア内のスレッドの識別子を返します。
apic	UNSIGNED INTEGER	システム内の論理プロセッサの識別子を返します。ソケット、コア、スレッドのエンコードが含まれます。
"group"	UNSIGNED INTEGER	Windows でオペレーティングシステムによって割り当てられたプロセッサグループ番号を返します。Solaris では、プロセッサセット識別子です。それ以外の場合は 0 です。
numa_node	UNSIGNED INTEGER	Windows でオペレーティングシステムによって割り当てられた NUMA ノードを返します。
online	BIT	プロセッサがオンラインの場合のみ 1 を返します。
in_use	BIT	データベースサーバのプロセスとのアフィニティマスクで、現在の行によって記述されている論理プロセッサがオンラインおよび有効な場合、1 を返します。それ以外の場合は 0 を返します。
user_selected	BIT	-gta データベースサーバオプションまたは ProcessorAffinity システムプロシージャを使用して指定された物理プロセッサを指定します。

備考

32 ビットおよび 64 ビットの Intel x86/x64 プラットフォームでは、このプロシージャにより返される情報に、基盤となるハードウェアが正確に記述されます。その他のすべてのプラットフォームで、システムで論理プロセッサあたり 1 つの行がありますが、カラム値は次のようになります。

- os_id、socket、apic は等しい
- core、thread、numa_node はゼロ
- group はゼロ (この値が pset 識別子である Solaris を除く)

一部の仮想マシンでは、結果が正確でない場合があります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、データベースサーバが実行されているコンピュータのプロセッサトポロジが含まれる結果セットを返します。

```
CALL sa_cpu_topology( );
```

関連情報

sa_server_option システムプロシージャ [1606 ページ]

1.6.8.18 sa_db_info システムプロシージャ

データベースのプロパティ情報をレポートします。

構文

```
sa_db_info( [ dbidparm ] )
```

パラメータ

dbidparm

任意の INTEGER パラメータを使用して、データベース ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。
Alias	VARCHAR(255)	データベース名を返します。
File	VARCHAR(255)	パスを含むデータベースルートファイル名を返します。
ConnCount	INTEGER	データベースとの接続の数を返します。プロパティ値には、内部処理に使用されている接続は含まれませんが、イベントと外部環境サポートに使用されている接続は含まれます。
PageSize	INTEGER	データベースのページサイズ (バイト単位) を返します。
LogName	VARCHAR(255)	パスを含むトランザクションログファイル名を返します。

備考

データベース ID を指定した場合、sa_db_info は、指定したデータベースの Number、Alias、File、ConnCount、PageSize、LogName を持つ 1 つのローを返します。

`dbidparm` が 0 より大きい場合、指定したデータベースのプロパティが返されます。`dbidparm` が 0 より小さい場合、現在のデータベースのプロパティが返されます。`dbidparm` を指定しない場合または NULL の場合、データベースサーバで実行されているすべてのデータベースのプロパティが返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

他のデータベースに対してこのシステムプロシージャを実行する場合は、SERVER OPERATOR または MONITOR のシステム権限も必要です。

関連する動作

なし

例

次の文は、サーバで実行されている各データベースのローを返します。

```
CALL sa_db_info( );
```

関連情報

sa_db_properties システムプロシージャ [1485 ページ]

1.6.8.19 sa_db_list システムプロシージャ

データベース ID を返します。

構文

```
sa_db_list( [ dbidparm ] )
```

パラメータ

dbidparm

任意の INTEGER パラメータを使用して、データベース ID 番号を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	データベース ID 番号。

備考

dbidparm が 0 より大きい場合、指定したデータベースの ID が返されます。dbidparm が 0 より小さい場合、現在のデータベースの ID が返されます。dbidparm を指定しない場合または NULL の場合、データベースサーバで実行されているすべてのデータベースの ID が返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

他のデータベースに対してこのシステムプロシージャを実行する場合は、SERVER OPERATOR または MONITOR のシステム権限も必要です。

関連する動作

なし

例

次の例は、データベースサーバで実行中のデータベースの数を返します。

```
SELECT count(*) FROM sa_db_list();
```

次の例は、システムプロシージャ sa_db_list と関数 DB_Name を使用して、データベースサーバ上で実行中のデータベースのリストを返します。

```
SELECT DB_NAME(Number) FROM sa_db_list();
```

関連情報

sa_conn_list システムプロシージャ [1468 ページ]
sa_conn_options システムプロシージャ [1469 ページ]
DB_NAME 関数 [システム] [322 ページ]

1.6.8.20 sa_db_option システムプロシージャ

データベースの実行中に、データベースオプションを上書きします。

構文

```
sa_db_option(
opt
, val
)
```

パラメータ

opt

この CHAR(128) パラメータを使用して、データベースオプション名を指定します。

val

この LONG VARCHAR パラメータを使用して、データベースオプションの新しい値を指定します。

備考

データベース管理者はこのプロシージャを使用して、データベースを再起動しないでデータベースオプションの一部を一時的に上書きできます。

このプロシージャを使用して変更されるオプション値は、データベースが停止するとデフォルト値にリセットされます。データベースを起動するたびにオプション値を変更する場合は、データベースの起動時に対応するデータベースオプションを指定します (存在する場合)。

次のオプション設定を変更できます。

オプション名	値	システム権限	その他の情報
DiskSandbox	ON、**OFF**	SERVER OPERATOR	DiskSandbox を有効にすると、データベースファイルの読み込み/書き込み操作が、メインデータベースファイルがあるディレクトリとそのサブディレクトリに制限されます。sa_db_option システムプロシージャを使用してディスクのサンドボックス設定を変更する場合は、manage_disk_sandbox 機能に対してセキュリティ保護済み機能キーを指定する必要があります。

オプション名	値	システム権限	その他の情報
PropertyHistoryList	データベースサーバプロパティのカンマ区切りのリスト	MANAGE ANY PROPERTY HISTORY	PropertyHistoryList が有効な場合、プロパティのデフォルトリストが選択されます。追跡するデータベースサーバプロパティのカンマ区切りのリストを指定することができます。デフォルトは、空のリストです。 このオプションが設定されていない場合は、データベースサーバまたは同じデータベース上の他のデータベースによって追跡されたプロパティだけがクエリされます。 すべての指定されたプロパティを追跡する必要があるメモリが、PropertyHistorySize データベースサーバオプションまたは使用可能なメモリによって設定された制限を超えている場合、エラーが発生し、データベースのプロパティリストは更されません。
CollectStmtPerfStats	**ON**、OFF	MONITOR	CollectStmtPerfStats が ON に設定されている場合、ステートメントパフォーマンスサマリ情報が収集されます。OFF に設定されると、データ収集が停止され、これまで収集されたデータが破棄されます。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例が機能するには、データベースサーバがオプション `-sk securefkey` で起動されていることが必要です。

この例は、MANAGE_KEYS を含むセキュリティ保護済み SYSTEM 機能セットを有効にして、大文字小文字が区別される認証コードである NewSecurityCode を持つ SECURITY と呼ばれるセキュリティ保護済み機能キーを新たに作成し、その新しいセキュリティ保護済み機能キーを使用して DiskSandbox オプションを有効にします。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'security', 'NewSecurityCode',
'manage_security' );
CALL sp_use_secure_feature_key( 'security', 'NewSecurityCode' );
CALL sa_db_option( 'DiskSandbox', 'on' );
```

関連情報

sa_server_option システムプロシージャ [1606 ページ]

1.6.8.21 sa_db_properties システムプロシージャ

データベースのプロパティ情報をレポートします。

構文

```
sa_db_properties( [ dbidparm ] )
```

パラメータ

dbidparm

データベース ID 番号を指定する任意の INTEGER パラメータです。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
Number	INTEGER	データベース ID 番号。
PropNum	INTEGER	データベースプロパティの番号。
PropName	VARCHAR(255)	データベースプロパティ名。
PropDescription	VARCHAR(255)	データベースプロパティの説明。
Value	LONG VARCHAR	データベースプロパティの値。

備考

データベース ID を指定した場合、sa_db_properties システムプロシージャはデータベース ID 番号と、使用可能な各データベースプロパティの PropNum、PropName、PropDescription、Value を返します。データベースに関するすべてのデータベースプロパティと統計情報の値が返されます。NULL 値を持つ有効なプロパティも返されます。

`dbidparm` が 0 より大きい場合、指定したデータベースのデータベースプロパティが返されます。`dbidparm` が 0 より小さい場合、現在のデータベースのデータベースプロパティが返されます。`dbidparm` を指定しない場合または NULL の場合、データベースサーバで実行されているすべてのデータベースのデータベースプロパティが返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

他のデータベースに対してこのシステムプロシージャを実行する場合は、SERVER OPERATOR または MONITOR のシステム権限も必要です。

関連する動作

なし

例

次の例では、sa_db_properties システムプロシージャを使用して、呼び出し側がシステム権限 SERVER OPERATOR または MONITOR を持っている場合のすべてのデータベースのデータベースプロパティをまとめた結果セットを返します。それ以外の場合は、現在のデータベースのデータベースプロパティが返されます。

```
CALL sa_db_properties( );
```

Number	PropNum	PropName	...
0	0	ConnCount	...
0	1	IdleCheck	...
0	2	IdleWrite	...
...	...	...	...

次の例では、sa_db_properties() システムプロシージャを使用して、2 番目のデータベースのデータベースプロパティをまとめた結果セットを返します。

```
CALL sa_db_properties( 1 );
```

関連情報

sa_db_info システムプロシージャ [1479 ページ]

1.6.8.22 sa_dependent_views システムプロシージャ

指定したテーブルまたはビューのすべての従属ビューリストを返します。

構文

```
sa_dependent_views(
[ tbl_name
[, owner_name ] ]
)
```

パラメータ

tbl_name

任意の CHAR(128) パラメータを使用して、テーブルまたはビューの名前を指定します。デフォルトは NULL です。

owner_name

任意の CHAR(128) パラメータを使用して、tbl_name の所有者を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
table_id	UNSIGNED INTEGER	テーブルまたはビューのオブジェクト ID。
dep_view_id	UNSIGNED INTEGER	従属ビューのオブジェクト ID。

備考

このプロシージャを使用して、テーブルとその従属ビューの ID リストを取得します。

テーブル名と所有者名について指定された条件を満たす既存のテーブルがない場合、エラーは発生しません。この他にも次の制約が適用されます。

- owner と tbl_name がどちらも NULL の場合は、従属ビューを持つすべてのテーブルに関する情報が返されます。
- tbl_name が NULL で owner が指定されている場合は、指定された所有者が所有するすべてのテーブルに関する情報が返されます。

- tbl_name が指定されていて owner が NULL の場合は、指定された名前を持ついずれかのテーブルに関する情報が返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、sa_dependent_views システムプロシージャを使用して、SalesOrders テーブルに依存するビューの ID リストを取得します。このプロシージャは、SalesOrders に対して table_id を、従属ビュー ViewSalesOrders に対して dep_view_id を返します。

```
CALL sa_dependent_views( 'SalesOrders' );
```

次の例では、sa_dependent_views システムプロシージャを SELECT 文で使用して、SalesOrders テーブルに依存するビューの名前リストを取得します。このプロシージャは、ViewSalesOrders ビューを返します。

```
SELECT t.table_name FROM SYSTAB t,
sa_dependent_views( 'SalesOrders' ) v
WHERE t.table_id = v.dep_view_id;
```

関連情報

SYSDEPENDENCY システムビュー [1799 ページ]

1.6.8.23 sa_describe_cursor システムプロシージャ

カーソルのカラムの名前と型に関する情報を記述します。

構文

```
sa_describe_cursor( cursor_name )
```

パラメータ

cursor_name

記述するオープンカーソルを識別する VARCHAR(256) 値。

結果セット

カラム名	データ型	説明
column_number	INTEGER	このローが記述するカラムの順序位置 (開始値は 1)。
name	VARCHAR(128)	カラム名。
domain_id	SMALLINT	カラムのデータ型。
domain_name	VARCHAR(128)	カラムのデータ型名。
domain_name_with_size	VARCHAR(160)	サイズと精度を含むデータ型名 (CREATE TABLE 関数または CAST 関数で使用されます)。
width	INTEGER	文字列パラメータでは長さ、数値パラメータでは精度、その他のデータ型では記憶領域のサイズをバイトで示します。
scale	INTEGER	数値データ型カラムでは小数点以下の桁数、その他のデータ型では 0 を示します。
declared_width	INTEGER	文字列パラメータでは長さ、数値パラメータでは精度、その他のデータ型では記憶領域のサイズをバイトで示します。
user_type_id	SMALLINT	該当する場合はユーザ定義のデータ型、それ以外の場合は NULL。
user_type_name	VARCHAR(128)	該当する場合はユーザ定義のデータ型、それ以外の場合は NULL。
correlation_name	VARCHAR(128)	該当する場合は式と関連付けられている相関名、それ以外の場合は NULL。
base_table_id	UNSIGNED INTEGER	式がカラムの場合は table_id、それ以外の場合は NULL。
base_column_id	UNSIGNED INTEGER	式がカラムの場合は column_id、それ以外の場合は NULL。
base_owner_name	VARCHAR(128)	式がカラムの場合は所有者名、それ以外の場合は NULL。
base_table_name	VARCHAR(128)	式がカラムの場合はテーブル名、それ以外の場合は NULL。
base_column_name	VARCHAR(128)	式がカラムの場合はカラム名、それ以外の場合は NULL。

カラム名	データ型	説明
nulls_allowed	BIT	式を NULL (1) にできるかどうかを示すインジケータ。
is_autoincrement	BIT	式が AUTOINCREMENT カラム (1) かどうかを示すインジケータ。
is_key_column	BIT	式が結果セット (1) のキーの一部かどうかを示すインジケータ。詳細については、以下の「備考」の項を参照してください。
is_added_key_column	BIT	式が追加されたキーカラム (1) かどうかを示すインジケータ。詳細については、以下の「備考」の項を参照してください。

備考

sa_describe_cursor システムプロシージャは、API に依存しないメカニズムによって、カーソルによって返されたカラムの説明を取得します。このシステムプロシージャは、動的 SQL で動作するストアドプロシージャを作成する場合に役立ちます。

sa_describe_cursor システムプロシージャは、CALL 文、または SELECT 文の FROM 句で使用できます。

`cursor_name` は、現在の接続のオープンカーソルを示すようにしてください。接続のオープンカーソルのリストを取得するには、sa_list_cursors システムプロシージャを使用します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次のバッチは、Products テーブルでカーソル myCursor を作成し、それを開きます。sa_describe_cursor システムプロシージャは、カーソルのカラムを説明するのに使用します。各カラムに 5 つの行がそれぞれ含まれている結果セットが作成されます。

```
BEGIN
  DECLARE myCursor CURSOR FOR
      SELECT ID, Name, Description, Color, Quantity FROM Products;
  OPEN myCursor;
  CALL sa_describe_cursor( 'myCursor' );
  CLOSE myCursor;
END
```

関連情報

sa_list_cursors システムプロシージャ [1541 ページ]
sa_copy_cursor_to_temp_table システムプロシージャ [1475 ページ]
SYSDOMAIN システムビュー [1800 ページ]
SYSUSER システムビュー [1855 ページ]
SYSTABCOL システムビュー [1845 ページ]
SYSUSERTYPE システムビュー [1858 ページ]
SYSDOMAIN システムビュー [1800 ページ]

1.6.8.24 sa_describe_query システムプロシージャ

1 つのローにクエリの各出力カラムを記述したクエリの結果セットを記述します。

構文

```
sa_describe_query(
query
[, add_keys ]
)
```

パラメータ

query

この LONG VARCHAR パラメータを使用して、記述されている SQL 文のテキストを指定します。

add_keys

任意の BIT パラメータを使用して、記述されているクエリの結果セットでローをユニークに識別するカラムのセットを決定するかどうかを指定します。デフォルトは 0 です。データベースサーバはカラムの識別を試行しません。このパラメータの詳細については、以下の備考部分を参照してください。

結果セット

カラム名	データ型	説明
column_number	INTEGER	このローが記述するカラムの順序位置 (開始値は 1)。
name	VARCHAR(128)	カラム名。

カラム名	データ型	説明
domain_id	SMALLINT	カラムのデータ型。
domain_name	VARCHAR(128)	データ型の名前。
domain_name_with_size	VARCHAR(160)	サイズと精度を含むデータ型名 (CREATE TABLE 関数または CAST 関数で使用されます)。
width	INTEGER	文字列パラメータでは長さ、数値パラメータでは精度、その他のデータ型では記憶領域のサイズをバイトで示します。
scale	INTEGER	数値データ型カラムでは小数点以下の桁数、その他のデータ型では 0 を示します。
declared_width	INTEGER	文字列パラメータでは長さ、数値パラメータでは精度、その他のデータ型では記憶領域のサイズをバイトで示します。
user_type_id	SMALLINT	1 つの場合はユーザ定義のデータ型の type_id、それ以外の場合は NULL。
user_type_name	VARCHAR(128)	1 つの場合はユーザ定義のデータ型の name、それ以外の場合は NULL。
correlation_name	VARCHAR(128)	使用できる場合、式と関連付けられている相関名、それ以外の場合は NULL。
base_table_id	UNSIGNED INTEGER	式がカラムの場合は table_id、それ以外の場合は NULL。
base_column_id	UNSIGNED INTEGER	式がカラムの場合は column_id、それ以外の場合は NULL。
base_owner_name	VARCHAR(128)	式がカラムの場合は所有者名、それ以外の場合は NULL。
base_table_name	VARCHAR(128)	式がカラムの場合はテーブル名、それ以外の場合は NULL。
base_column_name	VARCHAR(128)	式がカラムの場合はカラム名、それ以外の場合は NULL。
nulls_allowed	BIT	式を NULL にできる場合は 1、それ以外の場合は 0 を示すインジケータ。
is_autoincrement	BIT	式が、AUTOINCREMENT になると宣言されているカラムの場合は 1、それ以外の場合は 0 を示すインジケータ。
is_key_column	BIT	式が結果セットのキーの一部である場合は 1、それ以外の場合は 0 になるインジケータ。詳細については、以下の備考部分を参照してください。
is_added_key_column	BIT	式が追加されたキーカラムである場合は 1、それ以外の場合は 0 になるインジケータ。詳細については、以下の備考部分を参照してください。

備考

sa_describe_query プロシージャは、API に依存しないメカニズムによって、クエリの結果セットに含まれる式に関する名前とタイプの情報を記述します。

add_keys に 1 が指定されている場合、sa_describe_query プロシージャはクエリ対象のオブジェクトからカラムセットを検索しようとします。このカラムセットは、記述されるクエリの結果セットのローをユニークに識別するときに使用されます。キーの形式は、クエリ対象のオブジェクトに含まれる 1 つ以上のカラムです。また、クエリで明示的に参照されないカラムがキーに含まれる場合があります。オプティマイザによってキーが見つかった場合、キーで使用されるカラムは、結果で値が 1 の is_key_column によって識別されます。キーが見つからない場合はエラーが返されます。

キーに含まれていてもクエリで明示的に参照されていないカラムについては、is_added_key_column の値が 1 に設定されます。これはプロシージャの結果にカラムが追加されていることを示します。それ以外の場合、is_added_key_column の値は 0 です。

add_keys を指定しない場合、または 0 の値を指定する場合、オプティマイザは結果セットのキーを検索しません。is_key_column と is_added_key_column には NULL が含まれます。

declared_width 値と width 値のどちらもカラムのサイズを記述します。declared_width は、CREATE TABLE 文またはクエリによって定義されたカラムのサイズを記述します。width 値はデータ型に依存します。

クライアントの型表記は、データベースサーバとは異なることがあります。たとえば、return_date_time_as_string オプションがオンの場合、DATE、TIME、TIMESTAMP データ型は文字列に変換されます。TIMESTAMP WITH TIME ZONE データ型は、クライアントに送信されるとき、常に文字列としてフォーマットされます。これらの型の width 値は、対応するフォーマットオプション文字列の長さにのみ基づくため、実際にフォーマットされた長さを反映することもしないこともあります。たとえば、小数部分を 6 桁以上の精度で指定すると width 値に影響が出ますが、実際にフォーマットされた文字列の精度は最大限で 6 桁となります。

CHARACTER データ型の場合、character-length セマンティックで宣言されたカラムは、CREATE TABLE のサイズと一致する declared_width 値を持ちます。width 値は、戻り値の文字列を格納するために必要な最大バイト数を示します。

NUMERIC、DECIMAL、FLOAT、REAL、および DOUBLE データ型の場合、declared_width と width は同じになります。width は、記号や小数点を含むフォーマットされた文字列の長さを表すことはありません。数値または 10 進数の場合、width は精度を表します。浮動小数点値の場合、width は記憶領域のサイズ (バイト) を表します。

例の一部を次の表で示します。

宣言	width	declared_width
CHAR(10)	10	10
CHAR(10 CHAR)	40	10
DATE	date_format オプション文字列の長さによって変わります	8
DOUBLE	8	8
REAL	4	4
NUMERIC(10, 3)	10 (精度)	10 (精度)
TIME	time_format オプション文字列の長さによって変わります	8

宣言	width	declared_width
TIMESTAMP	timestamp_format オプション文字列の長さによって変わります	8
TIMESTAMP WITH TIME ZONE	timestamp_with_time_zone_format オプション文字列の長さによって変わります	8

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例は、Departments テーブルのすべてのカラムを問い合わせると返される情報を記述します。

```
SELECT *
FROM sa_describe_query( 'SELECT * FROM Departments DEPT' );
```

add_keys パラメータが指定されなかったため、結果では is_key_column と is_added_key_column の値が NULL になります。

次の例は、Departments テーブルとジョインした Employees テーブルの DepartmentName カラムと Surname カラムを問い合わせることで返される情報を記述します。

```
SELECT  *
FROM  sa_describe_query( 'SELECT DepartmentName, Surname
 FROM Employees E JOIN Departments D ON E.EmployeeID = D.DepartmentHeadId',
 add_keys = 1 );
```

結果では、結果セットのロー 3 および 4 に 1 が表示されます。これは、クエリの結果セットのローをユニークに識別するのに必要なカラムが Employees.EmployeeID と Departments.DepartmentID であることを示しています。さらに、記述されるクエリで Employees.EmployeeID と Departments.DepartmentID が明示的に参照されていなかったため、ロー 3 および 4 の is_added_key_column には 1 が表示されます。

関連情報

文字データ型 [125 ページ]
EXPRTYPE 関数 [その他] [363 ページ]
SYSDOMAIN システムビュー [1800 ページ]

SYSDOMAIN システムビュー [1800 ページ]
SYSUSERTYPE システムビュー [1858 ページ]
SYSTAB システムビュー [1842 ページ]
SYSUSER システムビュー [1855 ページ]

1.6.8.25 sa_describe_shapefile システムプロシージャ

ESRI シェイプファイルに含まれるカラムの名前と型を記述します。このシステム機能は、空間データ機能とともに使用します。

構文

```
sa_describe_shapefile(
shp_filename
, srid
[, encoding ]
)
```

パラメータ

shp_filename

ESRI シェイプファイルのロケーションを識別する VARCHAR(512) パラメータ。ファイル名の拡張子は `.shp` にする必要があります。また、同じディレクトリにある同じベース名の `.dbf` ファイルを関連付ける必要があります。パスは、クライアントアプリケーションではなく、データベースサーバを基準にした相対パスを指定します。

srid

シェイプファイル内のジオメトリの SRID を識別する INTEGER パラメータ。カラムに複数の SRID を格納できるようにするには、NULL を指定します。NULL を指定すると、ジオメトリ値に対して実行できる操作が制限されます。

encoding

シェイプファイルを読み込むときに使用するエンコードを識別する任意の VARCHAR(50) パラメータ。デフォルトは NULL です。エンコードが NULL の場合は、ISO-8859-1 文字セットが使用されます。

結果セット

カラム名	データ型	説明
column_number	INTEGER	このローが記述するカラムの順序位置 (開始値は 1)。
name	VARCHAR(128)	カラム名。

カラム名	データ型	説明
domain_name_with_size	VARCHAR(160)	サイズと精度を含むデータ型名 (CREATE TABLE 関数または CAST 関数で使用されます)。

備考

sa_describe_shapefile システムプロシージャは、ESRI シェイプファイル内のカラムの名前と型を記述するときに使用します。この情報を使用して、LOAD TABLE 文または INPUT 文によってシェイプファイルからデータをロードするテーブルを作成できます。また、このシステムプロシージャは、OPENSTRING...FORMAT SHAPEFILE の WITH 句を指定することで、シェイプファイルを読み込むときにも使用できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。その他:

- -gl データベースオプションが DBA に設定されている場合は、次のシステム権限が必要です。
  - ALTER ANY TABLE
  - ALTER ANY OBJECT
  - LOAD ANY TABLE
  - READ FILE
- -gl データベースオプションが ALL に設定されている場合、権限が必要ありません。
- -gl データベースオプションが NONE に設定されている場合は、READ FILE システム権限が必要です。

例

次の例は、シェイプファイルデータを格納するためのテーブルを作成するときに使用した文字列を表示します。

```
BEGIN
    DECLARE create_cmd LONG VARCHAR;
    SELECT 'create table if not exists esri_load( record_number int primary key,
' ||
         (SELECT list( name || ' ' || domain_name_with_size, ', ' ORDER BY
column_number )
     FROM sa_describe_shapefile( 'c:¥¥esri¥¥tgr36069trt00.shp', 1000004326 )
     WHERE column_number > 1 ) || ' )'
     INTO create_cmd;
     SELECT create_cmd;
     EXECUTE IMMEDIATE create_cmd;
END
```

LOAD ANY TABLE システム権限を持っており、-gl データベースオプションが NONE に設定されていない場合は、次の文を使用してシェイプファイルデータをテーブルにロードできます。

```
LOAD TABLE esri_load
USING FILE 'c:¥¥esri¥¥tgr36069trt00.shp'
FORMAT SHAPEFILE;
```

関連情報

LOAD TABLE 文 [1182 ページ]
INPUT 文 [Interactive SQL] [1160 ページ]

1.6.8.26 sa_disable_auditing_type システムプロシージャ

特定のイベントの監査を無効にします。

構文

```
sa_disable_auditing_type( types )
```

パラメータ

types

この VARCHAR(128) パラメータを使用して、カンマで区切られた文字列を指定します。この文字列には、次の 1 つ以上の値が含まれています。

all

すべてのタイプの監査を無効にします。
connect

成功した接続と失敗した接続の両方の監査を無効にします。
connectFailed

失敗した接続の監査を無効にします。
DDL

DDL 文の監査を無効にします。
options

パブリックオプションの監査を無効にします。
permission

パーミッションチェック、ユーザチェック、SETUSER 文の監査を無効にします。
permissionDenied

失敗したパーミッションチェックと失敗したユーザチェックの監査を無効にします。
triggers

トリガイベントに応じて監査を無効にします。
xp_cmdshell xp_cmdshell の監査を無効にします。

備考

sa_disable_auditing_type を使用して、除外する監査のタイプを指定します。このシステムプロシージャは、現在の監査イベントセットから指定されたイベントを削除します。現在の監査イベントセットにイベントを追加するには、sa_enable_auditing_type を使用します。これらのシステムプロシージャは、設定が永続的になるように PUBLIC *auditing_options* データベースオプションを設定します。

監査を有効または無効にするには、PUBLIC *auditing* データベースオプションを On または Off に設定します。

デフォルトでは、すべてのイベントが監査されます (types='all')。セットを小さくする場合、sa_disable_auditing_type システムプロシージャを使用して、不要なイベントをクリアします。sa_disable_auditing_type システムプロシージャを使用してすべてのイベントをクリアし、sa_enable_auditing_type システムプロシージャを使用して必要な監査のタイプを指定します。

イベントのセットが空で、PUBLIC *auditing* データベースオプションを On に設定した場合、監査情報は記録されません。監査を再確立するには、sa_enable_auditing_type システムプロシージャを使用して監査対象にする情報のタイプを指定します。

PUBLIC *auditing* データベースオプションを Off に設定した場合、検査情報は記録されません。

audit_log データベースオプションを使用してイベントがログに記録される場所を指定します。

権限

SET ANY SECURITY OPTION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例では、すべての監査が無効になります。

```
CALL sa_disable_auditing_type( 'all' );
```

次の例では、DDL のみが有効になり、監査がトリガされます。

```
CALL sa_disable_auditing_type( 'all' );
CALL sa_enable_auditing_type( 'DDL,triggers' );
```

次の例では、DDL とオプション監査を除くすべての監査が有効になります。

```
CALL sa_enable_auditing_type( 'all' );
CALL sa_disable_auditing_type( 'DDL,options' );
```

関連情報

sa_enable_auditing_type システムプロシージャ [1500 ページ]

1.6.8.27 sa_disk_free_space システムプロシージャ

DB 領域、トランザクションログ、トランザクションログミラー、テンポラリファイルに使用可能な領域に関する情報を報告します。

構文

```
sa_disk_free_space( [ p_dbspace_name ] )
```

パラメータ

p_dbspace_name

DB 領域、トランザクションログファイル、トランザクションログミラーファイル、またはテンポラリファイルの名前を指定する VARCHAR(128) パラメータです。デフォルトは NULL です。

log、mirror、または temp という DB 領域がある場合は、キーワードの前にアンダースコアを付けます。たとえば、log という DB 領域が存在する場合は、_log を使用してトランザクションログファイルに関する情報を取得します。

SYSTEM を指定すると、メインデータベースファイルに関する情報を取得できます。TEMPORARY または TEMP を指定するとテンポラリファイル、TRANSLOG を指定するとトランザクションログ、TRANSLOGMIRROR を指定するとトランザクションログミラーに関する情報を、それぞれ取得できます。

結果セット

カラム名	データ型	説明
dbspace_name	VARCHAR(128)	これは、DB 領域名、トランザクションログファイル、トランザクションログミラーファイル、またはテンポラリファイルです。
free_space	UNSIGNED BIGINT	ボリューム上の空きバイト数。
total_space	UNSIGNED BIGINT	DB 領域があるドライブで使用可能な合計ディスク領域。

備考

p_dbspace_name パラメータが指定されていないか NULL の場合、存在する場合はトランザクションログ、トランザクションログミラー、テンポラリファイルごとに 1 つのローと、さらに DB 領域ごとに 1 つのローが結果セットに含まれます。
p_dbspace_name が指定されている場合は、1 つまたは 0 個のローが返ります (このような DB 領域が存在しない場合や、log または mirror が指定されていてログファイルまたはミラーファイルが存在しない場合は 0 です)。

権限

MANAGE ANY DBSPACE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、sa_disk_free_space システムプロシージャを使用して、空き領域についての情報を格納した結果セットを返します。

```
CALL sa_disk_free_space( );
```

dbspace_name	free_space	total_space
system	10952101888	21410402304
translog	10952101888	21410402304
temporary	10952101888	21410402304

1.6.8.28 sa_enable_auditing_type システムプロシージャ

監査に含めるイベントを指定します。

構文

```
sa_enable_auditing_type( types )
```

パラメータ

types

この VARCHAR(128) パラメータを使用して、カンマで区切られた文字列を指定します。この文字列には、次の 1 つ以上の値が含まれています。

all

すべてのタイプの監査を有効にします。

connect

成功した接続と失敗した接続の両方の監査を有効にします。

connectFailed

失敗した接続の監査を有効にします。

DDL

DDL 文の監査を有効にします。

options

パブリックオプションの監査を有効にします。

permission

パーミッションチェック、ユーザチェック、SETUSER 文の監査を有効にします。

permissionDenied

失敗したパーミッションチェックと失敗したユーザチェックの監査を有効にします。

triggers

トリガイベントの監査を有効にします。

xp_cmdshell

xp_cmdshell 呼び出しの監査を有効にします。

備考

sa_enable_auditing_type を使用して、除外する監査のタイプを指定します。このシステムプロシージャは、現在の監査イベントセットに指定されたイベントを追加します。現在の監査イベントセットからイベントを削除するには、sa_disable_auditing_type を使用します。これらのシステムプロシージャは、設定が永続的になるように PUBLIC *auditing_options* データベースオプションを設定します。

監査を有効または無効にするには、PUBLIC *auditing* データベースオプションを On または Off に設定します。

デフォルトでは、すべてのイベントが監査されます (types='all')。セットを小さくする場合、sa_disable_auditing_type システムプロシージャを使用して、不要なイベントをクリアします。sa_disable_auditing_type システムプロシージャを使用してすべてのイベントをクリアし、sa_enable_auditing_type システムプロシージャを使用して必要な監査のタイプを指定します。

イベントのセットが空で、PUBLIC *auditing* データベースオプションを On に設定した場合、監査情報は記録されません。監査を再確立するには、sa_enable_auditing_type システムプロシージャを使用して監査対象にする情報のタイプを指定します。

PUBLIC *auditing* データベースオプションを Off に設定した場合、検査情報は記録されません。

audit_log データベースオプションを使用してイベントがログに記録される場所を指定します。

権限

SET ANY SECURITY OPTION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、すべての監査が有効になります。

```
CALL sa_enable_auditing_type( 'all' );
```

次の例では、DDL のみが有効になり、監査がトリガされます。

```
CALL sa_disable_auditing_type( 'all' );
CALL sa_enable_auditing_type( 'DDL,triggers' );
```

次の例は、DDL のみを有効にして、監査をトリガするもう 1 つの方法です。

```
CALL sa_disable_auditing_type( 'all' );
CALL sa_enable_auditing_type( 'triggers' );
CALL sa_enable_auditing_type( 'DDL' );
```

関連情報

sa_disable_auditing_type システムプロシージャ [1497 ページ]

1.6.8.29 sa_eng_properties システムプロシージャ

データベースサーバのプロパティ情報をレポートします。

構文

```
sa_eng_properties( )
```

結果セット

カラム名	データ型	説明
PropNum	INTEGER	データベースサーバプロパティの番号。
PropName	VARCHAR(255)	データベースサーバプロパティ名。
PropDescription	VARCHAR(255)	データベースサーバプロパティの説明。
Value	LONG VARCHAR	データベースサーバプロパティの値。

備考

使用可能な各サーバプロパティの PropNum、PropName、PropDescription、Value を返します。データベースサーバに関するすべてのデータベースサーバプロパティと統計情報の値が返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、使用可能な一連のサーバプロパティを返します。

```
CALL sa_eng_properties( );
```

PropNum	PropName	...
1	IdleWrite	...
2	IdleChkPt	...
...	...	...

関連情報

システム関数 [215 ページ]
xp_cmdshell システムプロシージャ [1759 ページ]

1.6.8.30 sa_error_stack_trace システムプロシージャ

エラーハンドラを呼び出したエラーのスタックトレースを返します。

構文

sa_error_stack_trace( )

結果セット

カラム名	データ型	説明
StackLevel	UNSIGNED SMALLINT	スタックの行番号 (1 番上の行が 1)。エラーを生成した文が最も大きい番号となります。
UserName	CHAR(128)	現在のレベルがバッチにある場合のプロシージャの所有者名または NULL。
ProcName	CHAR(128)	文が実行されたプロシージャまたはバッチタイプの名前。
LineNumber	UNSIGNED INTEGER	プロシージャ内の呼び出しの行番号。
IsResignal	BIT	文が送り返された場合は 1。それ以外の場合は 0。

備考

結果セットの各ローは、エラーの呼び出しスタックの 1 回の呼び出しを表します。複合文がプロシージャ、ファンクション、トリガ、イベントのいずれの一部でもない場合は、プロシージャ名ではなく、バッチのタイプ (`<watcom_batch>` または `<tsql_batch>`) が返されます。

この関数は、プロシージャの SYSPROCEDURE システムテーブルの proc_defn カラムで見つかった行番号を返します。これらの行番号は、プロシージャの作成に使用されるソース定義の行番号とは異なる可能性があります。

このプロシージャは、ERROR_STACK_TRACE 関数と同じ情報を返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

この例は、sa_error_stack_trace システムプロシージャの呼び出しの例を示しています。

```
CALL sa_error_stack_trace();
```

この例は、RESIGNAL による sa_error_stack_trace システムプロシージャの出力を示しています。

```
CREATE OR REPLACE PROCEDURE error_reporting_procedure()
BEGIN
    SELECT *
    FROM sa_error_stack_trace();
END;
CREATE OR REPLACE PROCEDURE proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL proc2();
    END CATCH
END TRY
BEGIN CATCH
    CALL error_reporting_procedure();
END CATCH;
CREATE OR REPLACE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;
CREATE OR REPLACE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;
CALL proc1();
```

上記の例を実行すると、次の結果セットが生成されます。

StackLevel	UserName	ProcName	LineNumber	IsResignal
1	DBA	proc1	8	0
2	DBA	proc2	3	0
3	DBA	proc3	3	1
4	DBA	proc1	5	0

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]

ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]

1.6.8.31 sa_event_schedules システムプロシージャ

イベントについてのスケジュール情報を表示します。

構文

```
sa_event_schedules( evt_id )
```

パラメータ

evt_id

イベントの ID 番号を取得する INTEGER パラメータです。

結果セット

カラム名	データ型	説明
sched_name	VARCHAR(128)	イベントを実行するスケジュールの名前です。
sched_def	LONG VARCHAR	スケジュール定義です。

備考

このプロシージャは、イベントが実行されるスケジュールや実行の頻度などの、指定したイベントに関する情報を返します。イベントにスケジュールがない場合、プロシージャは結果セットを返しません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、IncrementalBackup イベントのイベントスケジュールを取得します。

```
call sa_event_schedules( (SELECT event_id FROM SYSEVENT WHERE
event_name='IncrementalBackup') );
```

sched_name	sched_def
IncrementalBackup	START TIME '01:00:00' EVERY 24 HOURS

関連情報

CREATE EVENT 文 [805 ページ]
EVENT_CONDITION 関数 [システム] [353 ページ]
EVENT_CONDITION_NAME 関数 [システム] [355 ページ]
EVENT_PARAMETER 関数 [システム] [356 ページ]

1.6.8.32 sa_external_library_unload システムプロシージャ

外部ライブラリをアンロードします。

構文

```
sa_external_library_unload( [ lib_name ] )
```

パラメータ

lib_name

任意の LONG VARCHAR パラメータを使用して、アンロードするライブラリの名前を指定します。ライブラリを指定しない場合は、使用中でない外部ライブラリがすべてアンロードされます。デフォルトは NULL です。

備考

外部ライブラリを指定したとき、そのライブラリが使用中である場合、またはライブラリがロードされていない場合は、エラーが返されます。パラメータが指定されていない場合、エラーは返されません。

ライブラリ名は、元のライブラリ指定のパスと大文字と小文字の区別が正確に一致している必要があります。たとえば、ライブラリ名は、次の EXTERNAL NAME 句の @ 以降の文字列と正確に一致している必要があります。

```
EXTERNAL NAME 'xp_replicate@c:¥¥sqlany¥¥samples¥¥sqlanywhere¥¥ExternalProcedures¥
¥extproc.dll'
```

権限

MANAGE ANY EXTERNAL OBJECT システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、外部ライブラリ `extproc.dll` をアンロードします。

```
CALL sa_external_library_unload( 'extproc.dll' );
```

次の例は、現在使用中でないライブラリをすべてアンロードします。

```
CALL sa_external_library_unload();
```

1.6.8.33 sa_flush_cache システムプロシージャ

データベースサーバキャッシュ内の現在のデータベースに対するすべてのページを空にします。

構文

```
sa_flush_cache( )
```

備考

データベース管理者は、このプロシージャを使用して現在のデータベースのデータベースサーバキャッシュの内容を空にします。パフォーマンスの計測に使用し、同じ結果が繰り返し得られるようにします。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、データベースサーバキャッシュ内の現在のデータベースに対するすべてのページを空にします。

```
CALL sa_flush_cache( );
```

1.6.8.34 sa_flush_statistics システムプロシージャ

すべてのコストモデル統計をデータベースサーバキャッシュに保存します。

構文

sa_flush_statistics( )

備考

このプロシージャを使用して、データベースに現在キャッシュされている現在のコストモデル統計情報をディスクにフラッシュします。sa_get_histogram システムプロシージャまたはヒストグラムユーティリティ (dbhist) を使用して統計情報を取得できます。このシステムプロシージャを実行すると、ISYSCOLSTAT システムテーブルが更新されます。通常のオペレーションでは、サーバによって、ディスクへの統計値の自動書き出しが定期的に行われるため、このプロシージャを実行する必要はありません。

権限

MANAGE ANY STATISTICS または SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、すべてのコストモデル統計をデータベースサーバキャッシュに保存します。

```
CALL sa_flush_statistics( );
```

関連情報

sa_get_histogram システムプロシージャ [1515 ページ]
SYSCOLSTAT システムビュー [1796 ページ]

1.6.8.35 sa_get_bits システムプロシージャ

ビット文字列を取得し、文字列内の各ビットのローを返します。デフォルトでは、1 のビット値を持つローのみが返されます。

構文

```
sa_get_bits(
bit_string
[, only_on_bits ]
)
```

パラメータ

bit_string

この LONG VARBIT パラメータを使用して、ビットを取得する対象のビット文字列を指定します。`bit_string` パラメータが NULL の場合、ローは返されません。

only_on_bits

任意の BIT パラメータを使用して、オンのビット (値が 1 のビット) を持つローのみを返すかどうかを指定します。1 (デフォルト) を指定すると、オンのビットを持つローのみを返します。0 を指定すると、ビット文字列のすべてのビットのローを返します。

結果セット

カラム	データ型	説明
bitnum	UNSIGNED INTEGER	このローで記述されるビットの位置です。たとえば、ビット文字列の最初のビットは bitnum が 1 になります。
bit_val	BIT	bitnum 位置にあるビットの値です。`only_on_bits` が 1 に設定されている場合、この値は常に 1 です。

備考

sa_get_bits システムプロシージャは、ビット文字列を復号化し、ビット文字列の各ビットについて 1 つのローを返し、ビットの値を示します。`only_on_bits` が 1 (デフォルト) または NULL の場合、オンのビットに対応するローのみが返されます。最適化により、オンのビットが少ない長いビット文字列を効率的に処理できます。`only_on_bits` が 0 に設定されている場合、ビット文字列の各ビットについて 1 つのローが返されます。

たとえば、文 `CALL sa_get_bits( '1010' )` は、ビット文字列の位置 1 と 3 に ON のビットを示す、次の結果セットを返します。

bitnum	bit_val
1	1
3	1

sa_get_bits システムプロシージャは、ビット文字列を関係に変換するときに使用できます。これは、ビット文字列をテーブルに結合するときや、1 つのバイナリ値としてではなく、結果セットとしてビット文字列を取得するときに使用します。大量に 0 ビットがある場合、これらのビットを取得する必要はないため、ビット文字列を結果セットとして取得する方が効率的です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、sa_get_bits システムプロシージャを使用して、整数のセットをビット文字列としてコード化してから、ジョインで使用するために復号化する方法を示します。

```
CREATE VARIABLE @s_depts LONG VARBIT;
SELECT  SET_BITS( DepartmentID )
 INTO @s_depts
 FROM Departments
 WHERE DepartmentName like 'S%';
SELECT *
 FROM sa_get_bits( @s_depts ) B
  JOIN Departments D ON B.bitnum = D.DepartmentID;
```

関連情報

sa_split_list システムプロシージャ [1630 ページ]
SET_BIT 関数 [ビット配列] [522 ページ]
SET_BITS 関数 [集合] [524 ページ]
GET_BIT 関数 [ビット配列] [372 ページ]

1.6.8.36 sa_get_dtt システムプロシージャ

コストモデルの一部であるディスク転送時間 (DTT: Disk Transfer Time) モデルの現在の値をレポートします。

構文

```
sa_get_dtt( file_id )
```

パラメータ

file_id

この UNSIGNED SMALLINT パラメータを使用して、データベースファイル ID を指定します。

結果セット

カラム名	データ型	説明
BandSize	UNSIGNED INTEGER	ランダムアクセスが行われるディスクのサイズ (ページ単位)。
ReadTime	UNSIGNED INTEGER	1 ページの読み込みの分散コスト (マイクロ秒単位)。
WriteTime	UNSIGNED INTEGER	1 ページの書き込みの分散コスト (マイクロ秒単位)。
QueueDepth	UNSIGNED INTEGER	ランダムアクセスが行われるディスクの I/O キューの未処理の I/O の数。

備考

file_id は SYSDBSPACE システムビューから取得できます。

このプロシージャは、ISYSOPTSTAT システムテーブルからデータを取得します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、システム DB 領域のディスク転送時間 (DTT: Disk Transfer Time) モデルの現在の値をレポートします。

```
CALL sa_get_dtt( (select dbspace_id from SYSDBSPACE where
dbspace_name='system') );
```

関連情報

SYSDBSPACE システムビュー [1798 ページ]
SYSOPTSTAT システムビュー [1822 ページ]

sa_get_dtt_groupreads システムプロシージャ [1514 ページ]

1.6.8.37 sa_get_dtt_groupreads システムプロシージャ

データベースサーバでグループ読み込みを発行するコストを評価し、レポートします。

構文

```
sa_get_dtt_groupreads( dbspace_id )
```

パラメータ

dbspace_id

この UNSIGNED SMALLINT パラメータを使用して、データベースファイル ID を指定します。

結果セット

カラム名	データ型	説明
GroupSize	UNSIGNED INTEGER	ランダムアクセスが行われるディスクのサイズ (ページ単位)。
ReadTime	FLOAT	1 ページの読み込みの分散コスト (マイクロ秒単位)。

備考

dbspace_id は SYSDBSPACE システムビューから取得できます。sa_get_dtt_groupreads システムプロシージャによって返される評価は、コストモデルの一部です。この評価を使用して、ソートなどの操作時に適切なグループ読み込みのサイズが選択されます。

このプロシージャは、内部診断を目的としており、システムテーブル ISYSOPTSTAT からデータを取り出します。このテーブルにエントリが記録されていない場合、通常の値が返されます。使用するハードウェアの評価を調整するには、次の文を実行します。

```
ALTER DATABASE CALIBRATE GROUP READ;
```

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、システム DB 領域のグループ読み込みを発行するコストをレポートします。

```
CALL sa_get_dtt_groupreads( (select dbspace_id from SYSDBSPACE where
dbspace_name='system') );
```

関連情報

SYSDBSPACE システムビュー [1798 ページ]
SYSOPTSTAT システムビュー [1822 ページ]
ALTER DATABASE 文 [630 ページ]
sa_get_dtt システムプロシージャ [1512 ページ]

1.6.8.38 sa_get_histogram システムプロシージャ

カラムのヒストグラムを取得します。

構文

```
sa_get_histogram(
col_name
, tbl_name
[, owner_name ]
)
```

パラメータ

col_name

この CHAR(128) パラメータを使用して、ヒストグラムを取得する対象のカラムを指定します。

tbl_name

この CHAR(128) パラメータを使用して、col_name があるテーブルを指定します。

owner_name

任意の CHAR(128) パラメータを使用して、tbl_name の所有者を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
StepNumber	SMALLINT	ヒストグラムのバケット番号。最初のバケット (StepNumber = 0) の頻度は、NULL の選択性を示します。
Low	CHAR(128)	バケット内の最も低いカラム値 (その値を含む)。
High	CHAR(128)	バケット内の最も高いカラム値 (その値を含まない)。
Frequency	DOUBLE	バケット内の値の選択性。

備考

このプロシージャは、内部診断を目的としており、指定したカラムのデータベースサーバからカラム統計を取り出します。この統計が ISYSCOLSTAT システムテーブルに永続的に格納されている場合、サーバの実行中に統計はメモリに保持され、定期的に ISYSCOLSTAT に書き込まれます。この結果、sa_get_histogram システムプロシージャが返す統計情報は、任意の時点で ISYSCOLSTAT から選択して取得した情報とは異なる場合があります。

sa_flush_statistics システムプロシージャを使用してメモリに保存されている最新の統計情報で ISYSCOLSTAT を手動で更新できます。ただし、この方法は運用環境では推奨されません。診断目的でのみ使用してください。

単一バケットは、結果セット内の Low 値が対応する High 値と等しいことで示されます。

ヒストグラムを表示するには、ヒストグラムユーティリティを使用します。

文字列カラムに対する述部の選択性を決定するには、ESTIMATE 関数または ESTIMATE_SOURCE 関数を使用します。文字列カラムに対して、sa_get_histogram とヒストグラムユーティリティが ISYSCOLSTAT システムテーブルから取り出すものはありません。文字列データを取り出そうとすると、エラーが発生します。

たとえば、統計が最近削除された場合は、テーブルまたはマテリアライズドビューに対する統計 (ヒストグラムを含む) が存在しないことがあります。この場合、sa_get_histogram システムプロシージャの結果セットは空です。テーブルまたはマテリアライズドビューの統計を作成するには、CREATE STATISTICS 文を実行します。

権限

MANAGE ANY STATISTICS システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

たとえば、次の文は、SalesOrderItems テーブルの ProductID カラムのヒストグラムを取り出します。

```
CALL sa_get_histogram( 'ProductID', 'SalesOrderItems' );
```

関連情報

SYSCOLSTAT システムビュー [1796 ページ]
CREATE STATISTICS 文 [942 ページ]
sa_flush_statistics システムプロシージャ [1509 ページ]
ESTIMATE_SOURCE 関数 [その他] [352 ページ]
ESTIMATE 関数 [その他] [350 ページ]

1.6.8.39 sa_get_ldapserver_status システムプロシージャ

LDAP サーバの現在のステータスを特定できます。

構文

sa_get_ldapserver_status( )

結果セット

カラム名	データ型	説明
ldsrv_id	UNSIGNED BIGINT	LDAP サーバを識別するユニークな番号。
ldsrv_name	CHAR(128)	LDAP サーバの名前。
ldsrv_state	CHAR(9)	最後のチェックポイントにおける LDAP サーバの現在のステータス。

カラム名	データ型	説明
ldsrv_last_state_change	TIMESTAMP	状態が変更された日時。データベースサーバのローカルのタイムゾーンに関係なく、この値は協定世界時 (UTC: Coordinated Universal Time) で格納されます。この値はシミュレートされたタイムゾーンの影響を受けません。

備考

このプロシージャは、LDAP サーバの現在のステータスを示す結果セットを返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例は、sa_get_ldapserver_status システムプロシージャを使用して、LDAP サーバのステータスを返します。

```
CALL sa_get_ldapserver_status;
```

1.6.8.40 sa_get_request_profile システムプロシージャ

要求ログを分析し、同様の文の実行時間を判断します。

構文

```
sa_get_request_profile(
   [ filename
   [, conn_id
   [, first_file
   [, num_files ] ] ] ]
)
```

パラメータ

filename

任意の LONG VARCHAR パラメータを使用して、要求ロギングのファイル名を指定します。デフォルトは NULL です。

conn_id

任意の UNSIGNED INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは 0 です。

first_file

任意の INTEGER パラメータを使用して、分析する最初の要求ログファイルを指定します。デフォルトは -1 です。

num_files

任意の INTEGER パラメータを使用して、分析する要求ログファイルの数を指定します。デフォルトは 1 です。

備考

このプロシージャは、sa_get_request_times を呼び出して要求ログファイルを処理してから、結果をグローバルテンポラリテーブル SATMP_request_profile に要約します。このテーブルは、ログからの文、各文の実行回数、合計、平均、最大の各実行時間で構成されています。このテーブルをさまざまな方法でソートして、パフォーマンスの最適化のターゲットを識別できます。

要求ログファイル (`filename`) を指定しない場合、デフォルトは現在のログファイルです。このファイルは、-zo データベースサーバオプションを使用して指定するか、または次のように指定します。

```
sa_server_option( 'RequestLogFile', filename )
```

接続 ID を指定すると、ログからの情報のフィルタに使用され、その接続に関する要求だけが取り出されます。

権限

DIAGNOSTICS システムロールおよび MANAGE PROFILING システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

オートコミット

例

次のコマンドは、ファイル `req.out` 内の要求の要求回数を取得します。

```
CALL sa_get_request_profile('req.out');
```

次のコマンドは、ファイル req.out.3、req.out.4、req.out.5 内の要求の要求回数を取得します。

```
CALL sa_get_request_profile('req.out',0,3,3);
```

関連情報

sa_get_request_times システムプロシージャ [1520 ページ]
sa_statement_text システムプロシージャ [1636 ページ]
sa_server_option システムプロシージャ [1606 ページ]

1.6.8.41 sa_get_request_times システムプロシージャ

要求ログを分析し、文の実行時間を判別します。

構文

```
sa_get_request_times( [ filename
   [, conn_id
   [, first_file
   [, num_files ] ] ] ]
)
```

パラメータ

filename

任意の LONG VARCHAR パラメータを使用して、要求ロギングのファイル名を指定します。デフォルトは NULL です。

conn_id

任意の UNSIGNED INTEGER パラメータを使用して、接続 ID 番号を指定します。デフォルトは 0 です。

first_file

任意の INTEGER パラメータを使用して、分析する最初のファイルを指定します。デフォルトは -1 です。

num_files

任意の INTEGER パラメータを使用して、分析する要求ログファイルの数を指定します。デフォルトは 1 です。

備考

このプロシージャは、指定された要求ログを読み込み、ログから文とその実行時間をグローバルテンポラリテーブル SATMP_request_time に渡します。

INSERT や UPDATE のような文の場合、実行時間は単純です。クエリの場合は、文を記述する、カーソルを開く、ローをフェッチする、カーソルを閉じるなどの各操作を含め、文を準備してから削除するまでの時間が計算されます。ほとんどのクエリの場合、これには所要時間が正確に反映されます。カーソルが開いている間に、オペレータ操作やクライアント処理などの他のイベントが発生した場合は、時間値が大きく見えますが、クエリが高コストであることを示しているわけではありません。

このプロシージャは要求ログ内のホスト変数を認識し、それらの値をグローバルテンポラリテーブル SATMP_request_hostvar に移植します。このテンポラリテーブルがない古いデータベースの場合、ホスト変数の値は無視されます。

要求ログファイルを指定しない場合、デフォルトは現在のログファイルです。このファイルは、-zo を使用してコマンドで指定するか、または次のように指定します。

```
call sa_server_option( 'RequestLogFile', filename )
```

接続 ID を指定すると、ログからの情報のフィルタに使用され、その接続に関する要求だけが取り出されます。

権限

MANAGE PROFILING または MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

オートコミット

例

次のコマンドは、ファイル `req.out` 内の要求の実行回数を取得します。

```
CALL sa_get_request_times('req.out');
```

次のコマンドは、ファイル `req.out.3`、`req.out.4`、`req.out.5` 内の要求の実行回数を取得します。

```
CALL sa_get_request_times('req.out',0,3,3);
```

関連情報

sa_get_request_profile システムプロシージャ [1518 ページ]
sa_statement_text システムプロシージャ [1636 ページ]
sa_server_option システムプロシージャ [1606 ページ]

1.6.8.42 sa_get_server_messages システムプロシージャ [旧式]

データベースサーバメッセージウィンドウの定数を結果セットとして返すことができます。このシステムプロシージャは廃止される予定です。代わりに sa_server_messages システムプロシージャを使用します。

構文

```
sa_get_server_messages( first_line )
```

パラメータ

first_line

この INTEGER パラメータを使用して、サーバメッセージの表示を開始する行番号を指定します。

結果セット

カラム名	データ型	説明
line_num	INTEGER	サーバメッセージの行番号です。
message_text	VARCHAR(255)	サーバメッセージのテキストです。
message_time	TIMESTAMP	メッセージの時刻です。

備考

このプロシージャは、表示する最初の行番号を指定する INTEGER パラメータを受け取り、その行と後続するすべての行のローを返します。開始行が負の場合は、結果セットは使用可能な最初の行から始まります。結果セットには、行番号、メッセージテキスト、メッセージ時刻が含まれます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、sa_get_server_messages システムプロシージャを使用して、データベースサーバメッセージウィンドウの 16 行目からの内容を格納した結果セットを返します。

```
CALL sa_get_server_messages( 16 );
```

line_num	message_text	...
16	Windows 7 Build 7601 で実行中です。	...
17	Server built for X86_64 processor architecture	...
...	...	...

1.6.8.43 sa_get_table_definition システムプロシージャ

指定されたテーブルとそのインデックス、外部キー、トリガ、付与された権限の作成に必要な SQL 文を含む LONG VARCHAR 文字列を返します。

構文

```
sa_get_table_definition(
@owner
, @tabname
)
```

パラメータ

@owner

この VARCHAR(128) パラメータを使用して、tabname の所有者を指定します。

@tabname

この VARCHAR(128) パラメータを使用して、テーブルの名前を指定します。

備考

この関数は、指定されたテーブルとそのインデックス、外部キー、トリガ、付与された権限の作成に必要な SQL 文を含む LONG VARCHAR 文字列を返します。新しいテーブルを同じ定義を使用して作成するには、この関数が返す文字列を、EXECUTE IMMEDIATE 文および LOCATE、SUBSTRING、REPLACE の各関数と組み合わせて使用します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

SELECT ANY TABLE システム権限、または、SYSUSERPERM 互換ビューに対する SELECT 権限も必要です。

関連する動作

なし

例

次の文は、sa_get_table_definition 関数を使用して、Departments テーブルを作成するために必要な SQL 文を含む文字列を表示します。

```
SELECT sa_get_table_definition( 'GROUPO', 'Departments' );
```

関連情報

SYSUSERPERM 互換ビュー (廃止予定) [1899 ページ]
sa_split_list システムプロシージャ [1630 ページ]
EXECUTE IMMEDIATE 文 [SP] [1090 ページ]
LOCATE 関数 [文字列] [423 ページ]
SUBSTRING 関数 [文字列] [551 ページ]
REPLACE 関数 [文字列] [502 ページ]

1.6.8.44 sa_get_user_status システムプロシージャ

ユーザの現在のステータスを特定できます。

構文

```
sa_get_user_status( )
```

結果セット

カラム名	データ型	説明
user_id	UNSIGNED INTEGER	ユーザを識別するユニークな番号です。
user_name	CHAR(128)	ユーザの名前です。
connections	INTEGER	このユーザによる現在の接続数です。
failed_logins	UNSIGNED INTEGER	ユーザがログインしようとして失敗した回数です。
last_login_time	TIMESTAMP	最近の接続が確立されたときのデータベースサーバのローカル日時 (時、分までの精度) を返します。この値はシミュレートされたタイムゾーンの影響を受けます。
locked	TINYINT	ユーザアカウントがロックされているかどうかを示すインジケータです。
reason_locked	LONG VARCHAR	アカウントがロックされた理由です。
user_dn	CHAR(1024)	LDAP サーバに接続しているユーザ ID の識別名です (DN)。
user_dn_cached_at	TIMESTAMP	user_dn カラムが最後にキャッシュされた日時。この値は古い DN をパージするかどうか決めるために使用されます。データベースサーバのローカルのタイムゾーンに関係なく、この値は協定世界時 (UTC: Coordinated Universal Time) で格納されます。この値はシミュレートされたタイムゾーンの影響を受けません。
password_change_state	BIT	二重パスワードの変更が進行中であるかどうかを示す値です (0 = いいえ、1 = はい)。デフォルトは 0 です。
password_change_first_user	UNSIGNED INTEGER	二重パスワードの最初の部分を設定したユーザの user_id で、それ以外の場合は NULL です。
password_change_second_user	UNSIGNED INTEGER	二重パスワードの 2 番目の部分を設定したユーザの user_id で、それ以外の場合は NULL です。

備考

このプロシージャは、ユーザの現在のステータスを示す結果セットを返します。基本的なユーザ情報に加えて、ユーザがロックアウトされているかどうかを示すカラムや、ロックアウトの理由が格納されたカラムが含まれています。ユーザは、ポリシーによるロック、パスワードの失効、または失敗した試行回数の過多などの理由によって、ロックアウトされる可能性があります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

他のユーザの情報を表示する場合は、MANAGE ANY USER システム権限も必要です。

関連する動作

なし。

例

次の例は、sa_get_user_status システムプロシージャを使用して、データベースユーザのステータスを返します。

```
CALL sa_get_user_status;
```

1.6.8.45 sa_http_header_info システムプロシージャ

HTTP 要求ヘッダ名と値を返します。

構文

```
sa_http_header_info( [header_parm] )
```

パラメータ

header_parm

任意の VARCHAR(255) パラメータを使用して、HTTP ヘッダ名を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
[名前]	VARCHAR(255)	HTTP ヘッダ名。
Value	LONG VARCHAR	HTTP ヘッダの値。

備考

sa_http_header_info システムプロシージャは、HTTP ヘッダ名と値を返します。任意のパラメータを使用してヘッダ名を指定しない場合、結果セットにはすべてのヘッダの値が格納されます。

このプロシージャは、Web サービス内の HTTP 要求の処理中に呼び出された場合は、空でない結果セットを返します。

注記

要求に同名の HTTP ヘッダが複数含まれている場合、sa_http_header_info システムプロシージャが同名のローを複数返すことがあります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

Web サービスから呼び出された次の Web サービスプロシージャは、sa_http_header_info システムプロシージャを示します。

```
CREATE OR REPLACE PROCEDURE User1.HTTPHeaderExample()
RESULT ( html_string LONG VARCHAR )
BEGIN
    DECLARE myname VARCHAR(255);
    DECLARE myvalue LONG VARCHAR;
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    DECLARE curs CURSOR FOR
        SELECT Name, Value FROM sa_http_header_info();
    MESSAGE '=== HTTP Headers ===' TO CONSOLE;
    OPEN curs;
    FetchLoop: LOOP
        FETCH next curs INTO myname, myvalue;
        IF SQLSTATE = err_notfound THEN
            LEAVE FetchLoop;
        END IF;
        MESSAGE myname, '=', myvalue TO CONSOLE;
    END LOOP FetchLoop;
    CLOSE curs;
END;
```

この Web サービスプロシージャを呼び出した Web サービスを使用した場合、次のようなデータベースサーバメッセージウィンドウに出力が表示されます。

```
=== HTTP Headers ===
@HttpQueryString=param1=value1&param2=value2&param3=value3
```

```
User-Agent=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/
16.0
Authorization=Basic VXNlcjE6dXNlcg==
Cache-Control=max-age=0
Connection=keep-alive
Host=webserver-t3500.sap.com:8082
@HttpURI=/ShowHTTPHeaders?param1=value1&param2=value2&param3=value3
@HttpMethod=GET
Accept=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
@HttpVersion=HTTP/1.1
Accept-Language=en-US,en;q=0.5
Accept-Encoding=gzip, deflate
```

関連情報

Web サービス関数 [212 ページ]
NEXT_HTTP_HEADER 関数 [Web サービス] [452 ページ]
HTTP_HEADER 関数 [Web サービス] [395 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.46 sa_http_php_page システムプロシージャ

ヘッダ、GET または POST データ、プロトコルバージョン、要求 URL、メソッドなど、現在の HTTP 要求のコンテキスト情報を使用して、PHP インタプリタによって解釈される PHP コードを渡した結果を返します。

構文

```
sa_http_php_page( php_page )
```

パラメータ

php_page

この LONG VARCHAR パラメータには、開始マーカーと終了マーカー (<?php と ?>) を含めて、解釈される全 PHP コードが含まれます。

戻り値

この関数は LONG BINARY 値を返します。

備考

このシステムプロシージャを使用するには、PHP 外部環境をインストールしておく必要があります。

このシステムプロシージャの所有者は DBO です。ただし、セキュリティの向上のため、sa_http_php_page システムプロシージャは呼び出し元として実行されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、PHP インタープリタに phpinfo() クエリが送信され、HTML 結果が表示されます。

```
SELECT CAST( sa_http_php_page('<?php phpinfo(); ?>') AS LONG VARCHAR );
```

次の例では、PHP インタープリタに PHP スクリプトが送信され、XML 結果が表示されます。

```
SELECT CAST( sa_http_php_page('<?php '||
    '$conn = sasql_connect( "UID=DBA;PWD=sql" ); '||
    '$result = sasql_query( $conn, "SELECT * FROM Employees" ); '||
    'sasql_result_all( $result ); '||
    'sasql_free_result( $result ); '||
    'sasql_disconnect( $conn ); '||
    '?>') AS LONG VARCHAR );
```

関連情報

Web サービス関数 [212 ページ]
sa_http_php_page_interpreted システムプロシージャ [1530 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.47 sa_http_php_page_interpreted システムプロシージャ

ヘッダ、GET または POST データ、プロトコルバージョン、要求 URL、メソッドなど、指定したコンテキスト情報のパラメータを使用して、PHP インタプリタによって解釈される PHP コードを渡した結果を返します。

構文

```
sa_http_php_page_interpreted(
php_page
, method
, url
, version
, headers
, request_body
)
```

パラメータ

php_page

この LONG VARCHAR パラメータには、開始マーカーと終了マーカー (<?php と ?>) を含めて、解釈される全 PHP コードが含まれます。

method

この LONG VARCHAR パラメータには、HTTP 要求のメソッドが含まれます (たとえば、GET、POST、PUT、またはその他の標準的な要求メソッドのいずれか)。method の値は、現在の HTTP 要求の @HttpMethod の値を使用して決定できます。

url

この LONG VARCHAR パラメータには、クエリ文字列 (存在する場合) を含めて、完全な HTTP 要求の URL が含まれます。url の値は、現在の HTTP 要求の @HttpURI の値を使用して決定できます。

version

この LONG VARCHAR パラメータには、HTTP 要求のプロトコルバージョン (たとえば、HTTP/1.1 など) が含まれます。version の値は、現在の HTTP 要求の @HttpVersion の値を使用して決定できます。

headers

この LONG BINARY パラメータには、標準の HTTP ヘッダフォーマット Field-Name: Value¥r¥n の HTTP 要求ヘッダが含まれます。ヘッダの値は、次の SELECT 文を使用して、現在の HTTP 要求から取り出すことができます。

```
SELECT LIST( name || ': ' || value, CHAR(13) || CHAR(10) )
   FROM sa_http_header_info();
```

request_body

この LONG BINARY パラメータには、HTTP 要求の本文がバイナリ形式で含まれます。request_body の値は、HTTP_BODY 関数を使用して、現在の HTTP 要求から取り出すことができます。

戻り値

この関数は LONG BINARY 値を返します。

備考

このシステムプロシージャを使用するには、PHP 外部環境をインストールしておく必要があります。

このシステムプロシージャを Web サービス要求外で使用するには、要求情報を指定する必要があります。PHP コード内に設定されたヘッダは失われます。

このシステムプロシージャの所有者は DBO です。ただし、セキュリティの向上のため、sa_http_php_page_interpreted システムプロシージャは呼び出し元として実行されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、PHP インタープリタに phpinfo() クエリが送信され、HTML 結果が表示されます。

```
BEGIN
    DECLARE headers LONG VARCHAR;
    SELECT list( name || ': ' || value, char(13) || char(10) ) INTO headers
    FROM sa_http_header_info();
    SELECT CAST( sa_http_php_page_interpreted( '<?php phpinfo(); ?>',
        http_header( '@HttpMethod' ),
        http_header( '@HttpURI' ),
        http_header( '@HttpVersion' ),
        headers,
        HTTP_BODY() ) AS LONG VARCHAR);
END;
```

関連情報

Web サービス関数 [212 ページ]
HTTP_BODY 関数 [Web サービス] [391 ページ]

sa_http_php_page システムプロシージャ [1528 ページ]
sa_http_header_info システムプロシージャ [1526 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.48 sa_http_variable_info システムプロシージャ

HTTP 変数名と値を返します。

構文

```
sa_http_variable_info( [variable_parm] )
```

パラメータ

variable_parm

任意の VARCHAR(255) パラメータを使用して、HTTP 変数名を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
[名前]	VARCHAR(255)	HTTP 変数名。
Value	LONG VARCHAR	HTTP 変数の値。

備考

sa_http_variable_info システムプロシージャは、HTTP 変数名と値を返します。任意のパラメータを使用して変数名を指定しない場合、結果セットにはすべての変数の値が格納されます。

このプロシージャは、Web サービス内の HTTP 要求の処理中に呼び出された場合は、空でない結果セットを返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

Web サービスから呼び出された次の Web サービスプロシージャは、sa_http_variable_info システムプロシージャを示します。

```
CREATE OR REPLACE PROCEDURE User1.HTTPVariableExample()
RESULT ( html_string LONG VARCHAR )
BEGIN
    DECLARE myname VARCHAR(255);
    DECLARE myvalue LONG VARCHAR;
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    DECLARE curs CURSOR FOR
        SELECT Name, Value FROM sa_http_variable_info();
    MESSAGE '=== HTTP Variables ===' TO CONSOLE;
    OPEN curs;
    FetchLoop: LOOP
        FETCH next curs INTO myname, myvalue;
        IF SQLSTATE = err_notfound THEN
            LEAVE FetchLoop;
        END IF;
        MESSAGE myname, '=', myvalue TO CONSOLE;
    END LOOP FetchLoop;
    CLOSE curs;
END;
```

`?param1=value1&param2=value2&param3=value3` のような URL パラメータリストの場合、このサンプル Web サービスプロシージャからの出力は、パラメータ名 = value のような形式でデータベースサーバメッセージウィンドウに表示されます。

```
=== HTTP Variables ===
param1=value1
param3=value3
param2=value2
```

関連情報

Web サービス関数 [212 ページ]
NEXT_HTTP_VARIABLE 関数 [Web サービス] [455 ページ]
HTTP_VARIABLE 関数 [Web サービス] [399 ページ]
sa_http_header_info システムプロシージャ [1526 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.49 sa_index_density システムプロシージャ

インデックス内の断片化とスキューの量に関する情報をレポートします。

構文

```
sa_index_density(
[ tbl_name
[, owner_name ] ]
)
```

パラメータ

tbl_name

任意の CHAR(128) パラメータを使用して、テーブル名を指定します。デフォルトは NULL です。

owner_name

任意の CHAR(128) パラメータを使用して、所有者名を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
TableName	CHAR(128)	テーブルの名前。
TableId	UNSIGNED INTEGER	テーブル ID。
IndexName	CHAR(128)	インデックスの名前。
IndexId	UNSIGNED INTEGER	インデックスの ID。このカラムには、次のいずれかの値が含まれます。 **0** プライマリキーの場合 **SYSFKEY.foreign_key_id** 外部キーの場合 **SYSIDX.index_id** その他すべてのインデックスの場合

カラム名	データ型	説明
IndexType	CHAR(4)	インデックスタイプ。このカラムには、次のいずれかの値が含まれます。 **PKEY** プライマリキーの場合 **FKEY** 外部キーの場合 **UI** ユニークインデックスの場合 **UC** 一意性制約の場合 **NUI** ユニークでないインデックスの場合
LeafPages	UNSIGNED INTEGER	リーフページの数。
Density	DOUBLE	各インデックスページが (平均で) どの程度埋まっているかを示す 0 ～ 1 の間の小数。
Skew	DOUBLE	インデックス内の偏りのレベルを示す数。1 は完全にバランスの取れたインデックスであることを示します。値が大きくなるほどスキューも多くなることを示します。

備考

sa_index_density システムプロシージャは、インデックスの断片化とスキューの程度に関する情報を取得するときに使用します。多数のリーフページを持つインデックスの場合は、Density 値を大きくして不均衡値を小さくすることをお奨めします。

インデックスの密度は、インデックスページの平均の満杯度をパーセントとして反映しています。密度が 0.7 の場合、インデックスページに平均で 70% のインデックスデータが入っていることを示しています。インデックススキューは、平均密度からの一般的な偏差を反映しています。オプティマイザが選択性推定を行う場合、このスキュー量は重要です。

リーフページ数が少ない場合は、密度とスキューの値を考慮する必要はありません。密度とスキューの値は、リーフページ数が多いときだけ重要です。リーフページ数が多い場合、密度の値が低いと断片化を表し、不均衡値が高いとインデックスが偏っていることを示します。どちらも、パフォーマンス低下の要因になります。REORGANIZE TABLE 文を実行すると、この両方の問題に対処できます。

テーブルを指定しないでこのプロシージャを呼び出すと、データベース内のすべてのテーブルのすべてのインデックスに関する情報が返されます。

権限

次のシステム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

- MONITOR
- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY OBJECT
- DROP ANY OBJECT

関連する動作

なし

例

次の例は、sa_index_density システムプロシージャを使用して、データベースのすべてのインデックス内の断片化とスキューの量をまとめた結果セットを返します。

```
CALL sa_index_density( );
```

関連情報

REORGANIZE TABLE 文 [1266 ページ]

1.6.8.50 sa_index_levels システムプロシージャ

インデックス内のレベル数が報告され、パフォーマンスのチューニングに役立てることができます。

構文

```
sa_index_levels(
[ tbl_name
[, owner_name ] ]
)
```

パラメータ

tbl_name

任意の CHAR(128) パラメータを使用して、テーブル名を指定します。デフォルトは NULL です。

owner_name

任意の CHAR(128) パラメータを使用して、所有者名を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
TableName	CHAR(128)	テーブルの名前。
TableId	UNSIGNED INTEGER	テーブル ID。
IndexName	CHAR(128)	インデックスの名前。
IndexId	UNSIGNED INTEGER	インデックスの ID。このカラムには、次のいずれかが含まれます。 **0** プライマリキーの場合 **SYSFKEY.foreign_key_id** 外部キーの場合 **SYSIDX.index_id** その他すべてのインデックスの場合
IndexType	CHAR(4)	インデックスタイプ。このカラムには、次のいずれかの値が含まれます。 **PKEY** プライマリキーの場合 **FKEY** 外部キーの場合 **UI** ユニークインデックスの場合 **UC** 一意性制約の場合 **NUI** ユニークでないインデックスの場合
Levels	INTEGER	インデックス内のレベル数。

備考

インデックスツリー内のレベル数は、そのインデックスを使用しているローに対するアクセスに必要な I/O 操作の数を決定します。レベル数の少ないインデックスの方が、レベル数が多いインデックスよりも効率的です。

このプロシージャは、テーブル名、テーブル ID、インデックス名、インデックス ID、インデックスタイプ、インデックス内のレベル数で構成される結果セットを返します。

引数が指定されない場合は、データベースにあるすべてのインデックスのレベルが返されます。`tbl_name` のみを指定した場合、そのテーブルのすべてのインデックスのレベルが提示されます。`tbl_name` が NULL であり、`owner_name` が指定された場合は、そのユーザが所有しているテーブルにあるインデックスのレベルだけが返されます。

権限

次のシステム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY OBJECT
- DROP ANY OBJECT

関連する動作

なし

例

次の例では、sa_index_levels システムプロシージャを使用して、Products インデックス内のレベル数を返します。

```
CALL sa_index_levels( );
```

TableName	TableId	IndexName	...	Levels
Products	436	Products	...	1
...	...	...	...	...

関連情報

CREATE INDEX 文 [842 ページ]

1.6.8.51 sa_install_feature システムプロシージャ

空間機能の追加のように、追加機能をインストールします。

構文

```
sa_install_feature( feat_name )
```

パラメータ

feat_name

インストールする機能を識別する LONG VARCHAR パラメータです。デフォルトは NULL です。サポートされている機能名は次のとおりです。

値	説明
st_geometry_predefined_uom	新しいデータベースに、デフォルトではインストールされない事前定義の測定単位をインストールします。
st_geometry_predefined_srs	新しいデータベースに、デフォルトではインストールされない事前定義の空間参照系と測定単位をインストールします。
st_geometry_compat_func	一連の空間互換関数をインストールします。これらの関数は、空間メソッドの代わりに使用できます。

機能名の定義は、*%SQLANY17%*¥scripts スクリプトディレクトリにある st_geometry_config.tgz ファイルに指定されています。このファイルが削除されている場合に、このファイルに依存する機能をインストールしようとすると、エラーが返されます。

備考

feat_name 値を問い合わせて、インストールされる機能を確認できます。たとえば、次のクエリでは、st_geometry_predefined_uom にインストールされる測定単位が返されます。

```
SELECT * FROM st_geometry_predefined_uom( 'CREATE' );
```

また、上記の例ではパラメータ名も示されるため、WHERE 句を使用して特定の値を問い合わせることができます。たとえば、次の文では chain 測定単位の unit_name パラメータを問い合わせています。

```
SELECT * FROM st_geometry_predefined_uom( 'CREATE' ) WHERE unit_name='chain';
```

unit_name	unit_type	conversion_factor	...
chain	LINEAR	20.1168	...

次のクエリでは、foot に基づくすべての測定単位が返されます。

```
SELECT * FROM st_geometry_predefined_uom() WHERE unit_name LIKE '%foot%';
```

インストールされる空間参照系を調べるには、次のクエリを使用します。

```
SELECT * FROM st_geometry_predefined_srs();
```

次の文は、**organization** と **organization_coordsys_id** 別の空間参照系を問い合わせます。

```
SELECT * FROM st_geometry_predefined_srs() WHERE organization='EPSG' AND
organization_coordsys_id=2295;
```

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

st_geometry_predefined_uom と st_geometry_predefined_srs の場合は、MANAGE ANY SPATIAL OBJECT システム権限を持っている必要もあります。

st_geometry_compat_func の場合は、MANAGE ANY OBJECT PRIVILEGE、CREATE ANY PROCEDURE、SELECT ANY TABLE のシステム権限を持っている必要もあります。

sa_install_feature の場合は、MANAGE ANY SPATIAL OBJECT システム権限を持っている必要もあります。

例

次の文では、新しいデータベースに、デフォルトではインストールされないすべての事前定義の測定単位をインストールします。

```
CALL sa_install_feature( 'st_geometry_predefined_uom' );
```

次の文は、空間メソッドの代替として使用できる空間互換機能セットをインストールします。

```
CALL sa_install_feature( 'st_geometry_compat_func' );
```

1.6.8.52 sa_java_loaded_classes システムプロシージャ

データベースサーバによって Java VM に現在ロードされているクラスをリストします。

構文

sa_java_loaded_classes( )

結果セット

カラム名	データ型	説明
class_name	VARCHAR(512)	データベースサーバによって Java VM に現在ロードされているクラスの名前。

備考

データベースサーバによって Java VM に現在ロードされているすべての Java クラスの名前を含む結果セットを返します。

このプロシージャは、不足しているクラスを調べるのに便利です。また、特定のアプリケーションでどの jar ファイルのクラスが使用されているかを調べることもできます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、init カバー関数を呼び出して、サンプル Invoice Java クラスの init メソッドをロードして呼び出します。次に、sa_java_loaded_classes を呼び出して、ロードされたすべての Java クラスのリストを取得します。

```
CALL init( 'Work boots', 79.99, 'Hay fork', 37.49 );
CALL sa_java_loaded_classes( );
```

1.6.8.53 sa_list_cursors システムプロシージャ

現在の接続で使用されているカーソルのリストを返します。

構文

```
sa_list_cursors( )
```

結果セット

カラム名	データ型	説明
handle	UNSIGNED INTEGER	カーソルを識別するユニークなハンドル。
scope	INTEGER	カーソルが開いている呼び出しスタックのスコープ。
cursor_name	VARCHAR(128)	カーソル名。
is_open	BIT	カーソルが現在開いている (1) かどうかを示すインジケータ。
is_pinned	BIT	カーソルが再利用に備えて現在メモリで固定されている (1) かどうかを示すインジケータ。
fetch_count	UNSIGNED BIGINT	カーソルからフェッチされたローの数。

備考

sa_list_cursors システムプロシージャは、CALL 文、または SELECT 文の FROM 句で使用できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、接続のオープンカーソルのリストを返します。

```
CALL sa_list_cursors();
```

関連情報

sa_copy_cursor_to_temp_table システムプロシージャ [1475 ページ]
sa_describe_cursor システムプロシージャ [1488 ページ]

1.6.8.54 sa_list_statements システムプロシージャ

現在の接続で使用されている文のリストを返します。

構文

sa_list_statements( )

結果セット

カラム名	データ型	説明
handle	UNSIGNED INTEGER	文を識別するユニークなハンドル。
statement_number	INTEGER	接続内の文の番号。
num_client_prepares	UNSIGNED INTEGER	クライアントが同一の文テキストを準備した回数。
num_opens	UNSIGNED INTEGER	文がオープンされた回数。
schema_version_on_create	UNSIGNED INTEGER	内部でのみ使用。
dropped_by_app	BIT	文がクライアントにより削除されたが、キャッシュ用に保持されている場合、値は 1 です。それ以外の場合、値は 0 です。
invalid_cached_statement	BIT	キャッシュされた文だが有効ではなくなった (スキーマの変更などにより) 場合、値は 1 です。それ以外の場合、値は 0 です。
SQLStatement	LONG VARCHAR	文のテキスト。

備考

sa_list_statements システムプロシージャは、CALL 文、または SELECT 文の FROM 句で使用できます。
sa_list_statements システムプロシージャを実行する文は、結果に含められません。

SQLStatement カラムは、REWRITE 関数と同様の方法でセマンティック変形の最適化および正規化が行われるため、元のテキストから書き換えられます。暗号化キーやパスワードなどの機密情報は、*** に置き換えられます。これらの変更のため、アプリケーション内の文 (形式が異なる可能性があります) と比較するときは SQLStatement 値を解釈する必要があります。

権限

なし

関連する動作

なし

例

次の例では、接続の文のリストを返します。

```
CALL sa_list_statements();
```

次の例では、max_statement_count リソースガバナに提供される文のリストを返します。

```
SELECT *
FROM sa_list_statements()
WHERE dropped_by_app=0;
```

1.6.8.55 sa_load_cost_model システムプロシージャ

現在のコストモデルを、指定したファイルに格納されているコストモデルで置換します。

構文

```
sa_load_cost_model( file_name )
```

パラメータ

file_name

この CHAR(1024) パラメータを使用して、ロードするコストモデルファイルの名前を指定します。

備考

オプティマイザは、コストモデルを使用して、クエリの最適なアクセスプランを決定します。データベースサーバは、各データベースのコストモデルを管理します。データベースのコストモデルは、ALTER DATABASE 文の CALIBRATE SERVER 句を使用して、任意のタイミングで再調整できます。たとえば、データベースを非標準ハードウェアに移動する場合、コストモデルを再調整できます。

sa_load_cost_model システムプロシージャを使用すると、ファイル (`file_name`) に保存されているコストモデルをロードできます。コストモデルをロードすると、データベースの現在のコストモデルが置換されます。

注記

sa_unload_cost_model システムプロシージャは、sa_load_cost_model がロードするファイルに CALIBRATE PARALLEL READ 情報を含めません。

大量に同一ハードウェアのインストールがある場合、sa_load_cost_model システムプロシージャを使用すると、繰り返しの時間がかかる再調整動作を省略できます。

新規コストモデルをロードする場合、データベースを排他的に使用する必要があります。

コストモデルをロードするときは、類似するハードウェアにあるデータベースのモデルが生成されていないかどうかを検討します。大幅に異なるハードウェアに格納されているデータベースからコストモデルをロードすると、アクセスプランが非効率なため、パフォーマンスが低下する可能性があります。

コストモデルは、sa_unload_cost_model システムプロシージャを使用してファイルに保存されます。

権限

ALTER DATABASE および LOAD ANY TABLE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

データベースサーバは新規コストモデルをロードした後に COMMIT を実行します。

例

次の例は、costmodel8 というファイルからコストモデルをロードします。

```
CALL sa_load_cost_model( 'costmodel8' );
```

関連情報

ALTER DATABASE 文 [630 ページ]
sa_unload_cost_model システムプロシージャ [1649 ページ]

1.6.8.56 sa_locks システムプロシージャ

データベース内のすべてのロック (ニューテックスを含む) を表示します。

構文

```
sa_locks(
   [ connection
   [, creator
   [, table_name
   [, max_locks
   [, object_type ] ] ] ] ]
)
```

パラメータ

connection

任意の INTEGER パラメータを使用して、接続 ID 番号を指定します。プロシージャは、指定された接続に関するロック情報のみを返します。デフォルト値は 0 (NULL) で、この場合は全接続の情報が返されます。

creator

任意の CHAR(128) パラメータを使用して、ユーザ ID を指定します。プロシージャは、指定の所有者が所有するオブジェクトに関する情報のみを返します。creator パラメータのデフォルト値は NULL です。このパラメータが NULL に設定されている場合、sa_locks は次の情報を返します。

- `table_name` を指定しない場合、データベース内のすべてのオブジェクトのロック情報が返されます。
- `table_name` を指定した場合、現在のユーザが作成したテーブルのうち、指定された名前を持つテーブルのロック情報が返されます。

table_name

任意の CHAR(128) パラメータを使用して、オブジェクト名を指定します。プロシージャは、指定されたオブジェクトに関するロック情報のみを返します。デフォルト値は NULL で、この場合はすべてのオブジェクトの情報が返されます。

max_locks

任意の INTEGER パラメータを使用して、情報が返されるロックの最大数を指定します。デフォルト値は 1000 です。値 -1 では、すべてのロック情報が返されます。

object_type

任意の CHAR(5) パラメータを使用して、結果をロックに関連付けられたオブジェクトのタイプに制限します。すべてのオブジェクトタイプのロック情報を返すには、*ALL* を指定します。テーブル、グローバルテンポラリテーブル、マテリアライズドビューのロック情報を返す場合は、*TABLE* を指定します。ミューテックス情報を返す場合は、*MUTEX* を指定します。`object_type` を指定しない場合、プロシージャはすべてのオブジェクトタイプのロック情報を返します。

結果セット

カラム名	データ型	説明
conn_name	VARCHAR(128)	現在の接続の名前。
conn_id	INTEGER	接続 ID 番号。
user_id	CHAR(128)	接続のユーザ ID。
table_type	CHAR(6)	ロックに関連付けられたオブジェクトのタイプ。考えられる値は、次のとおりです。 • BASE (テーブル) • GLBTMP (グローバルテンポラリテーブル) • MVIEW (マテリアライズドビュー) • MUTEX (ミューテックス)
creator	VARCHAR(128)	オブジェクトの所有者。
table_name	VARCHAR(128)	ロックが保持されているオブジェクト。
index_id	INTEGER	インデックス ID または NULL。
lock_class	CHAR(16)	ロッククラス。考えられる値は、次のとおりです。*Schema*、*Row*、*Table*、*Position*、または NULL (ミューテックスの場合)。
lock_duration	CHAR(12)	ロックの継続期間。考えられる値は、次のとおりです。*Transaction*、*Position*、または *Connection*。
lock_type	CHAR(16)	ロックのタイプ。
row_identifier	UNSIGNED BIGINT	ローの識別子。8 バイトのロー識別子または NULL です。

備考

結果セットの lock_type カラムに含まれる情報は、テーブルロックにのみ関連しており、lock_class カラムで指定されたテーブルロック分類に応じて異なります。これらの 2 つのカラムの値の関係について理解するには、このテーブルを使用します。

ロッククラス	ロック種類	コメント
Schema	Shared (共有スキーマロック) Exclusive (排他スキーマロック)	スキーマロックの場合、row_identifier とインデックス ID 値は NULL です。

ロッククラス	ロック種類	コメント
Row	Read (読み取りロック) Intent (意図的ロック) ReadPK (読み取りロック) Write (書き込みロック) WriteNoPK (書き込みロック) Surrogate (代理ロック)	ローの読み取りロックは、短期のロック (独立性レベル 1 でスキャン) にするか、高い独立性レベルの長期ロックにできます。lock_duration カラムは、読み取りロックがカーソル安定性 (*Position*) のために短期になるか、COMMIT/ROLLBACK (*Transaction*) まで保持される長期になるかを指定します。ローのロックは、常に特定のローに保持されます。この 8 バイトのロー識別子は、row_identifier カラムの 64 ビット整数値としてレポートされます。代理ロックは、ローロックの特殊なケースです。代理ロックは代理エントリで保持されます。代理エントリは、参照整合性のチェックが遅延するときに作成されます。 テーブルで作成されるすべての代理エントリについてユニークな代理ロックはありません。ただし、代理ロックは、指定した接続で指定されるテーブルで作成される代理エントリのセットに対応します。row_identifier 値は、代理ロックに関連付けられているテーブルと接続ごとにユニークです。 必要に応じて、ローのキーとキーではない部分を別々にロックできます。1 つの接続で共有 (読み取り) アクセス用にローのキー部分に対する読み取りロックを取得して、他の接続でローのキーではない他のカラムに対する書き込みロックを取得できるようにできます。ローのキーではないカラムを更新しても、そのローを参照する外部ローの挿入と削除は妨げられません。
テーブル	Shared (共有テーブルロック) Intent (テーブルロックを更新する意図) Exclusive (排他テーブルロック)	なし。

ロッククラス	ロック種類	コメント
Position	Phantom (幻ロック) Insert (挿入ロック)	通常、位置ロックも特定のローに保持され、そのローの 64 ビットロー識別子は結果セットの row_identifier カラムに表示されます。ただし、row_identifier カラムが NULL の場合、位置ロックはすべてのスキャン (インデックススキャンまたは逐次スキャン) に保持されます。 位置ロックは、逐次テーブルスキャンまたはインデックススキャンと関連付けられます。index_id カラムは、位置ロックが逐次スキャンに関連付けられているかどうかを示します。逐次スキャンのために位置ロックを保持する場合、index_id カラムは NULL です。位置ロックが特定のインデックススキャンの結果として維持される場合、そのインデックスのインデックス識別子は、index_id カラムにリストされます。インデックス識別子は、SYSIDX ビューを使用して表示できる ISYSIDX システムテーブルのプライマリキーに対応します。位置ロックがすべてのインデックスのスキャンで保持される場合、インデックス ID 値は -1 です。

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次のクエリを実行して、ロックを識別できます。

```
CALL sa_locks( );
```

ロックを保持している接続、ロックの期間、ロックの種類タイプに関する情報など、現在データベースに保持されているロックを表示するには、sa_locks システムプロシージャを使用します。join 述部でテーブルの ROWID を使用することで、sa_locks システムプロシージャの結果を特定のテーブルに結合するクエリを実行します。

```
SELECT S.conn_id, S.user_id, S.lock_class, S.lock_type, E.*
  FROM sa_locks() S JOIN Employees E WITH( NOLOCK )
```

```
    ON ROWID(E) = S.row_identifier
  WHERE S.table_name = 'Employees';
```

sa_locks システムプロシージャの結果セットには、ロックが参照するテーブルのローを識別できる row_identifier カラムが含まれています。WITH NOLOCK 句を指定する必要はない場合があります。ただしクエリが 0 以外の独立性レベルで発行された場合、ロックが解放されるまでこのクエリがブロックされることがあり、この確認方法の便利さは低下してしまいます。

関連情報

SYSIDX システムビュー [1810 ページ]

1.6.8.57 sa_make_object システムプロシージャ (廃止予定)

ALTER 文を実行する前に、オブジェクトのスケルトンインスタンスが存在することを確認します。

構文

```
sa_make_object(
objtype
, objname
[, owner
[, tabname ]  ]
)
```

```
objtype:
'procedure'
| 'function'
| 'view'
| 'trigger'
| 'service'
| 'event'
```

パラメータ

objtype

この CHAR(30) パラメータを使用して、作成されるオブジェクトのタイプを指定します。objtype が 'trigger' の場合、この引数はトリガが作成されるテーブルの所有者を示します。

objname

この CHAR(128) パラメータを使用して、作成されるオブジェクトの名前を指定します。

owner

任意の CHAR(128) パラメータを使用して、作成されるオブジェクトの所有者を指定します。デフォルトは NULL です。

tabname

この CHAR(128) パラメータは、objtype が 'trigger' である場合にのみ必要です。トリガを作成するテーブルの名前を指定するときに使用します。デフォルトは NULL です。

備考

このプロシージャは、データベーススキーマを作成または変更するために繰り返し実行されるスクリプトで使用できますが、その使用は廃止される予定です。作成または変更する型オブジェクトには、可能であれば常に CREATE OR REPLACE 文の使用が推奨されるためです。CREATE OR REPLACE を使用したほうが効率的であり、すでに存在するオブジェクトを作成しようとするときの動作が正しくなります。

sa_make_object システムプロシージャを使用する場合、通常はこのシステムプロシージャの後にオブジェクト定義全体を含む ALTER 文を実行します。

権限

次のように、他の権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

呼び出し側が所有するプロシージャまたは関数

CREATE PROCEDURE システム権限、CREATE ANY PROCEDURE システム権限、または CREATE ANY OBJECT システム権限
他のユーザが所有するプロシージャまたは関数

CREATE ANY PROCEDURE システム権限または CREATE ANY OBJECT システム権限
サービス

MANAGE ANY WEB SERVICE システム権限
イベント

MANAGE ANY EVENT システム権限または CREATE ANY OBJECT システム権限
呼び出し側が所有するビュー

CREATE VIEW システム権限、CREATE ANY VIEW システム権限、または CREATE ANY OBJECT システム権限
他のユーザが所有するビュー

CREATE ANY VIEW システム権限または CREATE ANY OBJECT システム権限
トリガ

自分が所有するテーブルにトリガがある場合は、CREATE ANY TRIGGER システム権限または CREATE ANY OBJECT システム権限のいずれかが必要です。

他のユーザが所有するテーブルにトリガがある場合は、CREATE ANY TRIGGER システム権限または CREATE ANY OBJECT システム権限のいずれかが必要です。さらに、次のいずれかのシステム権限が必要です。

- ALTER ANY TABLE 権限
- ALTER ANY OBJECT システム権限
- トリガが作成されたテーブルの ALTER 権限

関連する動作

オートコミット

例

次の文は、スケルトンプロシージャ定義が作成されていることを確認し、プロシージャを定義し、プロシージャに対する権限を付与します。これらの命令が記述されたスクリプトファイルは、データベースに対して繰り返し実行でき、エラーは起こりません。

```
CALL sa_make_object( 'procedure', 'myproc' );
ALTER PROCEDURE myproc( in p1 INT, in p2 CHAR(30) )
BEGIN
    // ...
END;
GRANT EXECUTE ON myproc TO public;
```

次の例は、sa_make_object システムプロシージャを使用して、スケルトン Web サービスを追加します。

```
CALL sa_make_object( 'service', 'my_web_service' );
```

関連情報

ALTER EVENT 文 [643 ページ]
ALTER FUNCTION 文 [649 ページ]
ALTER PROCEDURE 文 [669 ページ]
ALTER SERVICE 文 [HTTP Web サービス] [683 ページ]
ALTER SERVICE 文 [SOAP Web サービス] [688 ページ]
ALTER TRIGGER 文 [730 ページ]
ALTER VIEW 文 [734 ページ]

1.6.8.58 sa_materialized_view_can_be_immediate システムプロシージャ

指定されたマテリアライズドビューを即時ビューとして定義できるかどうかを返します。

構文

```
sa_materialized_view_can_be_immediate(
view_name
, owner_name
)
```

パラメータ

view_name

この CHAR(128) パラメータを使用して、マテリアライズドビューの名前を指定します。`view_name` が NULL の場合、エラーが返されます。

owner_name

この CHAR(128) パラメータを使用して、マテリアライズドビューの所有者を指定します。`owner_name` が NULL の場合は、現在のユーザ ID が使用されます。

結果セット

カラム名	データ型	説明
SQLStateVal	CHAR(6)	返される SQLSTATE。
ErrorMessage	LONG VARCHAR	SQLSTATE に対応するエラーメッセージ。

備考

指定された手動ビューを即時ビューに変更できるかどうかについては、制限があります。このシステムプロシージャは、変更が許可されるかどうかを確認するときに使用します。

結果セット内のそれぞれのローが、ビューについて返される 1 つの SQLSTATE に対応します。そのため、マテリアライズドビュー定義が複数の制限に違反している場合は、そのビューの複数のローが結果に格納されます。

このシステムプロシージャの出力を sa_materialized_view_info システムプロシージャの出力と結合すると、ビューのステータスに関する情報と、ビューを即時ビューに変更できるかどうかの情報を取得できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、マテリアライズドビューの所有者であるか、ALTER ANY MATERIALIZED VIEW または ALTER ANY OBJECT のシステム権限を持っていることが必要です。

関連する動作

指定したマテリアライズドビューのすべてのメタデータと依存性は、サーバのキャッシュにロードされます。

例

次の文を実行し、手動ビュー view10 を作成してそのビューをリフレッシュします。

```
CREATE MATERIALIZED VIEW view10
   AS (SELECT C.ID, C.Surname, sum(P.UnitPrice) as revenue, C.CompanyName,
SO.OrderDate
         FROM Customers C, SalesOrders SO, SalesOrderItems SOI, Products P
         WHERE C.ID = SO.CustomerID
         AND SO.ID = SOI.ID
         AND P.ID = SOI.ProductID
         GROUP BY C.ID, C.Surname, C.CompanyName, SO.OrderDate);
REFRESH MATERIALIZED VIEW view10;
```

次のクエリを使用して、ビュー view10 を即時ビューに変更できない理由を検索します。

```
SELECT SQLStateVal AS "SQLstate", ErrorMessage AS Description
   FROM sa_materialized_view_can_be_immediate( 'view10', NULL )
   ORDER BY SQLSTATE;
```

関連情報

sa_materialized_view_info システムプロシージャ [1554 ページ]

1.6.8.59 sa_materialized_view_info システムプロシージャ

指定されたマテリアライズドビューに関する情報を返します。

構文

```
sa_materialized_view_info(
[ view_name
[, owner_name ] ]
)
```

パラメータ

view_name

任意の CHAR(128) パラメータを使用して、情報を返す対象のマテリアライズドビューの名前を指定します。デフォルトは NULL です。

owner_name

任意の CHAR(128) パラメータを使用して、マテリアライズドビューの所有者を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
OwnerName	CHAR(128)	ビューの所有者。
ViewName	CHAR(128)	ビューの名前です。
Status	CHAR(1)	ビューに関するステータス情報です。考えられる値は、次のとおりです。 **D** 無効 **E** 有効
DataStatus	CHAR(1)	ビュー内のデータに関するステータス情報です。考えられる値は、次のとおりです。 **E** 最後にリフレッシュしようとしたときにエラーが発生しました。ビューは有効ですが、初期化されていません。 **F** 最後のリフレッシュ以降、基本となるテーブルが変更されていません。ビューは新しいものと見なされます。ビューは有効であり、初期化されています。 **N** ビューは初期化されていません。これは、次のいずれかに当てはまるときに発生します。 • ビューが作成されて以降、リフレッシュされていない • ビューからデータがトランケートされている • ビューが無効である **S** 最後のリフレッシュ以降、基本となるテーブルが変更されています。ビューは古いものと見なされます。ビューは有効であり、初期化されています。
ViewLastRefreshed	TIMESTAMP	ビューを最後にリフレッシュしたデータベースサーバのローカル日時。ViewLastRefreshed の値が NULL の場合、ビューは初期化されていません。この値はシミュレートされたタイムゾーンの影響を受けます。
DataLastModified	TIMESTAMP	古いビューの場合、基本となるデータを最後に修正したデータベースサーバのローカル日時。この値はシミュレートされたタイムゾーンの影響を受けます。 初期化されていないビューまたは古いと見なされないビューの場合、この値は NULL です。

カラム名	データ型	説明
AvailForOptimization	CHAR(1)	オプティマイザによるビューの使用可能性に関する情報。考えられる値は、次のとおりです。 **D** オプティマイザによる使用は無効です。ビューの所有者は、オプティマイザがビューを使用することを許可していません。 **I** ビューの定義が必要な条件を満たしていないなど、何らかの内部的な理由によって、オプティマイザがビューを使用できません。ただし、所有者は、オプティマイザがビューを使用することを明示的に禁止していません。 **N** リフレッシュが実行されていないか、または失敗したために、ビューにデータがありません。ビューの所有者の許可により、オプティマイザはビューを使用できますが、ビューが初期化されていません。 **O** 現在の接続に互換性のないオプション値があります。オプティマイザはビューを使用でき、ビューの定義は必要な条件をすべて満たしていますが、現在のオプション設定が、ビューの作成に使用されたオプション設定と互換性がありません。 **Y** オプティマイザはビューを使用できます。ビューの所有者は、オプティマイザがビューを使用することを許可しています。また、ビュー定義は、オプティマイザが使用するために必要なすべての条件を満たしています。
RefreshType	CHAR(1)	ビューのリフレッシュタイプ。考えられる値は、次のとおりです。 **I** ビューは即時ビューです。即時ビューは、基本となるテーブルのデータへの変更が、マテリアライズドビューのデータに影響を与えるとすぐにリフレッシュされます。 **M** ビューは手動ビューです。手動ビューは、たとえば、REFRESH MATERIALIZED VIEW 文を使用するか、または sa_refresh_materialized_views システムプロシージャを使用して手動でリフレッシュします。

備考

`view_name` と `owner_name` のどちらも指定されていない場合、またはどちらも NULL の場合、データベースに含まれるすべてのマテリアライズドビューに関する情報が返されます。

`owner_name` を指定しない場合、または NULL の場合は、名前が `view_name` であるすべてのマテリアライズドビューに関する情報が返されます。

このプロシージャは、ビューの定義に問題があるために、オプティマイザによって考慮されないマテリアライズドビューのリストを決定する場合に便利です。このようなマテリアライズドビューの場合、AvailForOptimization 値は I です。

次の表は、AvailForOptimization プロパティがどのように決定されるかを示しています。左のカラムからローを横に読んでいき、考慮する必要のある条件を確認して最終的に AvailForOptimization カラムの値を決定します。

ビューを最適化に使用することをユーザが許可していますか	ビュー定義は使用に必要な条件をすべて満たしていますか	接続オプションはビューを使用するために必要なオプションと一致していますか	ビューは初期化されていますか	AvailForOptimization 値
はい	はい	はい	はい	Y
いいえ	該当なし	該当なし	該当なし	D
はい	いいえ	該当なし	はい	I
はい	該当なし	該当なし	いいえ	N
Yes	はい	いいえ	はい	O

初期化されたマテリアライズドビューが空であることがあります。これは、基本となるテーブルに、マテリアライズドビュー定義を満たすデータがない場合に発生します。空のビューは、同様にデータを持たない初期化されていないマテリアライズドビューと同じであるとは見なされません。ViewLastRefreshed プロパティの値を使用すると、ビューが初期化されていないのか (NULL)、または基本となるテーブルのデータによって空であるのか (NOT NULL) を区別できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

指定したマテリアライズドビューのすべてのメタデータと依存性は、データベースサーバのキャッシュにロードされます。

例

次の文は、データベース内のすべてのマテリアライズドビューに関する情報を返します。

```
SELECT *
   FROM sa_materialized_view_info();
```

sa_materialized_view_info システムプロシージャの結果と sa_materialized_view_can_be_immediate システムプロシージャの結果を組み合わせると、ステータス情報だけでなく、ビューを即時ビューに変更できるかどうかも返すことができます。次の文を実行し、この例で検査されるマテリアライズドビューを作成します。

```
CREATE MATERIALIZED VIEW view0 AS (
   SELECT ID, Name, Description, Size
   FROM Products
   WHERE Quantity > 0 );
CREATE UNIQUE INDEX u_view0
   ON view0( ID );
ALTER MATERIALIZED VIEW view0
```

```
   IMMEDIATE REFRESH;
CREATE MATERIALIZED VIEW view00 AS (
   SELECT ID, Name, Description, Size
   FROM Products
   WHERE Quantity <= 0 );
CREATE UNIQUE INDEX u_view00
   ON view00( ID );
CREATE MATERIALIZED VIEW view1 AS (
   SELECT ID, Name, Description, Size
   FROM Products
   WHERE Quantity = 0 );
ALTER MATERIALIZED VIEW view1
   DISABLE;
CREATE MATERIALIZED VIEW view100
   AS (SELECT C.ID, C.Surname, sum(P.UnitPrice) as revenue, C.CompanyName,
SO.OrderDate
         FROM Customers C, SalesOrders SO, SalesOrderItems SOI, Products P
         WHERE C.ID = SO.CustomerID
         AND SO.ID = SOI.ID
         AND P.ID = SOI.ProductID
         GROUP BY C.ID, C.Surname, C.CompanyName, SO.OrderDate);
REFRESH MATERIALIZED VIEW view100;
```

次の文を実行すると、所有しているビューのステータス情報と適格性情報を返します。

```
SELECT ViewName, Status, ViewLastRefreshed, AvailForOptimization, RefreshType,
CanBeImmediate
FROM sa_materialized_view_info() AS V,
   LATERAL( SELECT LIST( ErrorMessage, '' )
            FROM sa_materialized_view_can_be_immediate( V.ViewName,
V.OwnerName ) ) AS I( CanBeImmediate )
WHERE OwnerName  = USER_NAME();
```

ViewName	Status	ViewLastRefreshed	AvailForOptimization	RefreshType	CanBeImmediate
view0	E	(NULL)	N	I	
view00	E	(NULL)	N	M	
view1	D	(NULL)	N	M	Cannot use view 'view1' because it has been disabled

ViewName	Status	ViewLastRefreshed	AvailForOptimization	RefreshType	CanBeImmediate
view100	E	2008-02-12 16:47:00.000	Y	M	The materialized view view100 cannot be changed to immediate because it has already been initialized. The materialized view view100 cannot be changed to immediate because it does not have a unique index on non-nullable columns. The materialized view cannot be changed to immediate because COUNT(*) is required to be part of the SELECT list. The materialized view cannot be changed to immediate because it does not have a unique index on non-aggregate non-nullable columns.

結果は、次のことを示しています。

- view0 がリフレッシュされていない即時ビューです。
- view00 がリフレッシュされていない手動ビューです。

- view1 は無効です。
- view100 は手動ビューであり、最後にリフレッシュされたのが 2008-02-12 16:47:00.000 です。
- CanBeImmediate カラムにエラーメッセージがないため、view00 を即時ビューに変更できます。
- CanBeImmediate カラムにリストされている理由のため、view1 と view100 を即時ビューに変更できません。

関連情報

sa_materialized_view_can_be_immediate システムプロシージャ [1552 ページ]

1.6.8.60 sa_migrate システムプロシージャ

SQL Anywhere データベースにリモートテーブルセットをマイグレートします。

構文

```
sa_migrate(
base_table_owner
, server_name
[, table_name
[, owner_name
[, database_name
[, migrate_data
[, drop_proxy_tables
[, migrate_fkeys ] ] ] ] ] ]
)
```

パラメータ

base_table_owner

この VARCHAR(128) パラメータを使用して、ターゲットデータベースでマイグレート後のテーブルを所有するユーザを指定します。GRANT CONNECT 文を使用してこのユーザを作成します。このパラメータの値は必須です。

server_name

この VARCHAR(128) パラメータを使用して、リモートデータベースへの接続に使用されるリモートサーバの名前を指定します。CREATE SERVER 文を使用してこのサーバを作成します。このパラメータの値は必須です。

table_name

単一のテーブルをマイグレートする場合は、任意の VARCHAR(128) パラメータを使用してテーブルの名前を指定します。それ以外の場合は、このパラメータに NULL (デフォルト) を指定します。table_name と owner_name の両方のパラメータに NULL を指定しないでください。

owner_name

1 人の所有者に属するテーブルのみをマイグレートする場合は、任意の VARCHAR(128) パラメータを使用して所有者の名前を指定します。それ以外の場合は、このパラメータに NULL (デフォルト) を指定します。table_name と owner_name の両方のパラメータに NULL を指定しないでください。

database_name

任意の VARCHAR(128) パラメータを使用して、リモートデータベースの名前を指定します。リモートサーバ上の 1 つのデータベースのみからテーブルをマイグレートする場合は、データベース名を指定します。それ以外の場合は、このパラメータに NULL (デフォルト) を使用します。

migrate_data

任意の BIT パラメータを使用して、リモートテーブルのデータをマイグレートするかどうかを指定します。このパラメータは、0 (データをマイグレートしない) または 1 (データをマイグレートする) に設定できます。デフォルトでは、データがマイグレートされます (1 がデフォルト)。

drop_proxy_tables

任意の BIT パラメータを使用して、マイグレーションの完了後に、マイグレーションプロセス用に作成されたプロキシテーブルを削除するかどうかを指定します。このパラメータは、0 (プロキシテーブルを削除しない) または 1 (プロキシテーブルを削除する) に設定できます。デフォルトでは、プロキシテーブルが削除されます (1 がデフォルト)。

migrate_fkeys

任意の BIT パラメータを使用して、外部キーマッピングをマイグレートするかどうかを指定します。このパラメータは、0 (外部キーマッピングをマイグレートしない) または 1 (外部キーマッピングをマイグレートする) に設定できます。デフォルトでは、外部キーマッピングがマイグレートされます (1 がデフォルト)。

備考

このプロシージャを使用して、リモートの Oracle、IBM DB2、Microsoft SQL Server、Adaptive Server Enterprise、または SQL Anywhere のデータベースから SQL Anywhere にテーブルをマイグレートできます。このプロシージャでは、1 回の操作だけで、指定したサーバから外部キーマッピングを含めたリモートテーブルセットをマイグレートできます。sa_migrate システムプロシージャは、次のシステムプロシージャを呼び出します。

- sa_migrate_create_remote_table_list
- sa_migrate_create_tables
- sa_migrate_data
- sa_migrate_create_remote_fks_list
- sa_migrate_create_fks
- sa_migrate_drop_proxy_tables

マイグレーションを柔軟に行う必要がある場合は、sa_migrate の代わりにこれらのシステムプロシージャを使用できます。たとえば、別のユーザが所有するテーブルとの外部キー関係を持つテーブルをマイグレートする場合は、sa_migrate を使用すると外部キー関係を保持できません。

テーブルをマイグレートするためには、まず CREATE SERVER 文を使用してリモートデータベースに接続するためのリモートサーバを作成します。また、CREATE EXTERNLOGIN 文を使用して、リモートデータベースへの外部ログインを作成する必要がある場合もあります。

`base_table_owner` と `server_name` の各パラメータを指定するだけで、リモートデータベースのすべてのテーブルを SQL Anywhere データベースにマイグレートできます。ただし、これら 2 つのパラメータのみを指定した場合、ターゲットの SQL Anywhere データベースでは、マイグレートされたすべてのテーブルが 1 人の所有者に属することになります。リモート

データベースの各テーブルの所有者が異なり、SQL Anywhere データベースでもこれらのテーブルに異なる所有者を指定する場合は、所有者ごとに個別にテーブルをマイグレートし、sa_migrate プロシージャを呼び出すたびに base_table_owner パラメータと owner_name パラメータを指定します。

警告

table_name と owner_name の両方のパラメータに NULL を指定しないでください。table_name パラメータと owner_name パラメータの両方に NULL を指定すると、データベース内のシステムテーブルを含む、すべてのテーブルがマイグレートします。また、リモートデータベースで同じ名前で所有者が異なるテーブルは、ターゲットデータベースでは 1 人の所有者に属します。一度に移行するテーブルは、1 人の所有者に関連付けられたテーブルだけにしてください。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、次のシステム権限が必要です。

- CREATE TABLE または CREATE ANY TABLE (ベーステーブルの所有者でない場合)
- SELECT ANY TABLE (ベーステーブルの所有者でない場合)
- INSERT ANY TABLE (ベーステーブルの所有者でない場合)
- ALTER ANY TABLE (ベーステーブルの所有者でない場合)
- CREATE ANY INDEX (ベーステーブルの所有者でない場合)
- DROP ANY TABLE (ベーステーブルの所有者でない場合)

関連する動作

なし

例

次の文は、リモートデータベースからユーザ DBA に属するすべてのテーブルを外部キーマッピングとともにマイグレートし、リモートテーブル内のデータをマイグレートします。また、移行の完了時にプロキシテーブルを削除します。この例では、移行するすべてのテーブルが、SQL Anywhere ターゲットデータベースの LocalUser に属します。

```
CALL sa_migrate( 'LocalUser', 'RemoteSA', NULL, 'DBA', NULL, 1, 1, 1 );
```

次の文は、ユーザ DBA に属するすべてのテーブルをリモートデータベースからマイグレートします。対象の SQL Anywhere データベースで、これらのテーブルはユーザ LocalUser に属します。マイグレート中に作成されたプロキシテーブルは完了時に削除されず、LocalUser に属します。

```
CALL sa_migrate( 'LocalUser', 'RemoteSA', NULL, 'DBA', NULL, 1, 0, 1 );
```

関連情報

CREATE EXTERNLOGIN 文 [816 ページ]
CREATE SERVER 文 [914 ページ]
GRANT 文 [1131 ページ]
sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
sa_migrate_create_tables システムプロシージャ [1568 ページ]
sa_migrate_data システムプロシージャ [1569 ページ]
sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
sa_migrate_create_fks システムプロシージャ [1563 ページ]
sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]

1.6.8.61 sa_migrate_create_fks システムプロシージャ

dbo.migrate_remote_fks_list テーブルにリストされている各テーブルの外部キーを作成します。

構文

```
sa_migrate_create_fks( i_table_owner )
```

パラメータ

i_table_owner

この VARCHAR(128) パラメータを使用して、SQL Anywhere ターゲットデータベースで、マイグレート後の外部キーを所有するユーザを指定します。別のユーザに属しているテーブルをマイグレートする場合は、マイグレートするそれぞれのユーザに対してこのプロシージャを実行します。`i_table_owner` は、GRANT CONNECT 文を使用して作成されます。このパラメータの値は必須です。

備考

このプロシージャは、dbo.migrate_remote_fks_list テーブルにリストされている各テーブルの外部キーを作成します。`i_table_owner` 引数で指定したユーザが、ターゲットデータベース内の外部キーを所有します。

SQL Anywhere ターゲットデータベース内のテーブルの所有者がすべて同じでない場合は、外部キーをマイグレートする必要のあるテーブルを所有しているユーザごとにこのプロシージャを実行する必要があります。

注記

このシステムプロシージャは、他のいくつかのマイグレーションシステムプロシージャと組み合わせて使用されます。そのとき、以下の順序で実行する必要があります。

1. sa_migrate_create_remote_table_list
2. sa_migrate_create_tables
3. sa_migrate_data
4. sa_migrate_create_remote_fks_list
5. sa_migrate_create_fks
6. sa_migrate_drop_proxy_tables

または、sa_migrate システムプロシージャを使用すると、1 回ですべてのテーブルをマイグレートできます。

権限

ALTER ANY TABLE および CREATE ANY INDEX システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

最初の文は、dbo.migrate_remote_table_list に一覧されているテーブルの外部キーのリストを作成します。2 番目の文は、dbo.migrate_remote_fks_list テーブルに基づいて外部キーを作成します。SQL Anywhere ローカルデータベースでは、外部キーはユーザ LocalUser が所有するテーブルに属します。

```
CALL sa_migrate_create_remote_fks_list( 'RemoteSA' );
CALL sa_migrate_create_fks( 'LocalUser' );
```

関連情報

GRANT 文 [1131 ページ]
sa_migrate システムプロシージャ [1560 ページ]
sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
sa_migrate_create_tables システムプロシージャ [1568 ページ]
sa_migrate_data システムプロシージャ [1569 ページ]
sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]

1.6.8.62 sa_migrate_create_remote_fks_list システムプロシージャ

dbo.migrate_remote_fks_list テーブルに移植します。

構文

```
sa_migrate_create_remote_fks_list( server_name )
```

パラメータ

server_name

この VARCHAR(128) パラメータを使用して、リモートデータベースへの接続に使用されるリモートサーバの名前を指定します。リモートサーバは、CREATE SERVER 文で作成します。このパラメータの値は必須です。

備考

このプロシージャは、リモートデータベースからマイグレートできる外部キーのリストを dbo.migrate_remote_fks_list テーブルに移植します。このテーブルから、マイグレートしない外部キーのローを削除できます。

または、sa_migrate システムプロシージャを使用すると、1 回ですべてのテーブルをマイグレートできます。

このシステムプロシージャは、他のいくつかのマイグレーションシステムプロシージャと組み合わせて使用されます。sa_migrate_create_fks システムプロシージャの備考部分には、マイグレーションプロシージャのリストと、実行順序が記載されています。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、dbo.migrate_remote_table_list に一覧されているテーブルの外部キーのリストを作成します。

```
CALL sa_migrate_create_remote_fks_list( 'RemoteSA' );
```

関連情報

CREATE SERVER 文 [914 ページ]
sa_migrate システムプロシージャ [1560 ページ]
sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
sa_migrate_create_tables システムプロシージャ [1568 ページ]
sa_migrate_data システムプロシージャ [1569 ページ]
sa_migrate_create_fks システムプロシージャ [1563 ページ]
sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]

1.6.8.63 sa_migrate_create_remote_table_list システムプロシージャ

dbo.migrate_remote_table_list テーブルに移植します。

構文

```
sa_migrate_create_remote_table_list(
i_server_name
[, i_table_name
[, i_owner_name
[, i_database_name ] ] ]
)
```

パラメータ

i_server_name

この VARCHAR(128) パラメータを使用して、リモートデータベースへの接続に使用されるリモートサーバの名前を指定します。リモートサーバは、CREATE SERVER 文で作成します。このパラメータの値は必須です。

i_table_name

任意の VARCHAR(128) パラメータを使用して、マイグレートするテーブルの名前を指定します。すべてのテーブルをマイグレートする場合は NULL を指定します。デフォルト値は NULL です。`i_table_name` と `i_owner_name` の両方のパラメータに NULL を指定しないでください。

i_owner_name

任意の VARCHAR(128) パラメータを使用して、リモートデータベースでマイグレートするテーブルを所有するユーザを指定します。すべてのテーブルをマイグレートする場合は NULL を指定します。デフォルトは NULL です。`i_table_name` と `i_owner_name` の両方のパラメータに NULL を指定しないでください。

i_database_name

任意の VARCHAR(128) パラメータを使用して、テーブルのマイグレート元のリモートデータベースの名前を指定します。デフォルトは NULL です。Adaptive Server Enterprise と Microsoft SQL Server データベースからテーブルをマイグレートする場合は、データベース名を指定する必要があります。

備考

このプロシージャは、リモートデータベースからマイグレートできるテーブルのリストを dbo.migrate_remote_table_list テーブルに移植します。このテーブルから、マイグレートしないリモートテーブルのローを削除できます。

SQL Anywhere ターゲットデータベースで、マイグレート後のテーブルをすべて同じ所有者に属さないようにする場合は、マイグレートするテーブルを所有するユーザごとにこのプロシージャを実行します。

または、sa_migrate システムプロシージャを使用すると、1 回ですべてのテーブルをマイグレートできます。

警告

`i_table_name` と `i_owner_name` の両方のパラメータに NULL を指定しないでください。`i_table_name` パラメータと `i_owner_name` パラメータの両方に NULL を指定すると、データベース内のシステムテーブルを含む、すべてのテーブルがマイグレートします。また、リモートデータベースで同じ名前で所有者が異なるテーブルは、ターゲットデータベースでは 1 人の所有者に属します。一度に移行するテーブルは、1 人の所有者に関連付けられたテーブルだけにしてください。

このシステムプロシージャは、他のいくつかのマイグレーションシステムプロシージャと組み合わせて使用されます。sa_migrate_create_fks システムプロシージャの備考部分には、マイグレーションプロシージャのリストと、実行順序が記載されています。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、リモートデータベース上のユーザ DBA に属するテーブルのリストを作成します。

```
CALL sa_migrate_create_remote_table_list( 'RemoteSA', NULL, 'DBA', NULL );
```

関連情報

CREATE SERVER 文 [914 ページ]
sa_migrate システムプロシージャ [1560 ページ]
sa_migrate_create_tables システムプロシージャ [1568 ページ]
sa_migrate_data システムプロシージャ [1569 ページ]
sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
sa_migrate_create_fks システムプロシージャ [1563 ページ]
sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]

1.6.8.64 sa_migrate_create_tables システムプロシージャ

dbo.migrate_remote_table_list テーブルにリストされている各テーブルのプロキシテーブルとベーステーブルを作成します。

構文

```
sa_migrate_create_tables( i_table_owner )
```

パラメータ

i_table_owner

この VARCHAR(128) パラメータを使用して、SQL Anywhere ターゲットデータベースで、マイグレート後のテーブルを所有するユーザを指定します。このユーザは、GRANT CONNECT 文を使用して作成します。このパラメータの値は必須です。

備考

このプロシージャは、dbo.migrate_remote_table_list テーブル (sa_migrate_create_remote_table_list プロシージャを使用して作成したテーブル) にリストされている各テーブルに対して、ベーステーブルとプロキシテーブルを作成します。これらのプロキシテーブルとベーステーブルは、`i_table_owner` 引数で指定したユーザが所有します。このプロシージャは、リモートデータベースのリモートテーブルと同じプライマリキーインデックスとその他のインデックスも、新しいテーブルに対して作成します。

SQL Anywhere ターゲットデータベースで、マイグレート後のすべてのテーブルの所有者を同じにしたくない場合は、マイグレートするテーブルを所有するユーザごとに、sa_migrate_create_remote_table_list プロシージャと sa_migrate_create_tables プロシージャを実行します。

または、sa_migrate システムプロシージャを使用すると、1 回ですべてのテーブルをマイグレートできます。

このシステムプロシージャは、他のいくつかのマイグレーションシステムプロシージャと組み合わせて使用されます。sa_migrate_create_fks システムプロシージャの備考部分には、マイグレーションプロシージャのリストと、実行順序が記載されています。

権限

CREATE ANY TABLE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

最初の文は、リモートデータベース上のユーザ DBA に属するテーブルのリストを作成します。2 番目の文は、そのリストを使用する SQL Anywhere ターゲットデータベース上にベーステーブルとプロキシテーブルを作成します。これらのテーブルは、ユーザ LocalUser に属します。

```
CALL sa_migrate_create_remote_table_list( 'RemoteSA', NULL, 'DBA', NULL );
CALL sa_migrate_create_tables( 'LocalUser' );
```

関連情報

sa_migrate システムプロシージャ [1560 ページ]
sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
sa_migrate_data システムプロシージャ [1569 ページ]
sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
sa_migrate_create_fks システムプロシージャ [1563 ページ]
sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]

1.6.8.65 sa_migrate_data システムプロシージャ

リモートデータベーステーブルから SQL Anywhere ターゲットデータベースにデータをマイグレートします。

構文

```
sa_migrate_data( i_table_owner )
```

パラメータ

i_table_owner

この VARCHAR(128) パラメータを使用して、SQL Anywhere ターゲットデータベースで、マイグレート後のテーブルを所有するユーザを指定します。このユーザは、GRANT CONNECT 文を使用して作成します。このパラメータの値は必須です。

備考

このプロシージャは、リモートデータベースから SQL Anywhere データベースに、`i_table_owner` 引数で指定したユーザに属するテーブルのデータをマイグレートします。

SQL Anywhere ターゲットデータベース上のテーブルの所有者がすべて同じでない場合は、データをマイグレートするテーブルを所有しているユーザごとに、このプロシージャを実行する必要があります。

または、sa_migrate システムプロシージャを使用すると、1 回ですべてのテーブルをマイグレートできます。

このシステムプロシージャは、他のいくつかのマイグレーションシステムプロシージャと組み合わせて使用されます。sa_migrate_create_fks システムプロシージャの備考部分には、マイグレーションプロシージャのリストと、実行順序が記載されています。

権限

SELECT ANY TABLE および INSERT ANY TABLE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、ユーザ LocalUser に属するテーブルのデータを SQL Anywhere ターゲットデータベースに移行します。

```
CALL sa_migrate_data( 'LocalUser' );
```

関連情報

sa_migrate システムプロシージャ [1560 ページ]

sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
sa_migrate_create_tables システムプロシージャ [1568 ページ]
sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
sa_migrate_create_fks システムプロシージャ [1563 ページ]
sa_migrate_drop_proxy_tables システムプロシージャ [1571 ページ]

1.6.8.66 sa_migrate_drop_proxy_tables システムプロシージャ

マイグレーションの目的で作成されたプロキシテーブルを削除します。

構文

```
sa_migrate_drop_proxy_tables( i_table_owner )
```

パラメータ

i_table_owner

この VARCHAR(128) パラメータを使用して、SQL Anywhere ターゲットデータベースで、プロキシテーブルを所有するユーザを指定します。このユーザは、GRANT CONNECT 文を使用して作成します。このパラメータの値は必須です。

備考

このプロシージャは、マイグレーションのために作成されたプロキシテーブルを削除します。これらのプロキシテーブルを所有するユーザを、`i_table_owner` 引数で指定します。

SQL Anywhere ターゲットデータベースで、マイグレート後のテーブルをすべて同じユーザが所有しない場合、ユーザごとにこのプロシージャを呼び出して、すべてのプロキシテーブルを削除します。

または、sa_migrate システムプロシージャを使用すると、1 回ですべてのテーブルをマイグレートできます。

このシステムプロシージャは、他のいくつかのマイグレーションシステムプロシージャと組み合わせて使用されます。sa_migrate_create_fks システムプロシージャの備考部分には、マイグレーションプロシージャのリストと、実行順序が記載されています。

権限

DROP ANY TABLE オブジェクトレベル権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、SQL Anywhere ターゲットデータベースで、ユーザ LocalUser に属するプロキシテーブルを削除します。

```
CALL sa_migrate_drop_proxy_tables( 'LocalUser' );
```

関連情報

sa_migrate システムプロシージャ [1560 ページ]
sa_migrate_create_remote_table_list システムプロシージャ [1566 ページ]
sa_migrate_create_tables システムプロシージャ [1568 ページ]
sa_migrate_data システムプロシージャ [1569 ページ]
sa_migrate_create_remote_fks_list システムプロシージャ [1565 ページ]
sa_migrate_create_fks システムプロシージャ [1563 ページ]

1.6.8.67 sa_mirror_server_status システムプロシージャ

現在のサーバと現在のサーバから直接または間接的にログページを受信するすべてのサーバの接続ステータスを返します。プライマリサーバでは、接続されているすべてのサーバのステータスを返します。

構文

sa_mirror_server_status( )

結果セット

カラム名	データ型	説明
server_name	CHAR(128)	サーバの名前。

カラム名	データ型	説明
state	CHAR(20)	サーバの接続ステータス。値は次のいずれかです。 • connected • disconnected 場合によっては、接続されていないサーバに接続ステータスがある場合もあります。このようなことが発生する場合は、last_updated 時刻が過去になっています。
last_updated	TIMESTAMP WITH TIME ZONE	サーバステータスが最後に更新された時刻。
load_current	DOUBLE	データベースサーバで現在実行されている作業量。
load_last_1_min	DOUBLE	データベースサーバで過去 1 分間で実行された作業量。
load_last_5_mins	DOUBLE	データベースサーバで過去 5 分間で実行された作業量。
load_last_10_mins	DOUBLE	データベースサーバで過去 10 分間で実行された作業量。
num_connections	UNSIGNED INTEGER	データベースサーバとの接続の数。
num_processors	UNSIGNED INTEGER	データベースサーバプロセッサの数。
log_written	UNSIGNED BIGINT	サーバから受信した最後の更新に基づいてディスクに書き込まれた最新のトランザクションログの位置。
log_applied	UNSIGNED BIGINT	サーバから受信した最後の更新に基づいて適用されたトランザクションログ内の最後の操作。この値は、CurrentRedoPos プロパティの値と同じです。

備考

各サーバでは、サーバ自身とそのコピーノードのステータスが 5 秒ごとに更新されます。ミラーサーバの場合、ミラーサーバからログページを受信するすべてのコピーノードのステータスが返りますが、プライマリサーバのステータスは返りません。プレフィクス **load** が付いたカラムは、データベースサーバ上の負荷の計算値を表します。返される値は、データベースサーバの負荷を表し、他のプロセスからの負荷ではありません。負荷値が高い場合、データベースサーバで実行される作業量が多いことを示します。

last_updated カラムの時刻は、server_name カラムのサーバの時刻です。状態は、last_updated カラムで返される時刻に対して正確です。

NodeType 接続パラメータを指定すると、データベースサーバでは負荷情報に基づいて接続をリダイレクトします。データベースサーバは負荷が最も低いミラーサーバを選択します。すべてのサーバの負荷が同じ場合は、接続数が最も少ないサーバが使用されます。

コピーノードが親ノードとほとんど同じ時間に停止された場合、数分の間はコピーノードがまだ接続されているとプロシージャがレポートすることがあります。ただし、last_updated カラムはそのままで、更新されたステータスメッセージがコピーノードにレポートされておらず、おそらく切断されていることを示します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、ミラーリングが有効な場合に、現在のサーバと現在のサーバから直接または間接的にログページを受信するすべてのサーバの接続ステータスを返します。

```
CALL sa_mirror_server_status( );
```

関連情報

CREATE MIRROR SERVER 文 [858 ページ]
ALTER MIRROR SERVER 文 [662 ページ]
DROP MIRROR SERVER 文 [1047 ページ]

1.6.8.68 sa_nchar_terms システムプロシージャ

NCHAR 文字列を単語に分割し、各単語とその位置を 1 つのローとして返します。

構文

```
sa_nchar_terms(
text
[, config_name
[, owner ] ]
)
```

パラメータ

text

解析する LONG NVARCHAR 文字列。

config_name

このオプションの CHAR(128) パラメータを使用して、文字列の処理時に適用するテキスト設定オブジェクトを指定します。デフォルトは 'default_nchar' です。

owner

オプションの CHAR(128) パラメータを使用して、テキスト設定オブジェクトの所有者を指定します。デフォルト値は NULL です。所有者が指定されない場合、または NULL が指定されている場合には、現在のユーザが使用されます。

備考

このシステムプロシージャを使用すると、テキスト設定オブジェクトの設定が適用されたときに文字列がどのように解釈されるかを確認できます。インデックス処理で、またはクエリ文字列から、どの単語が削除されるかを確かめる場合に便利です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、デフォルトの NCHAR テキスト設定オブジェクト default_nchar を使用して、NCHAR 文字列 "It's a work-at-home day!" 内の単語を返します。

```
CALL sa_nchar_terms (N'It''s a work-at-home day!', 'default_nchar', 'sys');
```

term	position
It	1
s	2
a	3
work	4
at	5

term	position
home	6
day	7

関連情報

sa_char_terms システムプロシージャ [1452 ページ]

1.6.8.69 sa_performance_diagnostics システムプロシージャ

データベースサーバが要求のタイミングの記録を有効にしたとき、すべての接続について要求のタイミング情報の概要を返します。

構文

```
sa_performance_diagnostics( )
```

結果セット

カラム名	データ型	説明
Number	INTEGER	現在の接続の接続 ID (数値) を返します。

カラム名	データ型	説明
Name	VARCHAR(255)	現在の接続の名前を返します。 ConnectionName (CON) 接続パラメータを使用して接続名を指定できます。 データベースサーバによって作成されるテンポラリ接続には次の名前が使用されます。 • INT:ApplyRecovery • INT:BackupDB • INT:Checkpoint • INT:Cleaner • INT:CloseDB • INT:CreateDB • INT:CreateMirror • INT:DelayedCommit • INT:DiagRcvr • INT:DropDB • INT:EncryptDB • INT:Exchange • INT:FlushMirrorLog • INT:FlushStats • INT:HTTPReq • INT:PromoteMirror • INT:PurgeSnapshot • INT:ReconnectMirror • INT:RecoverMirror • INT:RedoCheckpoint • INT:RefreshIndex • INT:ReloadTrigger • INT:RenameMirror • INT:RestoreDB • INT:StartDB • INT:VSS
Userid	VARCHAR(255)	接続のユーザ ID を返します。
DBNumber	INTEGER	データベースの ID 番号を返します。
LoginTime	TIMESTAMP	接続が確立された日付と時刻を返します。この値はシミュレートされたタイムゾーンの影響を受けます。
TransactionStartTime	TIMESTAMP	COMMIT または ROLLBACK の後にデータベースが最初に変更された日時を返します。最後の COMMIT または ROLLBACK 以降にデータベースが変更されていない場合は、空の文字列を返します。この値はシミュレートされたタイムゾーンの影響を受けます。
LastReqTime	TIMESTAMP	指定された接続において最後の要求が開始された日時を返します。このカラムには、イベントなど、内部接続用の NULL が含まれています。この値はシミュレートされたタイムゾーンの影響を受けます。

カラム名	データ型	説明
ReqType	VARCHAR(255)	最後の要求のタイプを返します。接続が接続プールによってキャッシュされる場合、その ReqType 値は CONNECT_POOL_CACHE になります。
ReqStatus	VARCHAR(255)	要求のステータスを返します。値は次のいずれかです。 **Idle** この接続では現在、要求を処理していません。 **Unscheduled*** この接続では他の処理が実行されており、使用可能なデータベースサーバワーカを待っています。 **BlockedIO*** この接続はブロックされ、I/O 処理の完了を待っています。 **BlockedContention*** この接続はブロックされ、共有データベースサーバデータ構造体へのアクセスを待っています。 **BlockedLock** この接続はブロックされ、オブジェクトのロックの解放を待っています。 **Executing** この接続では現在、要求を実行しています。 アスタリスク (*) が付いている値が返されるのは、-zt サーバオプションによってデータベースサーバで要求タイミング情報のロギングがオンになっている場合に限られます。要求タイミング情報のロギングが実行されていない場合は (デフォルト)、値は Executing として返されます。
ReqTimeUnscheduled	DOUBLE	未スケジュール時間を返します。-zt オプションが指定されていない場合は、NULL を返します。
ReqTimeActive	DOUBLE	要求の処理にかかった時間 (秒単位)、または、-zt オプションが指定されていなかった場合は NULL を返します。
ReqTimeBlockIO	DOUBLE	I/O 処理の完了を待っていた時間 (秒単位)、または、-zt オプションが指定されていなかった場合は NULL を返します。
ReqTimeBlockLock	DOUBLE	ロックを待っていた時間 (秒単位)、または、-zt オプションが指定されていなかった場合は NULL を返します。
ReqTimeBlockContention	DOUBLE	アトミックアクセスを待っていた時間 (秒単位)、または、RequestTiming サーバプロパティが Off に設定されている場合は NULL を返します。
ReqCountUnscheduled	INTEGER	接続がスケジューリングを待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。
ReqCountActive	INTEGER	処理が完了した要求の数を返します。RequestTiming サーバプロパティが Off に設定されている場合は、NULL を返します。

カラム名	データ型	説明
ReqCountBlockIO	INTEGER	接続が I/O 処理の完了を待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。
ReqCountBlockLock	INTEGER	接続がロックの解放を待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。
ReqCountBlockContention	INTEGER	接続がアトミックアクセスを待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。
LastIdle	INTEGER	要求間のチック数を返します。
BlockedOn	INTEGER	現在の接続がブロックされていない場合は 0 を返し、ブロックされている場合はロック競合によってブロックされた接続の接続番号を返します。
UncommitOp	INTEGER	コミットされていないオペレーションの数を返します。
CurrentProcedure	VARCHAR(255)	接続で現在実行されているプロシージャの名前を返します。接続でネストされたプロシージャコールが実行されている場合は、現在のプロシージャの名前を返します。実行されているプロシージャがない場合は、空の文字列を返します。
EventName	VARCHAR(255)	接続でイベントハンドラが実行されている場合、関連付けられているイベントの名前を返します。それ以外の場合は、空の文字列を返します。
CurrentLineNumber	INTEGER	接続で実行されているプロシージャまたは複合文の現在の行番号を返します。実行されているプロシージャの名前は CurrentProcedure プロパティで取得できます。現在の行がクライアント側からの複合文の一部である場合は、空の文字列を返します。
LastStatement	LONG VARCHAR	現在の接続で最後に準備された SQL 文を返します。 LastStatement の値は、文が準備されると同時に設定され、文が削除されると同時にクリアされます。各接続につき 1 つの文の文字列のみが記憶されます。 ある接続について sa_conn_activity が空でない値を返した場合、その接続で現在実行されている文である可能性が高くなります。その文が完了している場合は、文がすでに削除され、このプロパティの値がクリアされている可能性があります。アプリケーションが複数の文を準備し、それらの文のステートメントハンドルを保持している場合、LastStatement が返す値は接続で現在実行されている処理を表しません。 クライアントでの文のキャッシュが有効であり、キャッシュされた文が再使用されているとき、このプロパティは空の文字列を返します。
LastPlanText	LONG VARCHAR	接続で最後に実行されたクエリの長いテキストプランを返します。最後のプランを記憶するかどうかは、sa_server_option システムプロシージャの RememberLastPlan オプションを設定するか、-zp サーバオプションを使用することで制御します。

カラム名	データ型	説明
AppInfo	LONG VARCHAR	接続を確立したクライアントに関する情報を返します。HTTP 接続では、ブラウザの情報が含まれています。jConnect または SAP Open Client の古いバージョンを使った接続については、情報は不完全の場合があります。 API 値は、DBLIB、ODBC、OLEDB、ADO.NET、iAnywhereJDBC、CAPI_JavaScript-XS、CAPI_Node.js、CAPI_PerlDBD、CAPI_PHP、CAPI_PYTHON、CAPI_RUBY、DBEXPRESS、またはその他のユーザ定義値のいずれかです。
LockCount	INTEGER	接続で保持されているロックの数を返します。
SnapshotCount	INTEGER	接続に関連付けられているスナップショットの数を返します。

備考

sa_performance_diagnostics システムプロシージャは、要求のタイミングプロパティで構成される結果セットと統計情報 (統計情報を収集するようにサーバに指示されている場合) を返します。要求のタイミング情報の記録は、sa_performance_diagnostics の呼び出し前にデータベースサーバでオンにする必要があります。オンにするには、データベースサーバの起動時に -zt オプションを指定するか、次の文を実行します。

```
CALL sa_server_option( 'RequestTiming','ON' );
```

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

現在実行中で、60 秒よりも長く実行されているすべての要求を検索します。

```
SELECT  Number, Name,
      CAST( DATEDIFF( second, LastReqTime, CURRENT TIMESTAMP ) AS DOUBLE ) AS
ReqTime
FROM  sa_performance_diagnostics()
WHERE ReqStatus <> 'IDLE' AND ReqTime > 60.0
ORDER BY ReqTime DESC;
```

関連情報

sa_performance_statistics システムプロシージャ [1581 ページ]
sa_server_option システムプロシージャ [1606 ページ]

1.6.8.70 sa_performance_statistics システムプロシージャ

サーバ、データベース、および接続のパフォーマンス統計を返します。

構文

sa_performance_statistics( )

結果セット

カラム名	データ型	説明
DBNumber	INTEGER	データベースの ID 番号を返します。 プロパティ型が Server の場合は NULL が返されます。
ConnNumber	INTEGER	現在の接続の接続 ID (数値) を返します。 Type が Server または Database の場合は NULL が返されます。
PropNum	INTEGER	プロパティ番号を返します。
PropName	VARCHAR(255)	プロパティ名を返します。
Value	BIGINT	プロパティの値が返されます。

備考

sa_performance_statistics システムプロパティは、一連のパフォーマンス統計で構成される結果セットを返します。この結果は、PROPERTY 関数、DB_PROPERTY 関数、CONNECTION_PROPERTY 関数を使用して返すことができる結果のサブセットです。

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例は、すべてのパフォーマンス統計情報を `dump_stats.txt` というテキストファイルにアンロードします。

```
UNLOAD
   SELECT CURRENT TIMESTAMP, *
   FROM sa_performance_statistics()
   TO 'dump_stats.txt'
   APPEND ON;
```

関連情報

sa_performance_diagnostics システムプロシージャ [1576 ページ]
sa_server_option システムプロシージャ [1606 ページ]

1.6.8.71 sa_post_login_procedure システムプロシージャ

ユーザのパスワードの有効期限切れが近づいているかどうかを判断します。

構文

sa_post_login_procedure( )

パラメータ

なし

結果セット

カラム名	データ型	説明
message_text	VARCHAR(255)	message_action が 1 の場合、message_text は表示するメッセージを返します。message_action が 0 の場合、message_text は NULL です。
message_action	INTEGER	パスワードの有効期限切れが近づいているかどうか (1= はい、0= いいえ)。

備考

sa_post_login_procedure システムプロシージャは、post_login_procedure データベースオプションのデフォルト設定です。

sa_post_login_procedure は、ユーザの password_life_time と password_grace_time の各ログインポリシーオプション値および現在の日付と時刻を使用して、ユーザのパスワードの有効期限切れが近づいているかどうかを判断します。有効期限切れに近い場合は、ユーザに表示するメッセージが結果セットに返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、sa_post_login_procedure を使用して、現在のユーザのパスワードの期限が迫っているかどうかを判断します。

```
CALL sa_post_login_procedure();
```

1.6.8.72 sa_procedure_profile システムプロシージャ

データベースで実行されたプロシージャ、関数、イベント、またはトリガ内の各行について、実行時間に関する情報をレポートします。

構文

```
sa_procedure_profile(
   [ filename
   [, save_to_file ] ]
)
```

パラメータ

filename

任意の LONG VARCHAR パラメータを使用して、プロファイル情報を保存するファイル、またはロードするファイルを指定します。デフォルトは NULL です。プロファイル情報の保存とロードの詳細については、以下の備考部分を参照してください。

save_to_file

任意の INTEGER パラメータを使用して、プロファイル情報をファイルに保存するか、前に保存したファイルからロードするかを指定します。デフォルトは 0 です。

結果セット

カラム名	データ型	説明
object_type	CHAR(1)	オブジェクトのタイプ。使用できるオブジェクトタイプのリストについては、以下の備考部分を参照してください。
object_name	CHAR(128)	ストアドプロシージャ、ファンクション、イベント、またはトリガの名前。object_type が C または D の場合、これはシステムトリガが定義された外部キーの名前です。
owner_name	CHAR(128)	オブジェクトの所有者。
table_name	CHAR(128)	トリガに対応するテーブル (他のオブジェクトタイプの場合、値は NULL)。
line_num	UNSIGNED INTEGER	プロシージャ内の行番号。
executions	UNSIGNED INTEGER	行の実行回数。
millisecs	UNSIGNED INTEGER	行の実行時間 (ミリ秒単位)。

カラム名	データ型	説明
percentage	DOUBLE	全体的な実行時間に対する特定の行の実行時間の割合。
foreign_owner	CHAR(128)	システムトリガの外部テーブルを所有するデータベースユーザ。
foreign_table	CHAR(128)	システムトリガの外部テーブルの名前。

備考

このプロシージャは、次の用途で使用できます。

詳細なプロシージャのプロファイル情報を返す

この場合、引数指定することなく、プロシージャを呼び出すだけです。

詳細なプロシージャのプロファイル情報をファイルに保存

この場合、`filename` 引数を含め、`save_to_file` 引数に 1 を指定します。

詳細なプロシージャのプロファイル情報を前に保存したファイルからロード

この場合、`filename` 引数を含め、`save_to_file` 引数に 0 を指定します。この方法でプロシージャを使用できるのは、ロードするファイルが、プロシージャを実行するデータベースと同じデータベースで作成されたものである場合です。それ以外の場合、結果は使用できない可能性があります。

結果セットには、プロシージャ、トリガ、関数、イベント内の個々の行に関する実行回数の情報と、行が使用するプロシージャの合計の実行回数の割合が含まれるため、このプロファイル情報を使用して、パフォーマンスを低下させる可能性がある遅いプロシージャを微調整できます。

プロファイリングを有効にしてから、データベースのプロファイルを作成します。

結果セットの object_type カラムは、次のようになります。

P

ストアドプロシージャ

F

ファンクション

E

イベント

T

トリガ

C

ON UPDATE システムトリガ

D

ON DELETE システムトリガ

各実行について、行ごとの詳細ではなく、概要情報が必要な場合、sa_procedure_profile_summary プロシージャを使用します。

権限

MONITOR または MANAGE PROFILING システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

次の権限も必要です。

- SELECT ANY TABLE (`filename` が NULL でなく、`save_to_file` が 1 の場合)
- LOAD ANY TABLE (`filename` が NULL でなく、`save_to_file` が 0 の場合)

関連する動作

[なし]

例

次の文は、データベースで実行されたすべてのプロシージャ、関数、イベント、またはトリガの各行について実行時間を返します。

```
CALL sa_procedure_profile( );
```

次の文は、上記の例と同じ詳細なプロシージャのプロファイル情報を返します。また、`detailedinfo.txt` というファイルに保存します。

```
CALL sa_procedure_profile( 'detailedinfo.txt', 1 );
```

次の文はいずれも、`detailedinfo.txt` というファイルから詳細なプロシージャのプロファイル情報をロードするときに使用できます。

```
CALL sa_procedure_profile( 'detailedinfo.txt', 0 );
```

```
CALL sa_procedure_profile( 'detailedinfo.txt' );
```

関連情報

sa_server_option システムプロシージャ [1606 ページ]
sa_procedure_profile_summary システムプロシージャ [1587 ページ]

1.6.8.73 sa_procedure_profile_summary システムプロシージャ

データベースで実行されたすべてのプロシージャ、ファンクション、イベント、またはトリガの実行時間に関する概要情報をレポートします。

構文

```
sa_procedure_profile_summary(
   [ filename
   [, save_to_file ] ]
)
```

パラメータ

filename

任意の LONG VARCHAR パラメータを使用して、プロファイル情報を保存するファイル、またはロードするファイルを指定します。デフォルトは NULL です。プロファイル情報の保存とロードの詳細については、以下の備考部分を参照してください。

save_to_file

任意の INTEGER パラメータを使用して、概要情報をファイルに保存するか、前に保存したファイルからロードするかを指定します。デフォルトは 0 です。

結果セット

カラム名	データ型	説明
object_type	CHAR(1)	オブジェクトのタイプ。使用できるオブジェクトタイプのリストについては、以下の備考部分を参照してください。
object_name	CHAR(128)	ストアドプロシージャ、ファンクション、イベント、またはトリガの名前。
owner_name	CHAR(128)	オブジェクトの所有者。
table_name	CHAR(128)	トリガに対応するテーブル (他のオブジェクトタイプの場合、値は NULL)。
executions	UNSIGNED INTEGER	各プロシージャの実行回数。
millisecs	UNSIGNED INTEGER	プロシージャの実行時間 (ミリ秒単位)。
foreign_owner	CHAR(128)	システムトリガの外部テーブルを所有するデータベースユーザ。
foreign_table	CHAR(128)	システムトリガの外部テーブルの名前。

備考

このプロシージャは、次の用途で使用できます。

現在の概要情報を返します。

この場合、引数指定することなく、プロシージャを呼び出すだけです。
現在の概要情報をファイルに保存

この場合、`filename` 引数を含め、`save_to_file` 引数に 1 を指定します。
保存された概要情報をファイルからロード

この場合、`filename` 引数を含め、`save_to_file` 引数に 0 を指定します。この方法でプロシージャを使用できるのは、ロードするファイルが、プロシージャを実行するデータベースと同じデータベースで作成されたものである場合です。それ以外の場合、結果は使用できない可能性があります。

このプロシージャは、ストアドプロシージャ、ファンクション、イベント、トリガの使用頻度や効率に関する情報を返すため、この情報を使用して、時間がかかるプロシージャを微調整してデータベースのパフォーマンスを改善できます。

プロファイリングを有効にしてから、データベースのプロファイルを作成します。

結果セットの object_type カラムは、次のようになります。

P

ストアドプロシージャ
F

ファンクション
E

イベント
T

トリガ
C

ON UPDATE システムトリガ
D

ON DELETE システムトリガ

各実行について、概要情報ではなく、行ごとの詳細が必要な場合、sa_procedure_profile プロシージャを使用します。

権限

MONITOR または MANAGE PROFILING システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

次の権限も必要です。

- SELECT ANY TABLE (`filename` が NULL でなく、`save_to_file` が 1 の場合)
- LOAD ANY TABLE (`filename` が NULL でなく、`save_to_file` が 0 の場合)

関連する動作

なし。

例

次の文は、データベースで実行されたすべてのプロシージャ、ファンクション、イベント、またはトリガについて実行時間を返します。

```
CALL sa_procedure_profile_summary( );
```

次の文は、前の例と同じ概要情報を返します。また、summaryinfo.txt というファイルに保存します。

```
CALL sa_procedure_profile_summary( 'summaryinfo.txt', 1 );
```

次の文はいずれも、summaryinfo.txt というファイルに保存されている概要情報をロードするときに使用できます。

```
CALL sa_procedure_profile_summary( 'summaryinfo.txt', 0 );
```

```
CALL sa_procedure_profile_summary( 'summaryinfo.txt' );
```

関連情報

sa_server_option システムプロシージャ [1606 ページ]
sa_procedure_profile システムプロシージャ [1584 ページ]

1.6.8.74 sa_recompile_views システムプロシージャ

カタログに格納された、カラム定義を持たないビュー定義を見つけ、カラム定義を作成します。

構文

```
sa_recompile_views( [ ignore_errors ] )
```

パラメータ

ignore_errors

任意の INTEGER パラメータを使用して、再コンパイル中に発生したエラーを返すかどうかを指定します。0 と指定すると、カラム定義が失敗したビューごとにエラーが返されます。1 または 0 以外の値を指定すると、エラーは返されません。デフォルトは 0 です。

備考

このプロシージャは、カラム定義を持たないビューをカタログで見つけ、RECOMPILE 句を指定した ALTER VIEW 文を実行してカラム定義を作成します。プロシージャは、コンパイルを必要とするものがなくなるまで、または残りのカラム定義を作成できなくなるまで、カラム定義を持たない各ビューに対してこの処理を実行します。プロシージャでビューを再コンパイルできない場合は、エラーがレポートされます。このプロシージャに 0 以外のパラメータを指定することによって、エラーを抑制できます。

警告

sa_recompile_views システムプロシージャは、`reload.sql` スクリプト内からのみ呼び出してください。このプロシージャはアンロードユーティリティ (dbunload) によって使用されます。明示的には使用しないでください。

sa_recompile_views システムプロシージャは、マテリアライズドビューまたは DISABLED と印が付いているビューを再コンパイルしません。

権限

ALTER ANY VIEW システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

VALID ステータスのない通常の各ビューについては、`ALTER VIEW owner.viewname ENABLE` 文が実行され、オートコミットが行われます。

例

`reload.sql` スクリプトからの次の例では、sa_recompile_views システムプロシージャを使用して、カラム定義を持たないカタログに保存されているビューの定義を検索して、カラム定義を作成します。エラーは無視されます。

```
CALL sa_recompile_views( 1 );
```

関連情報

ALTER VIEW 文 [734 ページ]

1.6.8.75 sa_refresh_materialized_views システムプロシージャ

初期化されていないステータスであるすべてのマテリアライズドビューを初期化します。

構文

```
sa_refresh_materialized_views( [ ignore_errors ] )
```

パラメータ

ignore_errors

任意の INTEGER パラメータを使用して、再コンパイル中に発生したエラーを返すかどうかを指定します。0 と指定すると、カラム定義が失敗したビューごとにエラーが返されます。1 または 0 以外の値を指定すると、エラーは返されません。デフォルトは 0 です。

備考

作成したばかりのため、再有効化したばかりのため、初期化や更新の最後の実行がエラーにより失敗したために、マテリアライズドビューが未初期化ステータスになることがあります。sa_refresh_materialized_views システムプロシージャは、このようなすべてのマテリアライズドビューについてデータベースをスキャンし、初期化を試みます。プロシージャでマテリアライズドビューの初期化時にエラーが発生した場合、残りの未初期化ビューの処理が続けられます。

また、REFRESH MATERIALIZED VIEW 文を使用して、マテリアライズドビューを初期化することもできます。

権限

ALTER ANY MATERIALIZED VIEW システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例では、sa_refresh_materialized_views システムプロシージャを使用して、初期化されていない状態にある実体化されたすべてのビューを初期化します。エラーは無視されます。

```
CALL sa_refresh_materialized_views( 1 );
```

関連情報

REFRESH MATERIALIZED VIEW 文 [1252 ページ]

1.6.8.76 sa_refresh_text_indexes システムプロシージャ

MANUAL REFRESH または AUTO REFRESH と定義されているすべてのテキストインデックスを再表示します。

構文

```
sa_refresh_text_indexes( )
```

備考

sa_refresh_text_indexes システムプロシージャは、MANUAL REFRESH または AUTO REFRESH と定義されているすべてのテキストインデックスを再表示します。IMMEDIATE REFRESH (デフォルト) と定義されているテキストインデックスは再表示されません。これは、このようなインデックスの変更は、基本となるテーブルでデータが変更されたときに行われるためです。

権限

CREATE ANY INDEX または ALTER ANY OBJECT システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

オートコミット

例

次の文は、データベースに含まれるすべての MANUAL REFRESH と AUTO REFRESH のテキストインデックスを再表示します。

```
CALL sa_refresh_text_indexes();
```

関連情報

DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE 文 [1366 ページ]
SYSTEXTIDX システムビュー [1850 ページ]
sa_text_index_stats システムプロシージャ [1641 ページ]
sa_text_index_vocab システムプロシージャ [1643 ページ]

1.6.8.77 sa_remove_tracing_data システムプロシージャ (廃止予定)

診断トレース機能は廃止予定です。SQL Anywhere プロファイラを使用して、データベースの問題を診断します。sa_remove_tracing_data システムプロシージャは、診断トレーシングテーブルから、指定したロギング (トレーシング) セッション ID に関連するすべてのレコードを永久的に削除します。

構文

```
sa_remove_tracing_data( log_session_id )
```

パラメータ

log_session_id

この UNSIGNED INTEGER パラメータを使用して、データを削除するロギングセッションの ID を指定します。

備考

指定した `log_session_id` にレコードがない場合、プロシージャでは何も行われません。このプロシージャには戻り値がありません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、DIAGNOSTICS システムロールと MANAGE PROFILING システム権限が必要です。

関連する動作

指定した log_session_id にレコードが見つからなかった場合でも、完了時にコミットが発生します。

例

この例では、ロギングセッション ID 1 に関連するすべてのレコードを診断トレーシングテーブルから完全に削除します。

```
CALL sa_remove_tracing_data( 1 );
```

関連情報

診断トレーシングテーブル (廃止予定) [1415 ページ]

1.6.8.78 sa_report_deadlocks システムプロシージャ

データベースサーバによって作成された内部バッファから、デッドロックに関する情報を取り出します。

構文

sa_report_deadlocks( )

結果セット

カラム名	データ型	説明
snapshotId	BIGINT	デッドロックインスタンス (特定のデッドロックに関するすべてのローが同じ ID を持ちます)。
snapshotAt	TIMESTAMP	デッドロックが発生したときのデータベースサーバのローカル日時。この値はシミュレートされたタイムゾーンの影響を受けます。
waiter	INTEGER	待機している接続の接続ハンドル。
who	VARCHAR(128)	待機している接続に関連付けられているユーザ ID。

カラム名	データ型	説明
what	LONG VARCHAR	待機している接続によって実行されているコマンド。 この情報は、データベースサーバのコマンドラインに -zl オプションを指定して最後に作成された SQL 文の取得をオンにするか、sa_server_option システムプロシージャを使用してこの機能をオンにした場合にのみ使用できます。
object_id	UNSIGNED BIGINT	ローを含むテーブルのオブジェクト ID。デッドロックにミューテックスが使用される場合、この値はデッドロックに使用されるミューテックスのオブジェクト ID です。
record_id	BIGINT	関連付けられているローのロー ID。
owner	INTEGER	待機しているロックを所有している接続の接続ハンドル。
is_victim	BIT	ロールバックされたトランザクションを識別。
rollback_operation_count	UNSIGNED INTEGER	トランザクションをロールバックした場合に失われる可能性のある、コミットされていないオペレーションの数。
table_id	UNSIGNED BIGINT	ロックされるオブジェクトに関連付けられたテーブル ID。

備考

log_deadlocks オプションが On に設定されている場合、データベースサーバは、デッドロックに関する情報を内部バッファに記録します。sa_report_deadlocks システムプロシージャを使用して、ログ内の情報を表示できます。

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次のクエリを実行して、デッドロックを識別できます。

```
CALL sa_report_deadlocks( );
```

関連情報

sa_server_option システムプロシージャ [1606 ページ]
sa_server_option システムプロシージャ [1606 ページ]

1.6.8.79 sa_reserved_words システムプロシージャ

予約語のリストを返します。SQL 文で使用するキーワードの多くは予約語です。

構文

```
sa_reserved_words( )
```

結果セット

カラム名	データ型	説明
reserved_word	CHAR(128)	予約語。

備考

このプロシージャはパラメータを持たず、1 つのローにつき 1 つの語を返します。予約語のリストは、データベースファイルを作成するために使用されるソフトウェアのバージョンではなく、クエリを実行するデータベースサーバのバージョンに基づきます。

リスト内のすべてのワードを予約語として有効化することはできません。たとえば、キーワード LIMIT は reserved_keywords オプションを使用することで予約語として有効化または無効化できます。

権限

なし。

関連する動作

そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、予約語のリストを返します。

```
SELECT * FROM sa_reserved_words();
```

関連情報

予約語 [6 ページ]

1.6.8.80 sa_reset_identity システムプロシージャ

次の identity の値をテーブルに設定できます。このシステムプロシージャを使用して、次に挿入されるローの AUTOINCREMENT 値を変更します。

構文

```
sa_reset_identity(
tbl_name
[, owner_name
[, new_identity ] ]
)
```

パラメータ

tbl_name

この CHAR(128) パラメータを使用して、識別値をリセットするテーブルを指定します。テーブルの所有者を指定しない場合、tbl_name はデータベースのテーブルをユニークに識別する必要があります。

owner_name

任意の CHAR(128) パラメータを使用して、識別値をリセットするテーブルの所有者を指定します。デフォルトは NULL です。owner_name を指定しない場合は、3 番目の引数に名前付きパラメータを使用します。次に例を示します。

```
CALL sa_reset_identity( 'Employees', new_identity=100 );
```

new_identity

任意の BIGINT パラメータを使用して、オートインクリメントの開始値を指定します。デフォルトは NULL です。

備考

テーブルに挿入されるローに対して生成される次の ID 値は、new_identity + 1 です。

new_identity + 1 がテーブルの既存のローと競合するかどうかを確認するチェックは発生しません。たとえば、new_identity を 100 と指定した場合、挿入した次のローは 101 という ID 値を取得します。ただし、テーブルに 101 がすでに存在する場合、ローの挿入は失敗します。

sa_reset_identity システムプロシージャは、AUTOINCREMENT または GLOBAL AUTOINCREMENT のデフォルトが指定されたカラムのないテーブルに対しては使用できません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、テーブルの所有者であるか、ALTER ANY TABLE システム権限を持っている必要があります。

関連する動作

値が更新された後にチェックポイントを発生させます。

例

次の文は次の ID 値を 101 にリセットします。

```
CALL sa_reset_identity( 'Employees', 'GROUPO', 100 );
```

1.6.8.81 sa_rowgenerator システムプロシージャ

指定された開始値と終了値の間のローを格納した結果セットを返します。

構文

```
sa_rowgenerator(
  [ rstart
  [, rend
  [, rstep ] ] ]
)
```

パラメータ

rstart

任意の INTEGER パラメータを使用して、開始値を指定します。デフォルト値は 0 です。

rend

任意の INTEGER パラメータを使用して、`rstart` 以上の終了値を指定します。デフォルト値は 100 です。

rstep

この INTEGER パラメータを使用して、シーケンス値の増分を指定します。デフォルト値は 1 です。

結果セット

カラム名	データ型	説明
row_num	INTEGER	シーケンス番号。

備考

sa_rowgenerator プロシージャをクエリの FROM 句で使用して、番号のシーケンスを生成できます。RowGenerator システムテーブルを使用する代わりに、このプロシージャを使用できます。sa_rowgenerator は次のようなタスクに使用できます。

- 結果セット内の既知の数のローについてテストデータを生成します。
- あらゆる範囲の値に対するローを格納した結果セットを生成します。たとえば、1 か月の毎日についてのローを生成したり、郵便番号の範囲を生成したりできます。
- 結果セット内に指定した数のローを格納するクエリを生成します。これは、クエリのパフォーマンスのテストに役立ちます。

正しい開始値、終了値、正の 0 以外の増分値を指定しないと、ローは返されません。

次の文を使用して、RowGenerator テーブルの動作をエミュレートできます。

```
SELECT row_num FROM sa_rowgenerator( 1, 255 );
```

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次のクエリは、現在の月の 1 日ごとに 1 つのローを含む結果セットを返します。

```
SELECT DATEADD( day, row_num-1,
        YMD( DATEPART( year, CURRENT DATE ),
            DATEPART( month, CURRENT DATE ), 1 ) )
    AS day_of_month
    FROM sa_rowgenerator( 1, 31, 1 )
    WHERE DATEPART( month, day_of_month ) = DATEPART( month, CURRENT DATE )
    ORDER BY row_num;
```

次のクエリは、郵便番号が (0-9999)、(10000-19999)、...、(90000-99999) の範囲の地域に居住している従業員数を示します。従業員のいない範囲もあり、その場合は警告が発生します。

sa_rowgenerator プロシージャを使用すると、ある範囲の郵便番号に従業員がいない場合でも、これらの範囲を生成できます。

```
SELECT row_num AS r1, row_num+9999 AS r2, COUNT( PostalCode ) AS zips_in_range
    FROM sa_rowgenerator( 0, 99999, 10000 ) D LEFT JOIN Employees
        ON PostalCode BETWEEN r1 AND r2
    GROUP BY r1, r2
    ORDER BY 1;
```

次の例は、データのローを 10 個生成し、それらを NewEmployees テーブルに挿入します。

```
INSERT INTO NewEmployees ( ID, Salary, Name )
    SELECT row_num, CAST( RAND() * 1000 AS INTEGER ), 'Mary'
    FROM sa_rowgenerator( 1, 10 );
```

次の例は、sa_rowgenerator システムプロシージャを使用して、すべての整数を含むビューを作成します。この例の値 2147483647 は、サポートされている最大の符号付き整数を表します。

```
CREATE VIEW Integers AS
    SELECT row_num AS n
    FROM sa_rowgenerator( 0, 2147483647, 1 );
```

次の例は、sa_rowgenerator システムプロシージャを使用して、0001-01-01 ～ 9999-12-31 の日付を含むビューを作成します。この例の値 3652058 は、0001-01-01 ～ 9999-12-31 (それぞれサポートされている最初の日と最後の日) の日数を表します。

```
CREATE VIEW Dates AS
    SELECT DATEADD( day, row_num, '0001-01-01' ) AS d
    FROM sa_rowgenerator( 0, 3652058, 1 );
```

次のクエリは、1900 年から 2058 年までの間の 54 週あるすべての年を返します。

```
SELECT DATEADD ( day, row_num, '1900-01-01' ) AS d, DATEPART ( week, d ) w
    FROM sa_rowgenerator ( 0, 63919, 1 )
    WHERE w = 54;
```

関連情報

RowGenerator テーブル (dbo) [1429 ページ]

1.6.8.82 sa_save_trace_data システムプロシージャ

トレーシングデータを基本テーブルに保存します。

構文

sa_save_trace_data( )

備考

トレースセッションが実行中に、診断データは診断トレーシングテーブルの一時的なバージョンに保存されます。トレースセッションを停止するときに、診断トレーシングの基本テーブルにトレーシングデータを永続的に保存するかどうかを指定します。データの保存を選択しない場合、sa_save_trace_data システムプロシージャを使用して、セッションを停止した後にデータを保存することもできます。

sa_save_trace_data システムプロシージャは、トレーシングが進行中の場合にエラーを返します。このシステムプロシージャを使用するには、トレーシングを停止する必要があります。

トレーシングの停止時にユーザが WITHOUT SAVING を指定した場合でも、sa_save_trace_data システムプロシージャを使用できます。また、プロシージャはトレーシングデータベースから呼び出す必要があります。

権限

MANAGE PROFILING システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

オートコミット。

例

この例では、診断トレーシングテーブルにトレーシングデータを保存します。

```
CALL sa_save_trace_data( );
```

関連情報

診断トレーシングテーブル (廃止予定) [1415 ページ]

1.6.8.83 sa_send_udp システム関数

指定されたアドレスに UDP パケットを送信します。

構文

```
sa_send_udp(
destAddress
, destPort
, msg
)
```

パラメータ

destAddress

この CHAR(254) パラメータを使用して、ホスト名または IP アドレスを指定します。

destPort

この UNSIGNED SMALLINT パラメータを使用して、使用するポート番号を指定します。

msg

この LONG BINARY パラメータを使用して、指定されたアドレスに送信するメッセージを指定します。この値が文字列の場合は、一重引用符で囲みます。

戻り値

この関数は INTEGER ステータスコードを返します。

備考

この関数は、指定されたアドレスに 1 つの UDP パケットを送信します。メッセージが正常に送信された場合は 0 を返し、エラーが発生した場合はエラーコードを返します。エラーコードは次のいずれかです。

- UDP ソケットで送信するにはメッセージが大きすぎる場合 (サイズはオペレーティングシステムによって決定)、または送信先アドレスに問題がある場合は -1
- オペレーティングシステムによって返される Winsock／Posix エラーコード

`msg` パラメータにバイナリデータが含まれる場合、または文字列よりも複雑な場合は、BINARY 変数を使用することもできます。次に例を示します。

```
CREATE VARIABLE v LONG BINARY;
SET v='This is a UDP message';
SELECT sa_send_udp( '10.25.99.124', 1234, v );
```

```
DROP VARIABLE v;
```

この関数を Mobile Link サーバで開始される同期とともに使用して、Mobile Link Listener ユーティリティ (`dblsn.exe`) を起動できます。Mobile Link Listener に通知する方法として sa_send_udp 関数を使用する場合は、UDP パケットに 1 を追加してください。この数字は、サーバで開始された同期プロトコル番号です。Mobile Link の将来のバージョンでは、新しいプロトコルバージョンによって Mobile Link Listener の動作が変更される可能性があります。

権限

MANAGE ANY WEB SERVICE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、メッセージ「This is a test」をポート 2345 の IP アドレス 10.25.99.196 に送信します。

```
SELECT sa_send_udp( '10.25.99.196', 2345, 'This is a test' );
```

1.6.8.84 sa_server_messages システムプロシージャ

データベースサーバメッセージウィンドウのメッセージを結果セットとして返すことができます。

構文

```
sa_server_messages(
[ first_msg
[, num_msgs ] ]
)
```

パラメータ

first_msg

`num_msgs` パラメータの符号に応じて、任意の UNSIGNED BIGINT パラメータを使用して、返される最初または最後のメッセージの ID を指定します。デフォルトは NULL です。つまり、`num_msgs` が NULL または正の場合、検索はリストの先頭から開始されます。`num_msgs` が負の場合、検索はリストの最後を過ぎたところから開始されます。

num_msgs

任意の BIGINT パラメータを使用して、返されるメッセージの数を指定します。符号は、first_msg から始まるメッセージを要求するか、または first_msg で終わるメッセージを要求するかを示します。デフォルトは NULL です。つまり、first_msg からリストの最後までのすべてのメッセージが返されます。

結果セット

カラム名	データ型	説明
msg_id	UNSIGNED BIGINT	ユニークなメッセージ ID。メッセージ ID は 0 から始まります。
msg_text	LONG VARCHAR	メッセージテキスト。
msg_time	TIMESTAMP	メッセージが発行されたときのデータベースサーバのローカル日時。この値はシミュレートされたタイムゾーンの影響を受けます。
msg_severity	VARCHAR(255)	メッセージの重大度レベル。このカラムには、次のいずれかの値が含まれます。 **INFO** 情報メッセージ。 **WARN** 警告。 **ERR** エラー。

カラム名	データ型	説明
msg_category	VARCHAR(255)	メッセージのカテゴリ。このカラムには、次のいずれかの値が含まれます。 **STARTUP/SHUTDOWN** データベースサーバまたはデータベースの開始やシャットダウンに関連するメッセージ。 **CHKPT** チェックポイントに関連するメッセージ。 **MSG** MESSAGE 文または PRINT 文を使用して生成されたメッセージ。 **DBA_MSG** ERVER OPERATOR システム権限を必要とする MESSAGE 文を使用して生成されたメッセージ。イベントログに送信されたメッセージなど。 **CONN** データベースサーバのコネクティビティに関するメッセージ。 **OTHER** その他のタイプのすべてのメッセージ。
msg_database	VARCHAR(255)	メッセージが特定の 1 つのデータベースに適用される場合は、そのメッセージに関連付けられたデータベース名。それ以外の場合は NULL。

備考

新しいメッセージがコンソールに送信される場合、メッセージの数が MessageCategoryLimit プロパティの値を超えていると、同じカテゴリまたは同じ重大度レベルの古いメッセージは削除されます。その結果、結果セットにギャップができ、連続した 2 つのローでメッセージ ID が不連続となる場合があります。

STARTUP/SHUTDOWN メッセージカテゴリでは、サーバのシャットダウンメッセージは表示されません。シャットダウンメッセージは、サーバで複数のデータベースが実行中で、そのうちの 1 つ以上がシャットダウンされた場合のみ表示されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次のコマンドは、ID が 3 のメッセージを先頭として、100 メッセージを要求します。

```
CALL sa_server_messages( 3, 100 );
```

次のコマンドは、メッセージ 4032 までの (このメッセージを含む) 500 メッセージを要求します。

```
CALL sa_server_messages( 4032, -500 );
```

次のコマンドは、メッセージ 3 以降のすべてのメッセージを要求します。

```
CALL sa_server_messages( 3, NULL );
```

```
CALL sa_server_messages( 3 );
```

次のコマンドは、リストの先頭からの 100 メッセージを要求します。

```
CALL sa_server_messages( NULL, 100 );
```

次のコマンドは、リストの最後の 100 メッセージを要求します。

```
CALL sa_server_messages( NULL, -100 );
```

次のコマンドは、リストのすべてのメッセージを要求します。

```
CALL sa_server_messages( NULL, NULL );
```

```
CALL sa_server_messages( );
```

1.6.8.85 sa_server_option システムプロシージャ

データベースサーバの実行中に、データベースサーバオプションを上書きします。

構文

```
sa_server_option(
opt
, val
)
```

パラメータ

opt

この CHAR(128) パラメータを使用して、データベースサーバオプション名を指定します。

val

この LONG VARCHAR パラメータを使用して、データベースサーバオプションの新しい値を指定します。

備考

このプロシージャを使用すると、データベースサーバを再起動しないでデータベースサーバオプションの一部を一時的に上書きできます。

このプロシージャを使用して変更されるオプション値は、データベースサーバが停止するとデフォルト値にリセットされます。データベースサーバを起動するたびにオプション値を変更する場合は、データベースサーバの起動時に対応するデータベースサーバオプションを指定します (存在する場合)。

次のオプション設定が変更可能です。デフォルト値は太字で表示されます。

オプション名	値	その他の情報
AutoMultiProgrammingLevel	**YES**、NO	YES に設定すると、データベースサーバでマルチプログラミングレベルが自動的に調整されます。マルチプログラミングレベルとは、一度にアクティブにできるタスクの最大数を制御するものです。このオプションを NO に設定して、マルチプログラミングレベルを手動で制御するようにした場合、マルチプログラミングレベルの初期値、最小値、最大値を設定できます。
AutoMultiProgrammingLevelStatistics	YES、**NO**	YES に設定すると、マルチプログラミングレベルの自動調整の統計がデータベースサーバのメッセージログに表示されます。
CacheSizingStatistics	YES、**NO**	YES に設定した場合、キャッシュサイズが変更されるたびに、データベースサーバメッセージウィンドウにキャッシュ情報を表示します。

オプション名	値	その他の情報
CockpitDB	コックピットを開始または停止するには、次の手順に従います。 'DBF=*filename* | DBF=AUTO**[;START=ON** | START=OFF**]'** コックピットをコピーするには、次の手順に従います。 **'SAVETO=***filename*[; **ACTION=CONTINUE** | ACTION=SWITCH]'	コックピットを起動または停止する場合や、コックピットデータベースを変更する場合は、CockpitDB オプションを使用します。パラメータを少なくとも 1 つ追加します。SAVETO および ACTION パラメータを使用して DBF および START パラメータを指定することはできません。 • 使用するコックピットのコックピットデータベースを指定するには、DBF パラメータを指定します。コックピット設定はこのファイルに保存されます。存在しないファイルを指定した場合、そのファイルが作成されます。ファイルのフルパスを指定します。相対パスを使用する場合、そのパスは現在の作業フォルダと相対関係となります (または、ディスクサンドボックスが有効化されている場合、相対パスは、メインデータベースファイルがあるディレクトリに対する相対パスとみなされます)。DBF パラメータを使用して指定された値は、その値が変更されるか、データベースサーバがシャットダウンされるまでデフォルト DBF 値になります。 テンポラリデータベースを使用して、コックピットを実行するための DBF=AUTO を設定します。コックピットが停止すると、テンポラリデータベースが削除されるため、コックピット内で施された変更は保存されません。 DBF パラメータは、コックピットが実行されていないときのみ指定できます。DBF パラメータが指定されると、START=OFF も指定されている場合以外は、コックピットが起動されます。START が ON に設定された場合、コックピットが起動します。START が OFF に設定された場合、コックピットが停止します。 • SAVETO パラメータを使用して、現在実行しているコックピットの設定を含むコックピットデータベースを作成します。ファイルのフルパスを指定します。相対パスを使用する場合、そのパスは現在の作業フォルダと相対関係となります (または、ディスクサンドボックスが有効化されている場合、相対パスは、メインデータベースファイルがあるディレクトリに対する相対パスとみなされます)。 このデータベースの使用開始直後に新しいコックピットが使用されるようにするには、ACTION=SWITCH を設定します。設定しない場合、ACTION=CONTINUE が指定されると、コックピットは現在のコックピットデータベースを使用し続けます。 コックピットを起動するには、`CALL sa_server_option('CockpitDB' 'START=ON;DBF=filename');` を実行します。 コックピットを停止するには、`CALL sa_server_option('CockpitDB' 'START=OFF');` を実行します。 コックピットのデフォルトファイル名を変更するには、まずコックピットを停止した後、`CALL sa_server_option('CockpitDB 'START=OFF;DBF=filename);` を実行して新しいデフォルトファイルを定義します。 コックピットデータベースのバックアップコピーを作成するには、`CALL sa_server_option ('CockpitDB' 'SAVETO=c:¥file.db');` を実行します。 新しいコックピットデータベースを作成して、コックピットがこのファイルを使用するようにするには、`CALL sa_server_option ('CockpitDB' 'SAVETO=c:¥file.db;ACTION=SWITCH');` を実行します。
CollectStatistics	**YES**、NO	YES に設定すると、データベースサーバはパフォーマンスモニタの統計情報を収集します。 sa_server_option を NO に設定すると、ステートメントパフォーマンスデータの収集が無効になります。

オプション名	値	その他の情報
ConnsDisabled	YES、**NO**	YES に設定すると、データベースサーバ上のデータベースに対する他の接続は許可されません。
ConnsDisabledForDB	YES、**NO**	YES に設定すると、その他の接続が現在のデータベースに許可されます。
ConsoleLogFile	`filename`	データベースサーバメッセージログ情報の記録に使用されるファイル名。空の文字列を指定すると、ファイルへのロギングが停止します。この値は SQL 文字列なので、パス内の円記号を二重にします。
ConsoleLogMaxSize	`file-size` (バイト単位)	データベースサーバメッセージログ情報の記録に使用されるファイルの最大サイズ (バイト単位)。データベースサーバメッセージログファイルが、このプロパティまたは -on サーバオプションで指定されたサイズに達すると、ファイルが拡張子 `.old` の付いた名前に変更される (既存のファイルが存在する場合は、同じ名前で置換される)。その後、データベースサーバメッセージログファイルが再開されます。
CurrentMultiProgrammingLevel	Integer。デフォルトは **20** です。	データベースサーバのマルチプログラミングレベルを設定します。
DatabaseCleaner	**ON**、OFF	このオプションの設定は、テクニカルサポートの指示があった場合を除いて、変更しないでください。
DeadlockLogging	ON、**OFF**、RESET、CLEAR	デッドロックのロギングを制御します。値 deadlock_logging もサポートされます。次の値がサポートされます。 **ON** デッドロックロギングを有効にします。 **OFF** デッドロックロギングを無効にしますが、デッドロックデータの表示は可能です。 **RESET** デッドロックデータが存在する場合は、記録済みのデッドロックデータをクリアし、その後、デッドロックロギングを有効にします。 **CLEAR** デッドロックデータが存在する場合は、記録済みのデッドロックデータをクリアし、その後、デッドロックロギングを無効にします。 デッドロックロギングが有効になると、sa_report_deadlocks システムプロシージャを使用してデータベースからデッドロック情報を取得することができます。
DebuggingInformation	YES、**NO**	診断メッセージなどのメッセージをトラブルシューティングのために表示します。メッセージは、データベースサーバメッセージウィンドウに表示されます。
DiskSandbox	ON、**OFF**	データベースサーバ上で起動されたデータベースのうちディスクサンドボックス設定が明示的に行われていないすべてのデータベースについて、デフォルトのディスクサンドボックス設定を行います。sa_server_option システムプロシージャを使用したディスクサンドボックス設定の変更内容は、データベースサーバですでに実行されているデータベースには影響しません。sa_server_option システムプロシージャを使用してディスクのサンドボックス設定を変更する場合は、manage_disk_sandbox 機能に対してセキュリティ保護済み機能キーを指定する必要があります。

オプション名	値	その他の情報
DropBadStatistics	**YES**、NO	自動統計管理で、不適切な推定値を返す統計をデータベースから削除できるようにします。
DropUnusedStatistics	**YES**、NO	自動統計管理で、連続 90 日間使用されなかった統計をデータベースから削除できるようにします。
IdleTimeout	Integer、分単位。デフォルトは **240** です。	minutes で指定された時間の間、要求を送信しなかった TCP/IP 接続を切断します。こうすることで、アクティブではない接続がロックを無制限に維持することが回避されます。 最小値は 0 (タイムアウトしない) で、最大値は 32767 です。
IPAddressMonitorPeriod	Integer、秒単位。ポータブルデバイスのデフォルトは **120**、それ以外の場合は **0** です。	新しい IP アドレスをチェックする時間を秒単位で設定します。最小値は 10 で、デフォルト値は 0 です。ポータブルデバイスの場合、デフォルト値は 120 秒です。
LivenessTimeout	Integer、秒単位。デフォルトは **120** です。	接続が維持されていることを確認するため、クライアント/サーバの TCP/IP ネットワークを介して、定期的に活性パケットが送信されます。ネットワークサーバが、活性パケットを検出することなく、LivenessTimeout 時間にわたって実行されると、通信は切断されます。 最小値は 0 (タイムアウトしない) で、最大値は 32767 です。
MaxMultiProgrammingLevel	Integer。デフォルトは CurrentMultiProgrammingLevel の値の 4 倍です。	データベースサーバのマルチプログラミングレベルの最大値を設定します。
MessageCategoryLimit	Integer。デフォルトは **400** です。	sa_server_messages システムプロシージャを使用して取り出すことのできるメッセージの最小数を、重大度レベルとカテゴリごとに設定します。
MinMultiProgrammingLevel	Integer。デフォルトは -gtc サーバオプション値とコンピュータの論理 CPU 数の小さい方の値です。	データベースサーバのマルチプログラミングレベルの最小値を設定します。
OptionWatchAction	**MESSAGE**、ERROR	リストにオプションを設定しようとしたときにデータベースサーバが実行するアクションを指定します。サポートされる値は MESSAGE と ERROR です。OptionWatchAction が MESSAGE に設定されており、OptionWatchList によって指定されるオプションが設定されている場合は、データベースサーバメッセージウィンドウにメッセージが表示され、設定されているオプションがオプションウォッチリストに入っていることが示されます。 OptionWatchAction が ERROR に設定されている場合は、オプションウォッチリストにオプションが入っているためにオプションを設定できないことを示すエラーが返されます。 次のクエリを実行することによって、このプロパティの現在の設定を表示できます。 `SELECT DB_PROPERTY( 'OptionWatchAction' );`

オプション名	値	その他の情報

OptionWatchList	カンマで区切られたデータベースオプションのリスト。	設定したときに通知の対象としたり、データベースサーバでエラーを返したりするデータベースオプションのリストを、カンマで区切って指定します。この文字列の長さは 128 バイトに制限されています。デフォルトでは、空の文字列です。たとえば、次のコマンドは、ウォッチするオプションのリストに automatic_timestamp、float_as_double、tsql_hex_constant の各オプションを追加します。
		`CALL sa_server_option( 'OptionWatchList','automatic_timestamp, float_as_double,tsql_hex_constant' );`
		次のクエリを実行することによって、このプロパティの現在の設定を表示できます。
		`SELECT DB_PROPERTY( 'OptionWatchList' );`

ProcedureProfiling	YES、**NO**、RESET、CLEAR	詳細については、ProcedureProfiling データベースプロパティの説明を参照してください。
ProcessorAffinity	プロセッサの番号、範囲、またはその両方のカンマ区切りのリスト。デフォルトは、すべてのプロセッサを使用するか、または -gta オプションの設定です。	どの論理プロセッサを Windows 上または Linux 上で使用するのかをデータベースサーバに指示します。プロセッサの番号、範囲、またはその両方のカンマ区切りのリストを指定します。範囲の下限の終了ポイントを省略すると、ゼロと見なされます。範囲の上限ポイントを省略すると、オペレーティングシステムで利用可能な最上位の CPU であると見なされます。sa_cpu_topology システムプロシージャで返される in_use カラムには、データベースサーバの現在のプロセッサのアフィニティが格納されます。in_use カラムは、データベースサーバがプロセッサを使用しているかどうかを示します。user_selected カラムは、-gta データベースサーバオプションまたは ProcessorAffinity サーバプロパティでどの物理プロセッサが指定されたかを示します。または、ProcessorAffinity データベースサーバプロパティの値はクエリすることも可能です。 次の場合には、指定されたすべての論理プロセッサーをデータベースサーバが使用しない場合があります。 • 指定された論理プロセッサの中に、存在しないもの、またはオフラインのものがある場合。 • ライセンスで使用が許可されない場合。 無効なプロセッサ ID を指定すると、sa_server_option がエラーを返します。
ProfileFilterConn	`connection-id`	他の接続がデータベースを使用する処理を阻害することなく、特定の接続 ID に関するプロファイル情報を取得するように、データベースサーバに指示します。接続フィルタが有効な場合、`SELECT PROPERTY( 'ProfileFilterConn' )` の戻り値はモニタ対象の接続の接続 ID。ID が指定されていない場合、または接続フィルタが無効な場合、戻り値は -1 です。
ProfileFilterUser	`user-id`	データベースサーバに対して、特定のユーザ ID のプロファイリング情報を取得するよう指示します。

オプション名	値	その他の情報
PropertyHistoryList	**ON**、OFF、NONE、データベースサーバプロパティのカンマ区切りのリスト	**ON** PropertyHistoryList が有効な場合、プロパティのデフォルトリストが追跡されます。デフォルトは ON です。 **OFF** PropertyHistoryList が無効な場合、データベースサーバのプロパティの追跡が無効になります。 **NONE** このプロパティを NONE に設定すると、プロパティの追跡が有効になりますが、プロパティはデータベースサーバにより追跡されません。データベースにより要求されたプロパティのみが追跡されます。 データベースサーバプロパティのカンマ区切りのリスト 追跡するデータベースサーバプロパティのカンマ区切りのリストを指定します。
PropertyHistorySize	time、memory-size、MAX、DEFAULT	追跡されたプロパティ値を保存する最小時間、または追跡されたプロパティ値の保存に使用するメモリの最大量を指定します。このプロパティを時間に設定するには、'[HH:]MM:SS' 形式を使用します。このプロパティをメモリサイズに設定するには、メモリサイズをバイト単位で指定します。たとえば、1M などです。デフォルト値は '00:10:00' (10 分) です。ただし、その時間が最大サイズ制限に違反している場合を除きます (この場合は MAX がデフォルト値として使用されます)。 メモリの固定最大量の使用を要求するには、MAX を指定します。最大メモリは、キャッシュの 2% または 256 MB のいずれか小さい方になります。 PropertyHistoryList が設定されている場合、プロパティ履歴の追跡に使用されるメモリ量が更新されます。プロパティ履歴の保存に許容されるメモリの固定最大量を超えた場合、エラーが発生します。使用されるメモリの量が減少し、指定されたプロパティの履歴を追跡するためのメモリが十分にない場合、エラーが発生します。エラーが発生した場合、プロパティ履歴が前の値に戻ります。 PropertyHistoryList により指定されたすべてのプロパティを追跡するためのメモリが十分にない場合、エラーが戻され、プロパティは追跡されたプロパティのリストに追加されません。
QuittingTime	有効な日付と時刻。	データベースサーバに、指定された時間にサーバを停止するよう指示します。
RememberLastPlan	YES、**NO**	接続で最後に実行されたクエリの長いテキストプランをキャプチャするように、データベースサーバに指示します。この設定は、-zp サーバオプションでも制御されます。 RememberLastPlan が ON の場合は、LastPlanText 接続プロパティの値を問い合わせることで、この接続で最後に実行されたクエリの計画のテキスト表現を取得します。 `SELECT CONNECTION_PROPERTY( 'LastPlanText' );`

オプション名	値	その他の情報

RememberLastStatement	YES、**NO**	データベースサーバに、サーバ上で実行されている各データベースに関して最後に作成された SQL 文を取得するように指示します。ストアドプロシージャコールの場合、プロシージャ内の文ではなく、最も外側のプロシージャコールのみが表示されます。
		
		RememberLastStatement が ON の場合は、LastStatement 接続プロパティの値を問い合わせることで、接続に関する LastStatement の現在の値を取得できます。
		
		`SELECT CONNECTION_PROPERTY( 'LastStatement' );`
		
		クライアントでの文のキャッシュが有効であり、キャッシュされた文が再使用されているとき、このプロパティは空の文字列を返します。
		
		RememberLastStatement が ON の場合、次の文によって指定された接続に対して最後に作成された文が返されます。
		
		`SELECT CONNECTION_PROPERTY( 'LastStatement', connection-id );`
		
		sa_conn_activity システムプロシージャは、すべての接続に対して同じ情報を返します。
		
		⚠ 警告
		
		-zl が指定されている場合、または RememberLastStatement サーバ設定がオンになっている場合、すべてのユーザが sa_conn_activity システムプロシージャを呼び出すか、LastStatement 接続プロパティの値を取得して、他のユーザが最後に作成した SQL 文を見つけることができます。このオプションは注意して使用してください。不要になったらオフにします。

RequestFilterConn	connection-id, -1	要求ログ情報をフィルタして、特定の接続の情報のみ記録されるようにします。このフィルタリングによって、多くのアクティブな接続または複数のデータベースのあるデータサーバを監視するときに、要求ログファイルのサイズを削減できます。次の文を実行して、接続 ID を取得できます。
		
		`CALL sa_conn_info( );`
		
		接続 ID を取得した後で特定の接続をロギングするには、次の文を実行します。
		
		`CALL sa_server_option( 'RequestFilterConn', connection-id );`
		
		フィルタリングは、明示的にリセットされるかデータベースサーバがシャットダウンされるまで有効なままになります。フィルタをリセットするには、次の文を使用します。
		
		`CALL sa_server_option( 'RequestFilterConn', -1 );`

オプション名	値	その他の情報

RequestFilterDB	database-id, -1	要求ログ情報をフィルタして、特定のデータベースの情報のみ記録されるようにします。これによって、複数のデータベースのあるサーバを監視するときに、要求ログファイルのサイズを削減できます。目的のデータベースに接続しているときに次の文を実行して、データベース ID を取得できます。
		`SELECT CONNECTION_PROPERTY( 'DBNumber' );`
		特定のデータベースの情報のみをロギングするには、次の文を実行します。
		`CALL sa_server_option( 'RequestFilterDB', database-id );`
		フィルタリングは、明示的にリセットされるかデータベースサーバがシャットダウンされるまで有効なままになります。フィルタをリセットするには、次の文を使用します。
		`CALL sa_server_option( 'RequestFilterDB', -1 );`

RequestLogFile	filename	要求情報の記録に使用されるファイルの名前。空の文字列を指定すると、要求ログファイルへのロギングが停止します。要求のロギングが有効でも、要求のログファイルを指定しなかった場合、または空の文字列に設定されている場合、サーバは要求をデータベースサーバメッセージウィンドウにロギングします。この値は SQL 文字列なので、パス内の円記号を二重にします。 クライアントでの文のキャッシュが有効であり、かつ、ログが `tracetime.pl` Perl スクリプトを使用して分析される場合は、要求ログが取得される間、max_client_statements_cached オプションを 0 に設定して、クライアントでの文のキャッシュを無効にする必要があります。

オプション名	値	その他の情報
RequestLogging	SQL、HOSTVARS、PLAN、PROCEDURES、TRIGGERS、OTHER、BLOCKS、REPLACE、ALL、YES、**NONE**、NO	データベースサーバオプション -zr と -zo とともにトラブルシューティングで使用するためにデータベースサーバに送信される個々の SQL 文のロギングを有効にします。次の値をプラス記号 (+) またはカンマで区切って組み合わせた値です。 **PLAN** 実行プランのロギングを有効にします (短期)。プロシージャ (PROCEDURES) のロギングが有効な場合は、プロシージャの実行プランも記録されます。 **HOSTVARS** ホスト変数の値のロギングを有効にします。HOSTVARS を指定した場合、SQL にリストされている情報もロギングされます。 **PROCEDURES** プロシージャ内から実行されている文のロギングを有効にします。 **TRIGGERS** トリガ内から実行されている文のロギングを有効にします。 **OTHER** SQL に含まれないその他の要求タイプのロギングを有効にします (FETCH や PREFETCH など)。ただし、OTHER を指定して SQL を指定しない場合、SQL+OTHER を指定した場合と同じです。OTHRE を含めると、要求ログファイルが急速に拡大し、サーバのパフォーマンス低下につながることがあります。 **BLOCKS** 別の接続で接続がブロックされたときと、接続のブロックが解除されたときに表示する詳細のロギングを有効にします。 **REPLACE** ロギングの開始時に、既存の要求ログは同じ名前を持つ新規の (空の) ログで置換されます。それ以外の場合、既存の要求ログが開き、新規エントリがファイルの末尾に追加されます。 **ALL** すべてのサポート情報をロギングします。この値は、SQL+PLAN+HOSTVARS+PROCEDURES+TRIGGERS+OTHER+BLOCKS を指定した場合と同じです。この設定では、要求ログファイルが急速に拡大し、サーバのパフォーマンス低下につながることがあります。 **NO** または **NONE** 要求ログに対するロギングを無効にします。 次のクエリを実行することによって、このプロパティの現在の設定を表示できます。 `SELECT PROPERTY( 'RequestLogging' );`

オプション名	値	その他の情報
RequestLogMaxSize	`file-size`（バイト単位）	要求ログ情報の記録に使用されるファイルのバイト単位での最大サイズ。0 を指定した場合は、要求ロギングファイルの最大サイズは適用されず、ファイルの名前は変更されません。この値はデフォルトです。 要求ログファイルが、sa_server_option システムプロシージャまたは -zs サーバオプションで指定されたサイズに達すると、ファイルが拡張子 `.old` の付いた名前に変更されます（既存のファイルが存在する場合は、同じ名前で置換されます）。これにより、要求ログファイルは再起動されます。
RequestLogNumFiles	Integer	保持する要求ログファイルのコピーの数。 要求ロギングが長期間にわたって有効になっていると、要求ログファイルが大きくなることがあります。-zn オプションを使用すると、保持する要求ログファイルのコピー数を指定できます。
RequestTiming	YES、**NO**	データベースに、新しい各接続のタイミング情報を保守するように指示します。この機能はデフォルトでオフになっています。オンにすると、データベースサーバはすべての新しい接続の累積タイマを管理します。このタイマは、接続が確立した状態でのサーバステータスごとの時間を示すものです。変更は新しい接続についてのみ有効で、各接続の期間持続します。 sa_performance_diagnostics システムプロシージャを使用して、このタイミング情報の概要を取得できます。または、次の接続プロパティを調べて、各値を取得できます。 • ReqCountUnscheduled • ReqTimeUnscheduled • ReqCountActive • ReqTimeActive • ReqCountBlockIO • ReqTimeBlockIO • ReqCountBlockLock • ReqTimeBlockLock • ReqCountBlockContention • ReqTimeBlockContention RequestTiming サーバプロパティがオンであると、追加のカウンタを管理するため、要求ごとに多少のオーバヘッドがかかります。

オプション名	値	その他の情報

SecureFeatures	feature-list	すでに動作しているデータベースサーバのセキュリティ保護済み機能を管理できます。feature-list は、機能名または機能セットのカンマ区切りリスト。このリストに機能を追加することで、追加した機能の可用性を制限します。セキュリティ保護済み機能のリストから項目を削除するには、機能名の前にマイナス記号 (-) を指定します。
		
		たとえば、2 つの機能を保護するには、次の構文を使用します。
		
		`CALL sa_server_option('SecureFeatures', 'console_log,webclient_log' );`
		
		この文を実行すると、変更内容に応じてセキュリティ保護済み機能のリストが設定されます。
		
		LOCAL 機能セットをセキュリティ保護しながら LOCAL_IO 機能サブセットを除外するには、次の構文を使用します。
		
		`CALL sa_server_option('SecureFeatures', 'local,-local_io' );`
		
		sa_server_option('SecureFeatures',...) を呼び出す場合は、接続で MANAGE_FEATURES 機能が有効になっている必要があります。-sf データベースサーバオプションは、デフォルトで MANAGE_FEATURES をセキュリティ保護します。SecureFeatures オプションを設定する現在のユーザ/接続において、明示的に有効化された機能がなく、機能のリストがその接続によって無効化された場合、接続は直ちに影響を受けます。
		
		機能へのアクセスを許可または禁止する変更を行うと、データベースサーバで直ちに有効になります。sa_server_option システムプロシージャを実行する接続は、その接続が使用しているセキュリティ保護済み機能と、指定した機能へのアクセスが許可されているかどうかによって、変更が有効または無効になります。

SingleCLRInstanceVersion	35、40、45	このオプションは、データベースサーバで実行されているすべてのデータベースにデータベースサーバが 1 つの CLR 外部環境を使用するかどうかを指定します。このオプションの値は、起動するバージョンを指定します。 **注記** データベースサーバが CLR 外部環境を起動した後にこのオプションを変更することはできません。
SingleJVMLocation	filename	このオプションは、データベースサーバで実行されているすべてのデータベースにデータベースサーバが使用する Java VM の場所を指定します。このオプションは、データベースサーバが -sjvm データベースサーバオプションを使用して起動された場合、または UseSingleJVMInstance データベースサーバオプションが ON に設定されている場合のみ設定できます。
StatisticsCleaner	**ON**、OFF	統計クリーナーでは、テーブルに対してスキャンを実行して、不適切な推定値を示す統計を修正します。デフォルトでは、統計クリーナーはバックグラウンドで実行され、パフォーマンスへの影響は最小限に抑えられます。 統計クリーナーをオフにしても統計ガバナーは無効になりませんが、統計クリーナーがオフのときには、統計はクエリの実行時にのみ作成または修正されます。

オプション名	値	その他の情報
TopologyAwareScheduling	**ON**、OFF	トポロジを考慮したスケジューリングをオンまたはオフにします。トポロジを考慮したスケジューリングでは、タスクは同じソケットの別のコアを使用しようとする前に、ソケットごとに 1 つのコアを使用します (タスクは、同じコアの他のスレッドを使用しようとする前に、コアごとに 1 つのコアを使用するようにスケジュールされます)。
UseSingleJVMInstance	ON、**OFF**	このオプションは、データベースサーバで実行されているすべてのデータベースにデータベースサーバが 1 つの Java VM を使用するかどうかを指定します。 **注記** データベースサーバが Java VM を起動した後にこのオプションを変更することはできません。
WebClientLogFile	`filename`	Web サービスクライアントログファイルの名前。Web サービスクライアントログファイルは、-zoc サーバオプションや WebClientLogFile プロパティを使用して、ファイル名を設定またはリセットするたびにトランケートされます。この値は文字列なので、パス内の円記号を二重にします。
WebClientLogging	ON、**OFF**	このオプションは、Web サービスクライアントのロギングを有効または無効にします。ログに記録される情報には、HTTP の要求と応答のデータが含まれています。ON を指定すると Web サービスクライアントログファイルへのロギングが開始され、OFF を指定するとファイルへのロギングが中止されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

次のオプションを使用する場合は、要求ロギングに関連する MANAGE PROFILING システム権限を持っている必要があります。

- ProcedureProfiling
- ProfileFilterConn
- ProfileFilterUser
- RequestFilterConn
- RequestFilterDB
- RequestLogFile
- RequestLogging
- RequestLogMaxSize
- RequestLogNumFiles

その他のすべてのオプションでは、SERVER OPERATOR システム権限が必要になります。

SingleJVMLocation オプションを使用するには、MANAGE ANY EXTERNAL ENVIRONMENT システム権限も必要です。

関連する動作

なし。

例

次の文を使用した場合は、キャッシュサイズを変更するたびに、データベースサーバメッセージウィンドウにキャッシュ情報が表示されるようになります。

```
CALL sa_server_option( 'CacheSizingStatistics', 'YES' );
```

次の文は、現在のデータベースへの新しい接続を禁止します。

```
CALL sa_server_option( 'ConnsDisabledForDB', 'YES' );
```

次の文は、すべての SQL 文、プロシージャの呼び出し、プラン、イベントのブロックとブロック解除のロギングを有効にし、新しい要求ログを開始します。

```
CALL sa_server_option( 'RequestLogging', 'SQL+PROCEDURES+BLOCKS+PLAN+REPLACE' );
```

関連情報

sa_performance_diagnostics システムプロシージャ [1576 ページ]
sa_db_option システムプロシージャ [1482 ページ]

1.6.8.86 sa_set_http_header システムプロシージャ

Web サービスが HTTP 応答ヘッダを設定できるようにします。

構文

```
sa_set_http_header(
fldname
, val
[, instance ]
)
```

パラメータ

fldname

この CHAR(128) パラメータを使用して、HTTP ヘッダフィールドのいずれかの名前を含む文字列を指定します。

val

この LONG VARCHAR パラメータを使用して、指定されたパラメータに設定する値を指定します。応答ヘッダを NULL に設定すると、そのヘッダは事実上削除されます。

instance

この UNSIGNED INT パラメータを使用して、設定する HTTP 応答ヘッダのインスタンスを指定します。デフォルトは 1 です。

備考

特別なヘッダフィールド @HttpStatus を設定すると、要求によってステータスコードが返されるように設定されます。ステータスコードは応答コードとも呼ばれます。たとえば、次のスクリプトはステータスコードを 404 Not Found に設定します。

```
CALL sa_set_http_header( '@HttpStatus', '404' );
```

3 桁のステータスコードを指定し、コロンで区切ったテキストメッセージを任意で追加することで、ユーザ定義のステータスメッセージを作成できます。たとえば、次のスクリプトは、メッセージ付きのステータスコード "999 User Code" を出力します。

```
CALL sa_set_http_header( '@HttpStatus', '999:User Code' );
```

注記

LogOptions プロトコルオプションを使用してログに記録している場合、ユーザ定義のステータステキストメッセージはデータベースの文字セットには変換されません。

エラーメッセージの本文は、自動的に挿入されます。有効な HTTP エラーコードだけが使用できます。ステータスに無効なコードを設定すると、SQL エラーが発生します。

sa_set_http_header プロシージャは、呼び出されたときに、ヘッダフィールドの既存のヘッダ値を必ず上書きします。

データベースサーバによって自動的に生成された応答ヘッダは削除できます。たとえば、次のコマンドは Expires 応答ヘッダを削除します。

```
CALL sa_set_http_header( 'Expires', NULL );
```

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、Set-Cookie ヘッダフィールドを type=chocolate に設定し、ヘッダの 3 番目のインスタンスを指定します。

```
CALL sa_set_http_header( 'Set-Cookie', 'type=chocolate', 3 );
```

関連情報

Web サービス関数 [212 ページ]
NEXT_HTTP_RESPONSE_HEADER 関数 [Web サービス] [454 ページ]
HTTP_RESPONSE_HEADER 関数 [Web サービス] [397 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.87 sa_set_http_option システムプロシージャ

プロセス制御のため、Web サービスが HTTP オプションを設定できるようにします。

構文

```
sa_set_http_option(
optname
, val
)
```

パラメータ

optname

この CHAR(128) パラメータを使用して、HTTP オプションのいずれかの名前を含む文字列を指定します 。

サポートされるオプションは次のとおりです。

CharsetConversion

このオプションを使用して、結果セットをデータベースの文字セットエンコードからクライアントの文字セットエンコードに自動的に変換するかどうかを制御します。指定できる値は、ON と OFF だけです。デフォルト値は ON です。

AcceptCharset

このオプションを使用して、応答の文字セットエンコードに関する Web サーバの優先度を指定します。1 つ以上の文字セットエンコードを優先度順に指定できます。このオプションの構文は、RFC2616 Hypertext Transfer Protocol の HTTP Accept-Charset 要求ヘッダフィールドの仕様に使用する構文に準拠します。

Web ブラウザなどの HTTP クライアントでは、Accept-Charset 要求ヘッダを使用して、優先度順の文字セットエンコードのリストを指定できます。必要に応じて、各エンコードには、対応する品質値 (q=`qvalue`) を指定できます。こ

の品質値は、そのエンコードに対するクライアントの優先度を表します。デフォルトでは、品質値は 1 (q=1) です。次に例を示します。

```
Accept-Charset: iso-8859-5, utf-8;q=0.8
```

AcceptCharset HTTP オプション値内のプラス記号 (+) は、現在のデータベースの文字セットエンコードを表すショートカットとして使用できます。また、プラス記号は、クライアントのリストでもエンコードが指定されている場合に、クライアントによって割り当てられている品質値に関係なく、データベースの文字セットエンコードを優先する必要があることも示します。

AcceptCharset HTTP オプション内のアスタリスク (*) は、クライアントとサーバのリストに共通部分がない場合に、クライアントで優先される文字セットエンコードがサーバでもサポートされていれば、Web サービスでその文字セットエンコードを使用する必要があることを示します。

応答の送信時には、クライアントと Web サービスの両方で優先される最初の文字セットエンコードが使用されます。クライアントの優先度順が最初に考慮されます。共通のエンコード優先度が存在しない場合は、Web サービスのリストにアスタリスク (*) がなければ、Web サービスで最も優先されるエンコードが使用されます。アスタリスクがある場合は、クライアントで最も優先されるエンコードが使用されます。

AcceptCharset HTTP オプションを使用しない場合は、クライアントで指定され、サーバでサポートされている最も優先度の高い文字セットエンコードが使用されます。クライアントで指定されたどのエンコードもサポートされていない場合 (またはクライアントが Accept-Charset 要求ヘッダを送信しない場合) は、データベースの文字セットエンコードが使用されます。

クライアントが Accept-Charset ヘッダを送信しない場合は、次のいずれかのアクションが実行されます。

- AcceptCharset HTTP オプションが指定されていない場合、Web サーバではデータベースの文字セットエンコードを使用します。
- AcceptCharset HTTP オプションが指定されている場合、Web サーバでは最も優先度の高い文字セットエンコードを使用します。

クライアントが Accept-Charset ヘッダを送信する場合は、次のいずれかのアクションが実行されます。

- AcceptCharset HTTP オプションが指定されていない場合、Web サーバではクライアントで優先される文字セットエンコードのいずれかを使用しようとします (最も優先度の高いエンコードから開始)。クライアントで優先されるどのエンコードも Web サーバでサポートされていない場合、Web サーバではデータベースの文字セットエンコードを使用します。
- AcceptCharset HTTP オプションが指定されている場合、Web サーバでは両方の優先度リストで共通する最初の文字セットエンコードを使用しようとします (クライアントで最も優先されるエンコードから開始)。たとえば、クライアントが送信する Accept-Charset ヘッダリストのエンコードの優先度順が iso-a、iso-b、iso-c であり、Web サーバでの優先度順が iso-b、iso-a、iso-c の場合、iso-a が選択されます。

  ```
  Web client: iso-a, iso-b, iso-c
  Web server: iso-b, iso-a, iso-c
  ```

  2 つのリストに共通部分がない場合、Web サーバで最初に優先される文字セットが使用されます。次の例では、エンコード iso-d が使用されます。

  ```
  Web client: iso-a, iso-b, iso-c
  Web server: iso-d, iso-e, iso-f
  ```

  AcceptCharset HTTP オプションにアスタリスク ('*') が含まれている場合は、クライアントのエンコードが優先されるため、iso-a が使用されます。基本的に、アスタリスクを使用すると、2 つのリストに共通部分がないという状況がなくなります。

理想的な状況は、クライアントと Web サービスの両方でデータベースの文字セットエンコードが使用される場合です。このような場合、文字セットの変換の必要性がなくなり、Web サーバの応答時間が向上します。

CharsetConversion オプションを OFF に設定している場合、AcceptCharset の処理は実行されません。

SessionID

このオプションを使用して、HTTP セッションの作成、削除、または名前の変更を行います。Web サービスでこのオプションを設定して HTTP セッションを作成するときにはデータベース接続は維持されますが、サーバの再起動時にはセッションは維持されません。すでにセッションコンテキスト内にある場合、この呼び出しによってセッションの名前が新しいセッション ID に変更されます。NULL 値を使用して呼び出した場合、Web サービスが終了するとセッションは削除されます。

生成されるセッションキーの長さは 128 文字に制限され、複数のデータベースがロードされている場合はデータベース間でユニークになります。

SessionTimeout

このオプションを使用して、非アクティブ状態の HTTP セッションを維持する時間を分単位で指定します。このタイムアウト期間は、HTTP 要求で指定のセッションが使用されると常にリセットされます。SessionTimeout を過ぎると、セッションは自動的に削除されます。HTTP セッションが作成されると、デフォルトのタイムアウト期間が http_session_timeout データベースオプションの現在の設定から取得されます。このデフォルトは、SessionTimeout オプションを使用して上書きできます。使用できる値は 1 ～ 525600 (365 日) です。

val

この LONG VARCHAR パラメータを使用して、指定されたオプションに設定する値を指定します。

備考

Web サービスを処理する文またはプロシージャ内でこのプロシージャを使用して、オプションを設定します。

sa_set_http_option が Web サービスによって開始されたプロシージャ内から呼び出され、オプションまたはオプションの値のいずれかが無効な場合は、エラーが返ります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、sa_set_http_option を使用して、データベースの文字セットエンコードに関する Web サービスの優先度を指定する方法を示します。2 番目の選択肢として UTF-8 エンコードが指定されています。アスタリスク (*) は、クライアントで

最も優先される文字セットエンコードが Web サーバでサポートされている場合に、Web サービスでその文字セットエンコードが使用されることを示しています。

```
CALL sa_set_http_option( 'AcceptCharset', '+,UTF-8,*');
```

次の例は、sa_set_http_option を使用して、Web サービスで使用されている文字エンコードを正しく識別する方法を示します。この例では、Web サーバは 1251CYR データベースに接続されており、キリル語のアルファベットを含む HTML ドキュメントを任意の Web ブラウザに提供するように準備されています。

```
CREATE OR REPLACE PROCEDURE cyrillic_html()
RESULT (html_doc XML)
BEGIN
  DECLARE pos INT;
  DECLARE charset VARCHAR(30);
  CALL sa_set_http_option( 'AcceptCharset', 'iso-8859-5, utf-8' );
  SET charset = CONNECTION_PROPERTY( 'CharSet' );
  -- Change any IANA labels like ISO_8859-5:1988
  -- to ISO_8859-5 for Firefox.
  SET pos = LOCATE( charset, ':' );
  IF pos > 0 THEN
    SET charset = LEFT( charset, pos - 1 );
  END IF;
  CALL sa_set_http_header( 'Content-Type', 'text/html; charset=' ||
        charset );
  SELECT  '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">' ||
    XMLCONCAT(
      XMLELEMENT('HTML',
        XMLELEMENT('HEAD',
          XMLELEMENT('TITLE', 'Cyrillic characters')
        ),
        XMLELEMENT('BODY',
          XMLELEMENT('H1', 'First 5 lowercase Russian letters'),
          XMLELEMENT('P', UNISTR('¥u0430¥u0431¥u0432¥u0433¥u0434'))
        )
      )
    );
END;
CREATE SERVICE cyrillic
TYPE 'RAW'
AUTHORIZATION OFF
USER DBA
AS CALL cyrillic_html();
```

Firefox などの Web ブラウザから Web サービスに配信される、次の Accept-Charset ヘッダを例に、使用する正しい文字セットエンコードを確立するプロセスについて説明します。このヘッダは、ブラウザでは ISO-8859-1 と UTF-8 のエンコードが優先されるものの、他のエンコードも処理できることを示しています。

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

転送される Web ページにキリル語の文字が含まれているため、Web サービスでは ISO-8859-1 文字セットエンコードは受け入れられません。Web サービスでは、sa_set_http_option の呼び出しで指定されているように、ISO-8859-5 または UTF-8 エンコードが優先されます。この例では、UTF-8 エンコードが選択されます (両方で許容されるため)。データベース接続プロパティ CharSet は、どのエンコードが Web サービスで選択されたかを示します。sa_set_http_header プロシージャは、Web ブラウザに対して HTML ドキュメントのエンコードを示すために使用します。

```
Content-Type: text/html; charset=UTF-8
```

Web ブラウザで Accept-Charset が指定されていない場合、Web サービスではデフォルトで最初の優先エンコード ISO-8859-5 が使用されます。sa_set_http_header プロシージャは、HTML ドキュメントのエンコードを示すために使用します。

```
Content-Type: text/html; charset=ISO_8859-5
```

次の例では、ユニークな HTTP セッション識別子を設定します。

```
BEGIN
  DECLARE sessionid VARCHAR(30);
  DECLARE tm TIMESTAMP;
  SET tm = NOW(*);
  SET sessionid = 'MySessions_' ||
      CONVERT( VARCHAR, SECONDS(tm)*1000 + DATEPART(millisecond,tm));
  SELECT sessionid;
  CALL sa_set_http_option('SessionID', sessionid);
END;
```

次の例では、HTTP セッションのタイムアウトを 5 分に設定します。

```
CALL sa_set_http_option('SessionTimeout', 5);
```

関連情報

Web サービス関数 [212 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.88 sa_set_soap_header システムプロシージャ

SOAP 応答の SOAP ヘッダの設定を許可します。このプロシージャは、SOAP Web サービスから呼び出されたストアドプロシージャ内で使用されます。

構文

```
sa_set_soap_header(
fldname
, val
)
```

パラメータ

fldname

特定のヘッダエントリを参照するときに使用する一意な文字列であるヘッダキーを指定する CHAR(128) パラメータ (`val` の localname と同じである必要はありません)。

val

この LONG VARCHAR パラメータを使用して、トップレベルのヘッダエントリと、SOAP ヘッダ要素の範囲内にある子供の未加工 XML を指定します。

備考

このプロシージャで設定されたすべての SOAP ヘッダエントリは、SOAP 応答メッセージを送信するときに、SOAP ヘッダ要素内でシリアル化されます。NULL の `val` はシリアル化されません。SOAP 応答のヘッダエントリが存在しない場合、SOAP エンベロープ内で内包するヘッダ要素は作成されません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、Hello に対する SOAP ヘッダのウェルカムメッセージを設定します。

```
CALL sa_set_soap_header( 'welcome', '<welcome>Hello</welcome>' )
```

関連情報

Web サービス関数 [212 ページ]
Web サービスシステムプロシージャ [1431 ページ]

1.6.8.89 sa_set_tracing_level システムプロシージャ (廃止予定)

診断トレース機能は廃止予定です。SQL Anywhere プロファイラ を使用して、データベースの問題を診断します。診断トレーシングテーブルに格納するトレーシング情報のレベルを初期化します。

構文

```
sa_set_tracing_level(
level
[, specified_scope
[, specified_name
[, do_commit ] ] ]
)
```

パラメータ

level

この INTEGER パラメータを使用して、実行する診断トレーシングのレベルを指定します。使用できる値は、次のとおりです。

0

トレーシングデータを生成しません。このレベルは、トレースセッションを開いたままにしますが、トレーシングデータを診断トレーシングテーブルに送信しません。

1

基本レベルのトレーシングを設定します。

2

中間レベルのトレーシングを設定します。

3

高レベルのトレーシングを設定します。

specified_scope

任意の LONG VARCHAR パラメータを使用して、トレーシングのスコープを指定します。たとえば、USER、DATABASE、CONNECTION_NAME、TRIGGER などを指定します。デフォルトは NULL です。

specified_name

任意の LONG VARCHAR パラメータを使用して、`specified_scope` に示されたオブジェクトの識別子を指定します。デフォルトは NULL です。

do_commit

任意の TINYINT パラメータを使用して、このプロシージャによって挿入されるローを自動的にコミットするかどうかを指定します。デフォルトは 1 です。1 を指定すると、ローは自動的にコミットされます (この設定をお奨めします)。0 を指定すると、ローは自動的にコミットされません。

備考

このプロシージャは、sa_diagnostic_tracing_level テーブルのローを置換し、トレーシングレベルとスコープはプロシージャの呼び出し時に指定した設定に変更されます。

レベル 0 に設定しても、トレースセッションは停止しません。その代わり、トレースセッションはトレーシングデータベースに所属したままで、トレーシングデータは送信されません。レベルが 0 のときでもトレースセッションはアクティブです。

このシステムプロシージャは、プロファイル対象のデータベースから呼び出す必要があります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、DIAGNOSTICS システムロールと MANAGE PROFILING システム権限が必要です。

関連する動作

なし

例

次の例は、トレースレベルを 1 に設定します。パフォーマンスカウンタデータや実行される文のサンプル用としてデータベース全体をプロファイルします。

```
CALL sa_set_tracing_level( 1 );
```

次の例は、トレーシングレベルを 3 に設定し、ユーザ AG84756 を指定します。AG84756 に関連付けられたアクティビティのみがトレースされます。

```
CALL sa_set_tracing_level( 3, 'user', 'AG84756' );
```

関連情報

sa_diagnostic_tracing_level テーブル (廃止予定) [1426 ページ]

1.6.8.90 sa_snapshots システムプロシージャ

現在アクティブなスナップショットのリストを返します。

構文

```
sa_snapshots( )
```

結果セット

カラム名	データ型	説明
connection_num	INTEGER	スナップショットが実行されている接続の接続 ID。
start_sequence_num	UNSIGNED BIGINT	スナップショットを識別するユニークな番号。
statement_level	BIT	スナップショットを statement-snapshot または readonly-statement-snapshot で作成した場合は true。それ以外の場合は false。

備考

文のスナップショットは、1 つの接続に複数存在できます。文のスナップショットアイソレーションレベルで実行されるネストされた文、またはインターリーブされた文の場合、それぞれ最初の読み取りまたは更新で異なる文のスナップショットが開始されます。

トランザクションのスナップショットは、通常は 1 つの接続に 1 つしかありません (statement_level=0 で、1 つの connection in sa_snapshots につき 1 エントリ)。ただし、カーソルに関連付けられたスナップショットは、カーソルの最初のフェッチ後も変化しません。また、WITH HOLD で開いたカーソルは、コミットまたはロールバックの間、開いたままです。カーソルがスナップショットに関連付けられている場合は、そのスナップショットも継続されます。そのため、複数トランザクションのスナップショットが同じ connection_num に対して存在する可能性があります。つまり、1 つは現在のトランザクションのスナップショット、WITH HOLD カーソルのために継続している古いトランザクションのスナップショットには 1 つ以上という場合です。

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次のクエリを実行して、現在アクティブなスナップショットを識別できます。

```
CALL sa_snapshots( );
```

関連情報

sa_transactions システムプロシージャ [1647 ページ]

1.6.8.91 sa_split_list システムプロシージャ

デリミタで区切った値の文字列を使用して、ローのセットを返します (各値に 1 つのロー)。

構文

```
sa_split_list(
str
  [, delim
  [, maxlen ] ]
)
```

パラメータ

str

この LONG VARCHAR パラメータを使用して、分割する値を含む文字列を指定します。各値は `delim` で区切られます。

delim

任意の CHAR(10) パラメータを使用して、`str` の値を分割するときに使用するデリミタを指定します。デリミタは、任意の文字の文字列 (10 バイトまで) にできます。`delim` が指定されない場合、デフォルトでカンマが使用されます。

maxlen

任意の INTEGER パラメータを使用して、戻り値の最大長を指定します。たとえば、`maxlen` が 3 に設定されている場合、結果セットの値は 3 文字の長さにトランケートされます。0 (デフォルト) を指定すると、値は任意の長さで指定できます。

結果セット

カラム名	データ型	説明
line_num	INTEGER	ローのシーケンス番号です。
row_value	LONG VARCHAR	文字列の値です。必要に応じて、`maxlen` にトランケートされます。

備考

sa_split_list プロシージャは、デリミタで区切られた値のリストが含まれる文字列を受け取り、1 つのローに 1 つの値を入れた結果セットを返します。これは、LIST 関数 [集合] が実行するアクションの逆です。文字列が次の値の場合、row_value には空の文字列が返されます。

- `delim` で始まります
- 文字列の途中で `delim` が 2 つ連続するインスタンスを含みます
- `delim` で終わります

入力文字列内の空白スペースには意味があります。デリミタがスペース文字の場合、入力文字列に余分なスペースがあると結果セットの中に余分なローが入ります。デリミタがスペース文字ではない場合、入力文字列のスペースは結果セット内の値から削除されません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次のクエリは、色が黒い製品のリストを返します。

```
SELECT list( Name )
  FROM Products
  WHERE Color = 'Black';
```

list (Products.Name)
Tee Shirt,Baseball Cap,Visor,Shorts

次の例では、sa_split_list プロシージャを使用して、集約されたリストから元の結果セットを返します。

```
SELECT *
```

```
  FROM sa_split_list( 'Tee Shirt,Baseball Cap,Visor,Shorts' );
```

line_num	row_value
1	Tee Shirt
2	Baseball Cap
3	Visor
4	Shorts

次の例は、ワードごとに 1 つのローを返します。row_value が空の文字列のローを返さないようにするには、WHERE 句を指定する必要があります。

```
SELECT *
  FROM sa_split_list( 'one||three|four||six|', '|' )
  WHERE row_value <> '';
```

line_num	row_value
1	one
3	three
4	four
6	six

次の例は、ProductsWithColor というプロシージャが作成されます。呼び出されると、ProductsWithColor プロシージャは sa_split_list を使用して、ユーザが指定した色の値を解析し、Products テーブルの Color カラムを検索し、ユーザが規定した色のいずれかに一致する製品ごとに名前、説明、サイズ、色を返します。

次のプロシージャを呼び出した結果は、色が白または黒であるすべての製品の名前、説明、サイズ、色です。

```
CREATE OR REPLACE PROCEDURE ProductsWithColor( IN color_list LONG VARCHAR )
BEGIN
  SELECT Name, Description, Size, Color
  FROM Products
  WHERE Color IN ( SELECT row_value FROM sa_split_list( color_list ) );
END;
SELECT * from ProductsWithColor( 'white,black' );
```

関連情報

LIST 関数 [集合] [420 ページ]

1.6.8.92 sa_stack_trace システムプロシージャ

現在の呼び出しの場所につながるスタックトレースを返します。

構文

```
sa_stack_trace(
[ stack_frames
[, detail_level
[, connection_id ] ] ]
)
```

パラメータ

stack_frames

任意の CHAR(128) パラメータを使用して、次のいずれかを指定します。

'procedure'

最も外側にある文ではなく、プロシージャを返します。これはデフォルトの動作です。

'caller'

最も外側にある文 (クライアントから受け取った文) のみ返します。

'procedure+caller' または **'caller+procedure'**

すべての文を返します。

detail_level

任意の CHAR(128) パラメータを使用して、次のいずれかを指定します。

'stack'

プロシージャ名と行番号を含めます。これはデフォルトの動作です。

'stack+sql' または **'sql+stack'**

プロシージャ名と行番号に加えて、各レベルで実行される文の SQL テキストを含めます。

connection_id

任意の UNSIGNED INTEGER パラメータを使用して、指定された接続 ID に返される結果をフィルタします。指定しない場合、現在の接続に関する情報が返されます。

結果セット

カラム名	データ型	説明
StackLevel	UNSIGNED SMALLINT	現在の行にはスタックレベル 1 があります。

カラム名	データ型	説明
UserName	CHAR(128)	現在のレベルがバッチにある場合のプロシージャまたはトリガの所有者名、または NULL。
ProcName	CHAR(128)	呼び出しが実行されたプロシージャまたはトリガの名前、またはバッチタイプの名前。
LineNumber	UNSIGNED INTEGER	プロシージャ、トリガ、またはバッチ内の呼び出しの行番号。
SQLStatement	LONG VARCHAR	実行される文。

備考

結果の各レコードは、スタック上の単一の呼び出しを表します。複合文がプロシージャ、ファンクション、トリガ、イベントのいずれの一部でもない場合は、プロシージャ名ではなく、バッチのタイプ (watcom_batch または tsql_batch) が返されます。

この関数は、プロシージャの SYSPROCEDURE システムテーブルの proc_defn カラムで見つかった行番号を返します。これらの行番号は、プロシージャの作成に使用されるソース定義の行番号とは異なる可能性があります。

このプロシージャは、STACK_TRACE 関数と同じ情報を返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

この例は、sa_stack_trace システムプロシージャからの結果セットカラムを取得する方法を示しています。

```
SELECT StackLevel, UserName, ProcName, LineNumber FROM sa_stack_trace();
```

この文がストアドプロシージャのコンテキストの外で実行されると、結果セットは空になります。

次の例は、結果をクライアントウィンドウに送信する一般的なスタックトレースプロシージャの実装を示しています。

```
CREATE OR REPLACE PROCEDURE StackDump( MSG CHAR(128) )
BEGIN
    DECLARE myStackLevel UNSIGNED SMALLINT;
    DECLARE myUserName CHAR(128);
    DECLARE myProcName CHAR(128);
    DECLARE myLineNumber UNSIGNED SMALLINT;
    DECLARE mySQLStatement long varchar;
```

```
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    DECLARE myStack CURSOR FOR
        SELECT StackLevel, UserName, ProcName, LineNumber, SQLStatement FROM
sa_stack_trace('caller+procedure', 'stack+sql');
    MESSAGE 'Stack Trace: ' || MSG TO CLIENT;
    OPEN myStack;
    StackLoop: LOOP
        FETCH NEXT myStack
            INTO myStackLevel, myUserName, myProcName, myLineNumber,
mySQLStatement;
        IF SQLSTATE = err_notfound THEN
            LEAVE StackLoop;
        END IF;
        IF myStackLevel != 1 THEN
            MESSAGE myStackLevel - 1 || ' ' || myUserName || ' ' || myProcName ||
' ' || myLineNumber || ' ' || mySQLStatement
                TO CLIENT;
        ENDIF
    END LOOP StackLoop;
    CLOSE myStack;
END;

CREATE OR REPLACE PROCEDURE Proc1()
BEGIN
    CALL Proc2();
END;

CREATE OR REPLACE PROCEDURE Proc2()
BEGIN
    CALL Proc3();
END;

CREATE OR REPLACE PROCEDURE Proc3()
BEGIN
    CALL StackDump('Snapshot from Proc3');
END;

CALL Proc1();
```

結果:

```
CALL Proc1();
-- Stack Trace: Snapshot from Proc3
-- 1 DBA proc3 3 call StackDump('Snapshot from Proc3')
-- 2 DBA proc2 3 call Proc3()
-- 3 DBA proc1 3 call Proc2()
-- Procedure completed
```

関連情報

TRY 文 [1370 ページ]
BEGIN 文 [745 ページ]
ERROR_LINE 関数 [その他] [340 ページ]
ERROR_MESSAGE 関数 [その他] [342 ページ]
ERROR_PROCEDURE 関数 [その他] [343 ページ]
ERROR_SQLCODE 関数 [その他] [344 ページ]
ERROR_SQLSTATE 関数 [その他] [346 ページ]

ERROR_STACK_TRACE 関数 [その他] [347 ページ]
STACK_TRACE 関数 [その他] [539 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
SYSPROCEDURE システムビュー [1824 ページ]

1.6.8.93 sa_statement_text システムプロシージャ

各行に項目が 1 つずつ表示されるように、SELECT 文をフォーマットします。これは、すべての改行文字が削除されている要求ログから、長い文を表示する場合に便利です。

構文

```
sa_statement_text( txt )
```

パラメータ

txt

この LONG VARCHAR パラメータを使用して、SELECT 文を指定します。

結果セット

カラム名	データ型	説明
stmt_text	LONG VARCHAR	SELECT 文の句です。

備考

テキストは文字列として (一重引用符で囲んで)、または文字列式として入力します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の呼び出しは、各行に項目が 1 つずつ表示されるように、SELECT 文をフォーマットします。

```
CALL sa_statement_text( 'SELECT * FROM car WHERE name=''Audi''' );
```

	stmt_text
1	SELECT *
2	FROM car
3	WHERE name = 'Audi'

関連情報

sa_get_request_times システムプロシージャ [1520 ページ]
sa_get_request_profile システムプロシージャ [1518 ページ]

1.6.8.94 sa_table_fragmentation システムプロシージャ

データベーステーブルの断片化に関する情報をレポートします。

構文

```
sa_table_fragmentation(
[ tbl_name
[, owner_name ] ]
)
```

パラメータ

tbl_name

任意の CHAR(128) パラメータを使用して、断片化を調べるテーブルの名前を指定します。デフォルトは NULL です。

owner_name

任意の CHAR(128) パラメータを使用して、tbl_name の所有者を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
TableName	CHAR(128)	テーブルの名前。
rows	UNSIGNED INTEGER	テーブル内のローの数。
row_segments	UNSIGNED BIGINT	テーブル内のローセグメントの数。
segs_per_row	DOUBLE	ローあたりのセグメントの数。

備考

データベース管理者は、このプロシージャを使ってデータベースのテーブルの断片化に関する情報を取得できます。引数を指定しないと、データベースにあるすべてのテーブルの結果が返されます。

データベーステーブルの断片化が極端に進んだときは、REORGANIZE TABLE を実行するか、データベースを再構築して、テーブルの断片化を減らし、パフォーマンスを向上させることができます。

権限

MANAGE ANY STATISTICS または MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

```
CALL sa_table_fragmentation( 'Products','GROUPO' );
```

```
CALL sa_table_fragmentation( owner_name='GROUPO' );
```

関連情報

REORGANIZE TABLE 文 [1266 ページ]

1.6.8.95 sa_table_page_usage システムプロシージャ

データベーステーブルのページの使用状況に関する情報をレポートします。

構文

```
sa_table_page_usage( )
```

結果セット

カラム名	データ型	説明
TableId	UNSIGNED INTEGER	テーブル ID。
TablePages	INTEGER	テーブルで使用されるテーブルページの数。
PctUsedT	INTEGER	使用されているテーブルページ領域の割合。
IndexPages	INTEGER	テーブルで使用されるインデックスページの数。
PctUsedI	INTEGER	使用されているインデックスページ領域の割合。
PctOfFile	INTEGER	テーブルが使用しているデータベースファイル総数の割合。
TableName	CHAR(128)	テーブル名。

備考

結果には、情報ユーティリティで提供される情報と同じ内容が含まれています。progress_messages データベースオプションが Raw または Formatted に設定されている場合、このシステムプロシージャの実行中にデータベースサーバからクライアントに進行状況メッセージが送信されます。

権限

MANAGE ANY DBSPACE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、SalesOrderItems テーブルのページの使用状況に関する情報を取得します。

```
SELECT * FROM sa_table_page_usage( )
    WHERE TableName = 'SalesOrderItems';
```

1.6.8.96 sa_table_stats システムプロシージャ

各テーブルから読み込まれたページ数に関する情報をレポートします。

構文

sa_table_stats( )

結果セット

カラム名	データ型	説明
table_id	INTEGER	テーブル ID。
creator	CHAR(128)	テーブルの作成者のユーザ名。
table_name	CHAR(128)	テーブル名。
"count"	UNSIGNED BIGINT	SYSTAB から取得したテーブルの推定ロー数。
table_page_count	UNSIGNED BIGINT	テーブルで使用されるメインページの数。
table_page_cached	UNSIGNED BIGINT	キャッシュに現在格納されているベーステーブルページの数。インデックスページは含まれません。
table_page_reads	UNSIGNED BIGINT	メインテーブルのページに実行されるページ読み込み数。
ext_page_count	UNSIGNED BIGINT	このテーブルで使用される拡張ページの総数。拡張ページには、長い文字列または他の BLOB タイプの末尾が格納されます。
ext_page_cached	UNSIGNED BIGINT	キャッシュに現在格納されている拡張ページの数。
ext_page_reads	UNSIGNED BIGINT	これまでキャッシュに読み込まれた拡張ページの数。

備考

sa_table_stats プロシージャが返す各ローは、ページの統計情報がオプティマイザによって管理されているテーブルを記述しています。sa_table_stats プロシージャは、キャッシュメモリを使用しているテーブルや、各テーブルでディスクの読み込みが実行される回数を検索するときに使用されます。たとえば、sa_table_stats プロシージャを使用して、最も多いディスク読み込みを生成しているテーブルを検索できます。このプロシージャの結果は推定値であり、診断目的以外には使用しないでください。

table_page_cached カラムは、現在キャッシュに格納されているテーブルのページ数を示します。table_page_reads カラムは、オプティマイザがテーブルのカウントを管理し始めてからディスクから読み込まれたテーブルのページ数を示します。これらの統計情報は、データベースに永続的には格納されません。これは、テーブルが初めてメモリにロードされた後、テーブルに行われたアクティビティを表します。

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次のクエリを実行して、各テーブルから読み込むページ数に関する情報を実行します。

```
CALL sa_table_stats( );
```

関連情報

SYSTAB システムビュー [1842 ページ]

1.6.8.97 sa_text_index_stats システムプロシージャ

データベース内のテキストインデックスに関する統計情報を返します。

構文

```
sa_text_index_stats( )
```

備考

sa_text_index_stats システムプロシージャは、データベース内のテキストインデックスごとに統計情報を表示するときに使用します。次の表に、sa_text_index_stats から返される情報を示します。

カラム名	データ型	説明
owner_id	UNSIGNED INTEGER	テーブルの所有者の ID。
table_id	UNSIGNED INTEGER	テーブルの ID。
index_id	UNSIGNED INTEGER	テキストインデックスの ID。
text_config_id	UNSIGNED BIGINT	インデックスが参照するテキスト設定オブジェクトの ID。
owner_name	CHAR(128)	所有者の名前。
table_name	CHAR(128)	テーブルの名前。
index_name	CHAR(128)	テキストインデックスの名前。
text_config_name	CHAR(128)	テキスト設定オブジェクトの名前。
doc_count	UNSIGNED BIGINT	テキストインデックスに含まれるインデックス付けされたカラム値の総数。
doc_length	UNSIGNED BIGINT	テキストインデックス内のデータの長さの合計。
pending_length	UNSIGNED BIGINT	保留中の変更の合計長。
deleted_length	UNSIGNED BIGINT	保留中の削除の合計長。
last_refresh	TIMESTAMP	データベースサーバで最後に行ったリフレッシュのローカル日時。この値はシミュレートされたタイムゾーンの影響を受けます。

IMMEDIATE REFRESH テキストインデックスの場合、pending_length、deleted_length、last_refresh の値は 0 です。

MANUAL REFRESH テキストインデックスの場合、doc_length、pending_length、deleted_length を使用して、テキストインデックスを再表示するかどうかや、実行する再表示の種類 (再構築か増分か) を決定できます。

権限

次のシステム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY OBJECT
- DROP ANY OBJECT

関連する動作

なし。

例

次の文は、データベース内のテキストインデックスごとに統計情報を返します。

```
CALL sa_text_index_stats( );
```

関連情報

DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]
sa_text_index_vocab システムプロシージャ [1643 ページ]
SYSTEXTIDX システムビュー [1850 ページ]

1.6.8.98 sa_text_index_vocab システムプロシージャ

CHAR テキストインデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数のリストを返します。

構文

```
sa_text_index_vocab(
indexname
, tabname
[, tabowner ]
)
```

パラメータ

indexname

テキストインデックスの名前を指定する CHAR(128) パラメータです。

tabname

テキストインデックスを構築するテーブルの名前を指定する CHAR(128) パラメータです。

tabowner

テーブルの所有者を指定する任意の CHAR(128) パラメータです。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
term	VARCHAR(60)	テキストインデックスに含まれる単語。
freq	BIGINT	単語が含まれるインデックス値の数。

備考

sa_text_index_vocab システムプロシージャは、テキストインデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数を返します (一部のインデックス値に単語が複数回含まれる場合は、出現する総数より小さくなります)。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、インデックス付けされたテーブルに対する SELECT オブジェクトレベル権限、または SELECT ANY TABLE システム権限が必要です。

関連する動作

なし

例

次の例は、サンプルデータベース内の Products.Description カラムにテキストインデックス VocabTxtIdx を構築します。

```
CREATE TEXT INDEX VocabTxtIdx2 ON Products( Description );
```

その次の文は、sa_text_index_vocab システムプロシージャを実行し、テキストインデックスに含まれるすべての単語を返します。

```
SELECT * FROM sa_text_index_vocab( 'VocabTxtIdx2', 'Products', 'GROUPO' );
```

term	freq
Cap	2
Cloth	1
Cotton	2
Crew	1
Hooded	1

term	freq
neck	2
...	...

関連情報

sa_text_index_vocab_nchar システムプロシージャ [1645 ページ]
DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]
SYSTEXTIDX システムビュー [1850 ページ]

1.6.8.99 sa_text_index_vocab_nchar システムプロシージャ

NCHAR テキストインデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数のリストを返します。

構文

```
sa_text_index_vocab_nchar(
indexname
, tabname
[, tabowner ]
)
```

パラメータ

indexname

この CHAR(128) パラメータを使用して、テキストインデックスの名前を指定します。

tabname

この CHAR(128) パラメータを使用して、テキストインデックスを構築するテーブルの名前を指定します。

tabowner

任意の CHAR(128) パラメータを使用して、テーブルの所有者を指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
term	NVARCHAR(60)	テキストインデックスに含まれる単語です。
freq	BIGINT	単語が含まれるインデックス値の数です。

備考

sa_text_index_vocab_nchar システムプロシージャは、テキストインデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数を返します (一部のインデックス値に単語が複数回含まれる場合は、出現する総数より小さくなります)。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

さらに、インデックス付けされたテーブルに対する SELECT オブジェクトレベル権限、または SELECT ANY TABLE システム権限が必要です。

関連する動作

なし

例

次の例は、NProducts.Description カラムに VocabNTxtIdx と呼ばれるテキストインデックスを構築します。NProducts テーブルは、CHAR カラムではなく NCHAR カラムを持つ Products テーブルのバージョンです。

```
CREATE TEXT INDEX VocabNTxtIdx2 ON NProducts( Description );
```

その次の文は、sa_text_index_vocab_nchar システムプロシージャを実行し、テキストインデックスに含まれるすべての単語を返します。

```
SELECT * FROM sa_text_index_vocab_nchar( 'VocabNTxtIdx2', 'NProducts' );
```

term	freq
Cap	2
Cloth	1
Cotton	2

term	freq
Crew	1
Hooded	1
neck	2
...	...

関連情報

sa_text_index_vocab システムプロシージャ [1643 ページ]
DROP TEXT INDEX 文 [1077 ページ]
REFRESH TEXT INDEX 文 [1255 ページ]
TRUNCATE TEXT INDEX 文 [1369 ページ]
sa_refresh_text_indexes システムプロシージャ [1592 ページ]
SYSTEXTIDX システムビュー [1850 ページ]

1.6.8.100 sa_transactions システムプロシージャ

現在アクティブなトランザクションのリストを返します。

構文

sa_transactions( )

結果セット

カラム名	データ型	説明
connection_num	INTEGER	トランザクションが実行されている接続の接続 ID。
transaction_id	INTEGER	データベースサーバが追跡している間、トランザクションをユニークに識別する ID。ID は古いトランザクション情報を破棄するときに再利用されます。
start_time	TIMESTAMP	トランザクションを開始したときのデータベースサーバのローカル日時。この値はシミュレートされたタイムゾーンの影響を受けます。

カラム名	データ型	説明
start_sequence_num	UNSIGNED BIGINT	トランザクションの開始シーケンス番号。
end_sequence_num	UNSIGNED BIGINT	トランザクションがコミットまたはロールバックされた場合はトランザクションの終了シーケンス番号。それ以外の場合は NULL です。
committed	BIT	トランザクションのステータス。トランザクションが COMMIT で終了した場合は true、ROLLBACK で終了した場合は false、トランザクションがまだアクティブな場合は NULL です。
version_entries	UNSIGNED INTEGER	トランザクションが保存したローバージョンの回数。

備考

このプロシージャは、現在データベースで実行されているトランザクションに関する情報を提供します。

権限

MONITOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次のクエリを実行して、現在アクティブなトランザクションを識別できます。

```
CALL sa_transactions( );
```

関連情報

sa_snapshots システムプロシージャ [1629 ページ]

1.6.8.101 sa_unload_cost_model システムプロシージャ

現在のコストモデルを特定のファイルにアンロードします。

構文

```
sa_unload_cost_model( file_name )
```

パラメータ

file_name

この CHAR(256) パラメータを使用して、データのアンロード先となるファイルの名前を指定します。システムプロシージャを実行するのはデータベースサーバであるため、`file_name` はデータベースサーバコンピュータ上のファイルを指定します。また、相対的な `file_name` の場合、データベースサーバの開始ディレクトリに相対的なファイルを指定します。

備考

オプティマイザは、コストモデルを使用して、クエリの最適なアクセスプランを決定します。データベースサーバは、各データベースのコストモデルを管理します。データベースのコストモデルは、ALTER DATABASE 文の CALIBRATE SERVER 句を使用して、任意のタイミングで再調整できます。たとえば、データベースを非標準ハードウェアに移動する場合、コストモデルを再調整できます。

sa_unload_cost_model システムプロシージャを使用すると、コストモデルを ASCII ファイルに保存できます (`file_name`)。次に、別のデータベースにログインして、sa_load_cost_model システムプロシージャを使用して、最初のデータベースから 2 番目のデータベースへとコストモデルをロードできます。これによって、2 番目のデータベースを再調整する必要はなくなります。

注記

sa_unload_cost_model システムプロシージャは、ファイルに CALIBRATE PARALLEL READ 情報を含めません。

大量に類似ハードウェアのインストールがある場合、sa_unload_cost_model システムプロシージャを使用すると、繰り返しの時間がかかる再調整動作を省略できます。

ファイルを作成するには書き込みのパーミッションが必要です。

権限

SELECT ANY TABLE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、costmodel8 というファイルにコストモデルをアンロードします。

```
CALL sa_unload_cost_model( 'costmodel8' );
```

関連情報

ALTER DATABASE 文 [630 ページ]
sa_load_cost_model システムプロシージャ [1544 ページ]

1.6.8.102 sa_user_defined_counter_add システムプロシージャ

指定した量でユーザ定義カウンタの値を調整します。

構文

```
sa_user_defined_counter_add(
counter_name
[, delta
[, apply_to_con
[, appy_to_db
[, apply_to_server ] ] ] ]
)
```

パラメータ

counter_name

この VARCHAR(128) パラメータを使用して、値を変更するユーザ定義カウンタの名前を指定します。ユーザ定義カウンタ名の例としては、UserDefinedCounterRate01 や UserDefinedCounterRaw01 があります。

delta

この BIGINT パラメータを使用して、ユーザ定義カウンタを増分または減分する量を指定します。デフォルトは 1 です。

apply_to_con

この INTGER パラメータを使用して、現在の接続のカウンタ値を調整するかどうかを指定します。0 は値を調整しないことを意味し、1 は値を調整することを意味します。デフォルトは 1 です。

apply_to_db

この INTGER パラメータを使用して、データベースのカウンタ値を調整するかどうかを指定します。0 は値を調整しないことを意味し、1 は値を調整することを意味します。デフォルトは 1 です。

apply_to_server

この INT パラメータを使用して、データベースサーバのカウンタ値を調整するかどうかを指定します。0 は値を調整しないことを意味し、1 は値を調整することを意味します。デフォルトは 1 です。

戻り値

この関数は INTEGER ステータスコードを返します。

備考

この関数は、`delta` が定義されている場合に 1 を、`delta` が定義されていない場合に 0 を返し、エラーが発生した場合はエラーコードを返します。次のようなエラーがあります。

- カウンタ名が無効です。
- apply_to_server、apply_to_db、または apply_to_con parameter の値が無効です。

カウンタへの同時アクセスは自動的に適用されるので、カウンタ値は複数の同時要求によって増分される可能性があります。

ユーザ定義カウンタは 32 ビットの UNSIGNED INTEGER 値として実装されます。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、現在の接続、データベース、データベースサーバ用の UserDefinedCounterRate01 の値を 2 ずつ増分します。

```
SELECT sa_user_defined_counter_add( 'UserDefinedCounterRate01', 2, 1, 1, 1 );
```

関連情報

sa_user_defined_counter_set システムプロシージャ [1652 ページ]

1.6.8.103 sa_user_defined_counter_set システムプロシージャ

指定した値にユーザ定義カウンタを設定します。

構文

```
sa_user_defined_counter_set(
counter_name
, value
[, apply_to_con
[, appy_to_db
[, apply_to_server ] ] ]
```

パラメータ

counter_name

この VARCHAR(128) パラメータを使用して、値を変更するユーザ定義カウンタの名前を指定します。ユーザ定義カウンタ名の例としては、UserDefinedCounterRate01 や UserDefinedCounterRaw01 があります。

value

この BIGINT パラメータを使用して、ユーザ定義カウンタに設定する値を指定します。

apply_to_con

この INT パラメータを使用して、現在の接続のカウンタ値を調整するかどうかを指定します。0 は値を調整しないことを意味し、1 は値を調整することを意味します。デフォルトは 1 です。

apply_to_db

この INT パラメータを使用して、データベースのカウンタ値を調整するかどうかを指定します。0 は値を調整しないことを意味し、1 は値を調整することを意味します。デフォルトは 0 です。

apply_to_server

この INT パラメータを使用して、データベースサーバのカウンタ値を調整するかどうかを指定します。0 は値を調整しないことを意味し、1 は値を調整することを意味します。デフォルトは 0 です。

戻り値

この関数は INTEGER ステータスコードを返します。

備考

この関数は、`value` が定義されている場合に 1 を、`value` が定義されていない場合に 0 を返し、エラーが発生した場合はエラーコードを返します。次のようなエラーがあります。

- カウンタ名が無効です。
- apply_to_server、apply_to_db、または apply_to_con parameter の値が無効です。

カウンタへの同時アクセスは自動的に適用されるので、カウンタ値は複数の同時要求によってリセットされる可能性があります。

ユーザ定義カウンタは 32 ビットの UNSIGNED INTEGER 値として実装されます。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、現在の接続、データベース、データベースサーバの UserDefinedCounterRate01 の値を 0 に設定します。

```
SELECT sa_user_defined_counter_set( 'UserDefinedCounterRate01', 0, 1, 1, 1 );
```

関連情報

sa_user_defined_counter_add システムプロシージャ [1650 ページ]

1.6.8.104 sa_validate システムプロシージャ

データベースの全部または一部を検証します。

構文

```
sa_validate(
[ tbl_name
[, owner_name ] ]
[, check_type ] ]
```

```
[, isolation_type ] ]
)
```

パラメータ

tbl_name

任意の CHAR(128) パラメータを使用して、検証するテーブルまたはマテリアライズドビューの名前を指定します。デフォルトは NULL です。この場合、データベース全体が検証されます。

owner_name

任意の CHAR(128) パラメータを使用して、所有者を指定します。所有者のみを指定すると、この所有者が所有するすべてのテーブルとマテリアライズドビューが検証されます。デフォルトは NULL です。

check_type

オプションの CHAR(10) パラメータを使用して、実行する検証のタイプを指定します。値は次のいずれかです。

EXPRESS

このパラメータが EXPRESS の場合、各テーブルは WITH EXPRESS CHECK 句を含む VALIDATE TABLE 文を使用して検証されます。

NULL このパラメータが NULL (デフォルト) の場合、各テーブルは VALIDATE TABLE 文を使用して検証されます。

isolation_type

アクティブなトランザクションを持つテーブルを検証する場合は、破損したテーブルに関する誤ったエラーを受信することを避けるため、このオプションを使用します。値は次のいずれかです。

DATA LOCK 指定したテーブルに排他データロックを適用することで、トランザクションによってテーブルスキーマまたはデータが修正されることを回避します。同時実行トランザクションは、テーブルデータまたはスキーマを読み込むことができますが、修正することはできません。

SNAPSHOT スナップショットアイソレーションを適用して、コミットされたデータのみがチェックされるようにします。トランザクションは、データを読み込むことも修正することもできます。この句を使用するには、データベースで (allow_snapshot_isolation データベースオプションを使用して) スナップショットアイソレーションが有効になっている必要があります。この句ではスナップショットアイソレーションを使用するため、パフォーマンスに影響することがよくあります。

結果セット

カラム名	データ型	説明
Messages	CHAR(128)	エラーなく検証が成功した場合は、カラムに `No error detected` が格納されます。
IsValid	BIT	有効な場合は 1、検証エラーが検出された場合は 0

カラム名	データ型	説明
ObjectName	CHAR(261)	• 有効な場合は空の文字列 • データベースの検証が失敗した場合は Database • テーブルの検証が失敗した場合はテーブルの所有者と名称

備考

指定される引数	検証タイプ
なし。	データベース内のすべてのテーブル、マテリアライズドビュー、インデックスが検証されます (各テーブルの VALIDATE TABLE と同じ)。データベース自体も検証されます (VALIDATE DATABASE 文と同じ)。チェックサムの検証も含まれます。
tbl_name	owner_name が NULL の場合、tbl_name と一致するすべてのテーブル、そのマテリアライズドビュー、そのインデックスが検証されます。
owner_name	指定されたユーザが所有するすべてのテーブル、マテリアライズドビュー、インデックスが検証されます。
tbl_name および owner_name	指定されたユーザにより所有される指定されたテーブルまたはマテリアライズドビューと、そのインデックスすべてが検証されます。
check_type	指定されたテーブル、マテリアライズドビュー、またはインデックスが、WITH EXPRESS CHECK を使用して検証されます。チェックサムの検証を含め、データベース自体も検証されます。
isolation_type	指定されたテーブル、マテリアライズドビュー、またはインデックスが検証されます。テーブルには、排他データロックが適用されているか、コミットされたデータのみが評価されます。チェックサムの検証を含め、データベース自体も検証されます。

警告

isolation_type を指定しない場合、データベースを変更している接続がない状態のときにのみ検証を実行してください。それ以外の場合は、データベースの何らかの破損を示す誤ったエラーが報告される場合があります。

グローバルチェックサムまたは書き込みチェックサムを使用するデータベースのディスクページを検証できます。データベースでグローバルチェックサムが有効になっている場合は、すべてのデータベースページが検証されます。データベースで書き込みチェックサムのみが使用されている場合は、チェックサムを含むページのみが検証されます。

チェックサムを有効にしたデータベースの場合、チェックサムは各データベースページで計算され、ページがディスクに書き込まれる際にその値が格納されます。検証ユーティリティ (dbvalid)、VALIDATE 文、sa_validate システムプロシージャ、または SQL Central のデータベース検証ウィザードを使用してチェックサムの検証を実行できます。この検証では、ディスクからのデータベースページの読み込みとそのページに対するチェックサムの計算が行われます。計算されたチェックサムが格納されているページのチェックサムと一致しない場合、ディスク上で、または書き込み中に、そのページが変更または破壊されています。1 つまたは複数のページが破壊されている場合、エラーが返されて無効なページに関する情報がデータベースサーバメッセージウィンドウに表示されます。

権限

VALIDATE ANY OBJECT システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

`isolation type` が DATA LOCK の場合、排他データロックが指定されたテーブルまたはビューに適用されます。

両方の `isolation_type` オプションのオートコミット。

例

次の文は、ユーザ pjones が所有するテーブルとマテリアライズドビューの検証を実行します。

```
CALL sa_validate( owner_name = 'pjones' );
```

関連情報

VALIDATE 文 [1400 ページ]

1.6.8.105 sa_verify_password システムプロシージャ

現在のユーザのパスワードを検証します。

構文

```
sa_verify_password( curr_pswd )
```

パラメータ

curr_pswd

この CHAR(128) パラメータを使用して、現在のデータベースユーザのパスワードを指定します。

戻り値

INTEGER 値を返す関数。

備考

このプロシージャは sp_password で使用されます。パスワードが一致した場合は 0 が返され、エラーは発生しません。パスワードが一致しなかった場合、エラーと診断されます。パスワードが一致しない場合は、接続が終了しません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例では、現在のユーザが DBA または User1 の場合に現在の接続のパスワードの検証を試行します。現在のパスワードが一致しない場合は、エラーが発生します。

```
IF USER_NAME() = 'DBA' THEN
    SELECT sa_verify_password( 'sql' );
ELSEIF USER_NAME() = 'User1' THEN
    SELECT sa_verify_password( 'user' );
END IF;
```

関連情報

Adaptive Server Enterprise システムプロシージャ [1437 ページ]

1.6.8.106 sp_alter_secure_feature_key システムプロシージャ

認証コード、機能リスト、またはその両方を変更して、以前に定義されたセキュリティ保護済み機能キーを変更します。

構文

```
sp_alter_secure_feature_key(
name
, auth_key
, features
)
```

パラメータ

name

変更するセキュリティ保護済み機能キーの VARCHAR (128) 名。指定した名前のキーが既存である必要があります。

auth_key

セキュリティ保護済み機能キーの、大文字と小文字を区別する新しい CHAR (128) 認証コード。認証コードは、6 文字以上の空でない文字列か、または既存の認証コードが変更されないことを示す NULL である必要があります。

features

キーで有効にできる機能のカンマ区切りのリストである LONG VARCHAR。`features` が NULL の場合、既存の機能セットは変更されません。

備考

このプロシージャを実行するには、接続で MANAGE_KEYS 機能を有効にする必要があります。

このプロシージャを使用して、認証コードまたは既存のセキュリティ保護済み機能キーの機能リストを変更できます。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例が機能するには、サーバが次のオプションで起動されていることが必要です。`-sk securefkey`。

この例では、MANAGE_KEYS を含む SYSTEM セキュリティ保護済み機能キーの有効化、大文字と小文字が区別される認証キー SecureMySet と新しいセキュリティ保護済み機能キー MYSET の作成、セットに属するセキュリティ保護済み機能の変更を行い、sp_list_secure_feature_keys を使用して現在定義されているセキュリティ保護済み機能キーのリストを取得できます。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'myset', 'SecureMySet', 'local' );
CALL sp_alter_secure_feature_key( 'myset', NULL, 'local,remote' );
CALL sp_list_secure_feature_keys( );
```

関連情報

sp_create_secure_feature_key システムプロシージャ [1664 ページ]
sp_drop_secure_feature_key システムプロシージャ [1675 ページ]
sp_list_secure_feature_key システムプロシージャ [1693 ページ]
sp_use_secure_feature_key システムプロシージャ [1751 ページ]

1.6.8.107 sp_auth_sys_role_info システムプロシージャ

以前のバージョンの SQL Anywhere から、それに対応する互換ロールへの、権限のマッピングを返します。

構文

```
sp_auth_sys_role_info( )
```

結果セット

カラム	タイプ	説明
auth	VARCHAR(20)	権限の名前。
role_name	CHAR(128)	対応する互換ロールの名前。
role_id	UNSIGNED INTEGER	互換ロールの ID 番号。

備考

なし

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、16.0 より前のバージョンの SQL Anywhere からの権限 (auth) のリストを返します。これらの権限は、バージョン 16.0 で提供される対応する互換ロール (role_name) にマップされます。

```
CALL sp_auth_sys_role_info();
```

auth	role_name	role_id
DBA	SYS_AUTH_DBA_ROLE	2147485648
RESOURCE	SYS_AUTH_RESOURCE_ROLE	2147485649
BACKUP	SYS_AUTH_BACKUP_ROLE	2147485650
VALIDATE	SYS_AUTH_VALIDATE_ROLE	2147485651
READFILE	SYS_AUTH_READFILE_ROLE	2147485652
PROFILE	SYS_AUTH_PROFILE_ROLE	2147485653
READCLIENTFILE	SYS_AUTH_READCLIENTFILE_ROLE	2147485654
WRITECLIENTFILE	SYS_AUTH_WRITECLIENTFILE_ROLE	2147485655
WRITEFILE	SYS_AUTH_WRITEFILE_ROLE	2147485656
REMOTE DBA	SYS_RUN_REPLICATION_ROLE	2147485664

1.6.8.108 sp_copy_directory システムプロシージャ

ディレクトリを指定された場所にコピーします。

構文

```
sp_copy_directory(
source_path
, destination_path
)
```

パラメータ

source_path

この LONG NVARCHAR パラメータを使用して、コピーするディレクトリのパスを指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスはコンピュータ上で、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

destination_path

この LONG NVARCHAR パラメータを使用して、ディレクトリのコピー先のパスを指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスはコンピュータ上で、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。ディレクトリは存在しない場合には作成されます。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数は、ソースディレクトリのディレクトリとファイルを指定されたディレクトリにコピーします。ディレクトリとファイルは、ソースディレクトリに残ります。ソースディレクトリを削除する場合は、sp_delete_directory システムプロシージャを使用します。

権限

READ FILE および WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、SQLAnywhere.bkp の SQLAnywhere ディレクトリにあるサブディレクトリとファイルのコピーを作成します。

```
SELECT sp_copy_directory('c:¥¥sqlany¥¥samples¥¥SQLAnywhere', 'c:¥¥sqlany¥¥samples¥
¥SQLAnywhere.bkp');
```

サブディレクトリとファイルを含むディレクトリ全体が複製されます。

関連情報

sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.109 sp_copy_file システムプロシージャ

ファイルを指定された場所にコピーします。

構文

```
sp_copy_file(
source_path
, destination_path
)
```

パラメータ

source_path

この LONG NVARCHAR パラメータを使用して、移動するファイルのパスをファイル名を含めて指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスはコンピュータ上で、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

destination_path

この LONG NVARCHAR パラメータを使用して、ファイルの新しいディレクトリを指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスはコンピュータ上で、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数は、ディレクトリ間でファイルをコピーします。ソースディレクトリのファイルはそのまま存続します。ソースディレクトリのファイルを削除する場合は、sp_delete_file システムプロシージャを使用します。

ディスクサンドボックス機能が有効な場合、ソースパスとターゲットパスにアクセスできる必要があります。

権限

READ FILE および WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、ファイル `license.txt` を `c:¥temp` ディレクトリにコピーして、新しい名前を付けます。

```
SELECT sp_copy_file('c:¥¥sqlany¥¥license.txt', 'c:¥¥temp¥¥license.bkp');
```

`license.txt` ファイルの重複コピーは、別の名前で `temp` ディレクトリに存在します。

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.110 sp_create_directory システムプロシージャ

コンピュータ上にディレクトリを作成します。

構文

```
sp_create_directory( root_path )
```

パラメータ

root_path

この LONG NVARCHAR パラメータを使用して、作成するディレクトリのパスを指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数により、新しいディレクトリが作成されます。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、`c:¥sqlany¥samples` ディレクトリにデータベースファイル `SQLAnywhere.bkp` を作成します。

```
SELECT sp_create_directory( 'c:¥¥sqlany¥¥samples¥¥SQLAnywhere.bkp' );
```

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.111 sp_create_secure_feature_key システムプロシージャ

新しいセキュリティ保護済み機能キーを作成します。

構文

```
sp_create_secure_feature_key(
name
, auth_key
```

```
, features
)
```

パラメータ

name

新しいセキュリティ保護済み機能キーの VARCHAR (128) 名。この引数は、NULL または空白文字列にできません。

auth_key

セキュリティ保護済み機能キーの、大文字と小文字を区別する CHAR (128) 認証キー。認証コードは、6 文字以上の空でない文字列にする必要があります。

features

新しいキーで有効にできる機能のカンマ区切りのリストである LONG VARCHAR。

機能の前に - を指定すると、セキュリティ保護済み機能キーが取得されるとき、機能が再度有効になりません。

備考

このシステムプロシージャを実行するには、接続で MANAGE_KEYS 機能を有効にする必要があります。

このプロシージャにより、どのユーザにも提供される新しいセキュリティ保護済み機能が作成されます。SYSTEM セキュリティ保護済み機能キーは、-sk データベースサーバオプションを使用して作成されます。

データベースサーバごとのセキュリティ機能キーは 1000 個に制限されています。

キーは、6 文字以上の空でない文字列にする必要があり、二重引用符、制御文字 (0x20 未満のすべての文字)、またはバックスラッシュを含めることはできません。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例が機能するには、サーバが次のオプションで起動されていることが必要です。`-sk securefkey`。

この例では、MANAGE_KEYS を含むセキュリティ保護済み SYSTEM 機能キーを有効にして、大文字と小文字が区別される認証コード SecureMySet と新しいセキュリティ保護済み機能キー MYSET の作成を行い、sp_list_secure_feature_keys を使用して、現在定義されているセキュリティ保護済み機能キーのリストを取得できます。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'myset', 'SecureMySet', 'local' );
CALL sp_list_secure_feature_keys( );
```

この例は、MANAGE_KEYS を含むセキュリティ保護済み SYSTEM 機能キーを有効にして、次に大文字と小文字が区別される認証コード SecureMySet2 と新しいセキュリティ保護済み機能キー MYSET2 の作成を行います。database 機能は機能セットから除外されます。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'myset2', 'SecureMySet2', 'local,-database' );
```

関連情報

sp_alter_secure_feature_key システムプロシージャ [1657 ページ]
sp_drop_secure_feature_key システムプロシージャ [1675 ページ]
sp_list_secure_feature_key システムプロシージャ [1693 ページ]
sp_use_secure_feature_key システムプロシージャ [1751 ページ]

1.6.8.112 sp_db_cache_contents システムプロシージャ

現在キャッシュにあるページを示すビットマップとともに、DB 領域ごとにローを 1 つ返します。

構文

```
sp_db_cache_contents( [ dbspace_id ] )
```

パラメータ

dbspace_id 単一の DB 領域を指定するには、この SMALLINT パラメータを使用します。dbspace_id が NULL (デフォルト) の場合、現在のデータベースに属する DB 領域ごとにローが 1 つ返されます。それ以外の場合、指定された DB 領域のローが 1 つ返されます。

備考

このプロシージャは、LONG VARBIT オブジェクトを持つ DB 領域ごとにローが 1 つずつ存在する結果セットを返します (現在キャッシュにある DB 領域のページごとに 1 ビット)。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。SERVER OPERATOR システム権限または MONITOR システム権限が必要です。

関連する動作

なし

例

1. 次の文を実行し、DB 領域を作成します。

```
CREATE DBSPACE mydbs
AS 'C:¥¥mydb¥¥mydbs.db';
```

2. 新しい DB 領域にテーブルを作成し、次の文を実行してデータを追加します。

```
CREATE TABLE mytable( col1 INT, col2 CHAR(128) ) IN mydbs;
```

```
INSERT INTO mytable
    SELECT column_id, column_name
    FROM sys.syscolumn;
```

```
COMMIT;
```

3. 次の文を実行し、新しい DB 領域の ID を調べます。

```
SELECT * FROM SYS.SYSDBSPACE WHERE dbspace_name = 'mydbs';
```

   新しい DB 領域の ID を書き留め、saved_cache_pages カラムが NULL であることに注目します。

4. 次の文を実行することで、新しい DB 領域のどのページが現在キャッシュにあるかを確認します。

```
SELECT * FROM dbo.sp_db_cache_contents( );
```

   上の文は、データベース内のすべての DB 領域の情報を返します。結果を新しい mydbs DB 領域に制限するには、次の文を実行します。ここで、`mydbs-ID` は SYS から取得された新しい mydbs DB 領域の ID です。SYSDBSPACE:

```
SELECT * FROM dbo.sp_db_cache_contents( mydbs-ID );
```

5. 次の文を実行することで、データベースの現在の安定した状態を保存します。

```
ALTER DATABASE SAVE CACHE;
```

6. その後、他の多くのトランザクションが実行され、キャッシュに大幅な変更が加えられた後、保存された安定した状態に戻すことができます。すべての DB 領域の安定した状態のページをすべてキャッシュに読み込むことも、特定の DB 領域の安定した状態のページをすべてキャッシュに読み込むこともできます。
   すべての DB 領域の安定した状態のページをすべてキャッシュに読み込むには、次の文を実行します。

   ```
   ALTER DATABASE RESTORE CACHE;
   ```

   ある DB 領域 (新しい DB 領域 mydbs など) の安定した状態のページをすべて読み込む場合は、次のような一連の文を実行します。

   ```
   CREATE VARIABLE cache_pages LONG VARBIT;
   ```

   ```
   SELECT saved_cache_pages
       INTO cache_pages
       FROM SYS.SYSDBSPACE
       WHERE dbspace_name='mydbs';
   ```

   ```
   CALL dbo.sp_read_db_pages( mydbs-ID, cache_pages );
   ```

関連情報

sp_read_db_pages システムプロシージャ [1712 ページ]

1.6.8.113 sp_delete_directory システムプロシージャ

指定されたディレクトリを削除します。

構文

```
sp_delete_directory( root_path )
```

パラメータ

root_path

この LONG NVARCHAR パラメータを使用して、削除するディレクトリのパスを指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数は、指定した場所からディレクトリを削除します。

権限

READ FILE および WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、ディレクトリ `SQLAnywhere.bkp` を削除します。

```
SELECT sp_delete_directory('c:¥¥sqlany¥¥samples¥¥SQLAnywhere.bkp');
```

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.114 sp_delete_file システムプロシージャ

指定されたファイルをコンピュータから削除します。

構文

```
sp_delete_file( file_path )
```

パラメータ

file_path

この LONG NVARCHAR パラメータを使用して、削除するファイルのパスをファイル名を含めて指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数は、指定されたファイルをコンピュータから削除します。

このプロシージャを実行する場合は、ファイルがあるコンピュータ上のデータベースサーバに接続している必要があります。

権限

READ FILE および WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、temp ディレクトリから license.bkp ファイルを削除します。

```
SELECT sp_delete_file('c:¥¥temp¥¥license.bkp');
```

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]

ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.115 sp_disk_info システムプロシージャ

特定のパスのディスクドライブ情報を返します。

構文

```
sp_disk_info( path )
```

パラメータ

path

ディスクドライブ情報を返すファイルまたはディレクトリのパス (名前を含む) を指定するには、この LONG VARCHAR パラメータを返します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、現在のデータベースの場所に対する相対パスです。このパラメータのデフォルトは NULL です。

結果セット

カラム名	データ型	説明
free_space	UNSIGNED BIGINT	ディスクの空き領域 (バイト単位)。
total_space	UNSIGNED BIGINT	ディスクの合計領域 (バイト単位)。

備考

パスパラメータを指定しなかった場合、NULL を指定した場合、または空の文字列を指定した場合、sp_disk_info システムプロシージャは現在のデータベースの場所のディスク情報を返します。

ユーザがディスク情報にアクセスできないようにするには、sp_disk_info セキュリティ機能を有効にします。

権限

ACCESS DISK INFORMATION システム権限が必要です。

関連する動作

なし

例

myfile.txt ファイルに使用可能なディスクの空き領域と合計領域 (バイト単位) を返すには、次の文を実行します。

```
SELECT * FROM sp_disk_info( 'myfile.txt' );
```

次のような結果が返されます。

free_space	total_space
83388907520	242042793984

関連情報

ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.116 sp_displayroles システムプロシージャ

指定されたシステム権限、システムロール、ユーザ定義ロール、またはユーザ名に付与されたすべてのロールを返すか、またはロールの階層ツリー全体を表示します。

構文

```
sp_displayroles(
user_role_name
, display_mode
, grant_type
)
```

パラメータ

user_role_name

この CHAR(128) パラメータを使用して、システム権限、システムロール、ユーザ定義ロールの名前、またはユーザ名を指定します。指定されていない場合または NULL の場合、デフォルトで現在のユーザが使用されます。

display_mode

この VARCHAR(30) パラメータを使用して、user_role_name に対する親レベルまたは子レベルを指定します。display_mode が指定されていない場合、または NULL の場合、明示的に付与されたロールと権限 (継承されたロールまたは権限) のみが返されます。display_mode で使用できる値には次のものがあります。

expand_up

user_role_name の親階層ツリーの user_role_name に付与されているシステムロールを表示します。

expand_down

user_role_name の子レベルのロール階層ツリーを含む、user_role_name に付与されているシステムロールと権限を表示します。

grant_type

この VARCHAR(30) パラメータを使用して、返される付与のタイプを制御します。指定されない場合、デフォルトで ALL が使用されます。grant_type で使用できる値には次のものがあります。

no_admin

WITH NO ADMIN OPTION 句または WITH ADMIN OPTION 句とともに user_role_name に付与されているすべてのロールとシステム権限を表示します。

admin

WITH NO ADMIN OPTION 句または WITH ADMIN ONLY OPTION 句とともに user_role_name に付与されているすべてのロールとシステム権限を返します。

all

user_role_name に付与されているすべてのロールと権限を表示します。

結果セット

カラム名	データ型	説明
role_name	CHAR(128)	user_role_name に付与されているロールまたはシステム権限。
parent_role_name	CHAR(128)	user_role_name の親のロール名。
grant_type	CHAR(10)	user_role_name に管理権限があるかどうかに関する情報。考えられる値は、次のとおりです。NO ADMIN、ADMIN、または ADMIN ONLY。
role_level	SMALLINT	expand_down モードの場合直接付与されるロールの場合はレベル 1、その下のレベルに付与されるロールのレベルは 2、以下同様です。 expand_up モードの場合 user_role_name が付与されているロールのレベルは 0、1 つ上の階層のレベルは -1、以下同様です。

備考

システム権限については、システム権限ロール名ではなくシステム権限名が結果に表示されます。expand_down モードでは、レベル 1 (直接付与されたロール) で parent_role_name が NULL となります。デフォルトモードでは、直接付与されたロールのみが表示されるため、role_level カラムが 1 で parent_role_name が NULL となります。

このプロシージャを expand_up モードのユーザで使用した場合、そのユーザはどのロール階層でもトップレベルにないため、結果が返されません。同様に、このプロシージャを expand_down モードの不変のシステム権限で使用した場合、不変のシステム権限はどのロール階層でも最下位にあるため、結果が返されません。デフォルトモードでは、直接付与されたロールとシステム権限のみが表示されます。

標準

ANSI/ISO SQL 標準

標準になし。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

別のユーザ ID またはロールのシステム権限またはロールを返すには、MANAGE ROLES システム権限が必要です。

関連する動作

なし

例

次の文は、コマンドを発行したユーザに付与されているすべてのロールを返します。

```
CALL sp_displayroles();
```

この例では、SYS_SPATIAL_ADMIN_ROLE システムロールに付与されているシステム権限のリストが返されます。

```
CALL sp_displayroles( 'SYS_SPATIAL_ADMIN_ROLE' );
```

role_name	parent_role_name	grant_type	role_level
PUBLIC	(NULL)	NO ADMIN	1
MANAGE ANY SPATIAL OBJECT	(NULL)	NO ADMIN	1

この例では、SYS_SPATIAL_ADMIN_ROLE に付与されているシステム権限のリストが、ロール階層でその上にあるすべてのロールも含めて返されます。

```
CALL sp_displayroles( 'SYS_SPATIAL_ADMIN_ROLE', 'expand_up');
```

role_name	parent_role_name	grant_type	role_level
SYS_AUTH_DBA_ROLE	dbo	ADMIN	-3
SYS_AUTH_DBA_ROLE	SYS_RUN_REPLICATION_ROLE	ADMIN	-3
SYS_AUTH_SSO_ROLE	SYS_AUTH_DBA_ROLE	ADMIN	-2
MANAGE ROLES	SYS_AUTH_SSO_ROLE	ADMIN	-1
MANAGE ROLES	SYS_REPLICATION_ADMIN_ROLE	NO ADMIN	-1
SYS_SPATIAL_ADMIN_ROLE	MANAGE ROLES	ADMIN ONLY	0

次の文は、ユーザ User1 に付与されているすべてのシステム権限を返します。

```
CALL sp_displayroles( 'User1' );
```

関連情報

sp_sys_priv_role_info システムプロシージャ [1734 ページ]
sp_has_role システムプロシージャ [1685 ページ]
sp_proc_priv システムプロシージャ [1705 ページ]
sp_objectpermission システムプロシージャ [1698 ページ]

1.6.8.117 sp_drop_secure_feature_key システムプロシージャ

セキュリティ保護済み機能キーを削除します。

構文

```
sp_drop_secure_feature_key( name )
```

パラメータ

name

削除するセキュリティ保護済み機能キーの VARCHAR (128) 名。

備考

このプロシージャを実行するには、接続で MANAGE_KEYS 機能を有効にする必要があります。

この名前の付いたキーが存在しない場合は、エラーが返されます。この名前の付いたキーが存在する場合、それが機能とセキュリティ保護済み機能キーの管理が可能な最後のセキュリティ機能キーでないかぎり、削除されます。SYSTEM セキュリティ保護済み機能キーは、MANAGE_FEATURES と MANAGE_KEYS 機能が有効な別のキーがないかぎり、削除できません。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例が機能するには、サーバが次のオプションで起動されていることが必要です。`-sk securefkey`。

この例では、MANAGE_KEYS を含む SYSTEM セキュリティ保護済み機能キーを有効にして、MYSET というセキュリティ保護済み機能キーを削除します。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_drop_secure_feature_key( 'myset' );
```

関連情報

sp_alter_secure_feature_key システムプロシージャ [1657 ページ]
sp_create_secure_feature_key システムプロシージャ [1664 ページ]
sp_list_secure_feature_key システムプロシージャ [1693 ページ]
sp_use_secure_feature_key システムプロシージャ [1751 ページ]

1.6.8.118 sp_find_top_statements システムプロシージャ

ログに記録された文とプランの各組み合わせに対するパフォーマンス統計をレポートします。

構文

```
sp_find_top_statements( stmt_text, stmt_hash )
```

パラメータ

stmt_text

任意の LONG VARCHAR パラメータを使用して、SQL 文の文字列を指定します。デフォルトは NULL です。

stmt_hash 任意の UNSIGNED BIGINT パラメータを使用して、ステートメントハッシュを指定します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
stmt_hash	UNSIGNED BIGINT	文の識別子を返します。
owner_name	CHAR(128)	ストアドプロシージャの所有者名を返します。文がストアドプロシージャ、ユーザ定義の機能、トリガ、またはイベントの一部ではない場合、値は NULL になります。文がストアドプロシージャの一部として実行され、のちにストアドプロシージャが削除された場合も、値は NULL となります。ストアドプロシージャの一部として実行された文は、文のテキストではなくプロシージャに関連するハッシュを取得します。
proc_name	CHAR(128)	文が属しているプロシージャの名前を返します。文がストアドプロシージャ、ユーザ定義の機能、トリガ、またはイベントの一部ではない場合、値は NULL になります。文がストアドプロシージャの一部として除外され、ストアドプロシージャが削除された場合も、値は NULL となります。ストアドプロシージャの一部として実行された文は、文のテキストではなくプロシージャに関連するハッシュを取得します。

カラム名	データ型	説明
reusable_stmt_id	UNSIGNED INTEGER	プロシージャ内の文に割り当てられた一意の識別子を返します (行番号とは限りません)。文がストアドプロシージャ、ユーザ定義の機能、トリガ、またはイベントの一部ではない場合、値は NULL になります。
plan_hash	UNSIGNED BIGINT	プランの識別子を返します。
max_seconds	DOUBLE	現在のプランでの実行時、文で検出された最大ランタイムを返します。
sum_runtime	DOUBLE	現在のプランを使用した文に対する合計ランタイムを返します。
sum_square_runtime	DOUBLE	検出されたランタイムの 2 乗和を返します。この値は、標準偏差の計算に使用されます。
max_blocking_time	DOUBLE	現在のプランでの実行時、文で検出された最大ブロック時間を返します。
sum_blocking_time	DOUBLE	現在のプランを使用した文の実行に対する合計ブロック時間を返します。
num_exec	UNSIGNED BIGINT	現在のプランを使用して文が実行された回数を返します。
total_num_rows	UNSIGNED BIGINT	現在のプランでのすべての実行において、文によって返された、または修正されたローの合計数を返します。
last_max_time_utc	TIMESTAMP	最大ランタイムが最後に更新された日時を、協定世界時 (UTC) で返します。
last_time_utc	TIMESTAMP	現在のプランで文の統計が最後に更新された日時を、協定世界時 (UTC) で返します。

備考

このプロシージャは、ログに記録された各文に対して 1 つまたは複数のローを返しますが、各ローは使用された実行プランを示します。

指定した文に対するハッシュや、文に対して記録された結果を表示するには、stmt_text パラメータを指定します。ハッシュにデータがない場合、ハッシュと NULL を含む 1 つのローが返されます。文に対するデータをフェッチし、ハッシュを取得する場合は、stmt_hash パラメータを指定します。不要であれば、どちらのパラメータも指定しないでください。両方のパラメータを同時に指定することは、サーバによって許可されていません。

このシステムプロシージャは、結果を制御するパラメータを指定しない限り、サーバによって収集されたすべてのデータを返します。これらの統計を表示することで、文の実行が遅くなっている理由を特定できます。

注記

返された文のリストが長い場合、領域の制限により、すべてのデータがキャプチャされない場合があります。

権限

システムプロシージャに関する MONITOR および MANAGE PROFILING 権限が必要です。

関連する動作

なし。

例

次のクエリは、両方に共通するデータについて、ログに記録された各文に対するパフォーマンス統計を返します。

```
SELECT *
FROM dbo.sp_find_top_statements( ) TS
INNER JOIN SYS.GTSYSPERFCACHESTMT PS ON TS.stmt_hash = PS.stmt_hash
ORDER BY TS.stmt_hash;
```

すべてのデータについては、OUTER JOIN を使用します。

関連情報

sp_top_k_statements システムプロシージャ [1736 ページ]
GTSYSPERFCACHESTMT システムビュー [1792 ページ]
GTSYSPERFCACHEPLAN システムビュー [1791 ページ]

1.6.8.119 sp_forward_to_remote_server システムプロシージャ

アプリケーションがリモートサーバ上で SQL 文を実行し、その文によって生成される結果セットを取得できます。

構文

```
sp_forward_to_remote_server(
@server_name
, @sql
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、SQL 文が実行されるリモートサーバの名前を指定します。

@sql

この LONG VARCHAR パラメータを使用して、リモートサーバで実行する SQL 文を指定します。

備考

SQL 文はリモートサーバにそのまま送信されるため、データベースサーバが文を解析する必要はありません。

このシステムプロシージャを使用するには、CREATE SERVER 文でリモートサーバを定義する必要があります。

FORWARD TO 文とは異なり、sp_forward_to_remote_server はプロシージャ内で使用できます。ただし、このストアドプロシージャは、リモート結果セットのスキーマが任意なので、SELECT 文の FROM 句の中では使用できません。リモート結果セットのフェッチは、sp_forward_to_remote_server プロシージャで呼び出されるストアドプロシージャでカーソルを宣言することで実行できます。

注記

SQL 文が複数の結果セットを返す場合、sp_forward_to_remote_server ストアドプロシージャは順に各リモート結果セットを返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

このストアドプロシージャの実行に、ローカルで関連する動作はありません。しかし、リモートサーバ上で実行される SQL 文は任意なので、リモートサーバでは関連する動作がある可能性があります。

例

次の例は、sp_forward_to_remote_server ストアドプロシージャを使用して、RemoteSA と呼ばれるリモート SQL Anywhere17 データベースの非システムテーブルの数を決定します。

```
CALL sp_forward_to_remote_server( 'RemoteSA',
    'SELECT COUNT(*) FROM SYS.SYSTAB WHERE CREATOR NOT IN (0,3,6)' );
```

次の例は、sp_forward_to_remote_server ストアドプロシージャを使用して、newSalesData という名前の Microsoft Excel スプレッドシートからカラムを読み込みます。

```
call sp_forward_to_remote_server( 'RemoteExcel', 'SELECT * FROM newSalesData' );
```

関連情報

FORWARD TO 文 [1110 ページ]
CREATE SERVER 文 [914 ページ]

1.6.8.120 sp_generate_key_pair システムプロシージャ

新しい RSA 暗号化キーペアを作成します。

構文

```
sp_generate_key_pair system procedure(
keysize
, public_key
, private_key
, algorithm
)
```

パラメータ

keysize この正の INTEGER パラメータを使用して、キーサイズをビット単位で指定します。2048 ビットより短いキーは安全ではありません。

public_key この LONG VARCHAR 出力パラメータを使用して、パブリックキーを返します。

private_key この LONG VARCHAR 出力パラメータを使用して、プライベートキーを返します。

algorithm この任意の文字列パラメータを使用して、暗号化アルゴリズムを指定します。'RSA' のみサポートされます。デフォルトは RSA です。

備考

RSA キーペアを使用する場合、データを暗号化するにはパブリックキーを使用し、復号化するにはプライベートキーを使用する必要があります。どちらのキーも PEM 形式で返されます。

キーサイズが大きいければ大きいほどキーペアの作成にかかる時間が長くなりますが、セキュリティは高くなり、長いメッセージを暗号化できるようになります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例では、2048 ビット暗号化キーペアを作成します。

```
DECLARE @publickey LONG VARCHAR;
DECLARE @privatekey LONG VARCHAR;
CALL sp_generate_key_pair( 2048, @publickey, @privatekey );
```

1.6.8.121 sp_get_last_synchronize_result システムプロシージャ

SYNCHRONIZE 文によって最後に開始された同期に関する情報を返します。

構文

```
sp_get_last_synchronize_result(
[ @conn_id
[, @complete_only ] ]
)
```

パラメータ

@conn_id

任意の INTEGER パラメータを使用して、SYNCHRONIZE 文が実行された接続の接続 ID 番号を指定します。デフォルトは NULL です。パラメータを指定しない場合、または NULL の場合、現在の接続の接続 ID が使用されます。

@complete_only

完了した同期に関する情報が返されるようにするには、任意の BIT パラメータを 1 に設定します。現在アクティブな同期に関する情報が返されるようにするには、このパラメータを 0 に設定します。デフォルトは 1 です。

結果セット

カラム名	データ型	説明
row_id	BIGINT	ローがテーブルに挿入された順序を判断するために使用するテーブルのプライマリキー。
conn_id	UNSIGNED INTEGER	このイベントを生成した SYNCHRONIZE 文を実行した接続の接続 ID 番号。
result_time	TIMESTAMP	イベントが synchronize_results テーブルに追加された時刻。この値はシミュレートされたタイムゾーンの影響を受けます。
result_type	CHAR(128)	イベントのタイプ。
parm_id	INTEGER	各イベントには対応する 0 個以上のパラメータが存在する可能性があります。parm_id カラムは、各イベントに対応するパラメータの順序を決定します。
parm_result	LONG VARCHAR	イベントパラメータに関連付けられているメッセージテキスト。

備考

過去または現在の同期の詳細を表示するには、グローバル共有テンポラリテーブルの synchronize_results と synchronize_parameters を直接問い合わせる代わりに、sp_get_last_synchronize_result ストアドプロシージャを使用できます。このストアドプロシージャでは、指定した接続 ID 番号に対する最後の同期の結果のみを返します。パラメータを何も指定しないと、現在の接続に対して完了した最後の同期が返されます。

また、このストアドプロシージャを使用して、現在の接続以外の接続に対する同期の進行状況をモニタすることもできます。別の接続に対する同期の進行状況をモニタするには、次の手順に従います。

1. SELECT CONNECTION_PROPERTY 文を実行して、現在の接続の接続 ID を確認します。
2. SELECT CONNECTION_PROPERTY 文によって返された接続 ID を使用して SYNCHRONIZE 文を実行します。
3. 別の接続に対して、SELECT CONNECTION_PROPERTY 文を実行し、`complete_only` パラメータを 0 に設定します。指定した接続に対する最後の同期に関する情報が返されます (同期が未完了の場合も同様)。指定した接続に対する最後の同期に関する情報が返されます (同期が未完了の場合も同様)。

synchronize_parameters テーブルのイベントとそれに関連付けられた parm_id 値のリストを次に示します。

イベント	parm_id 値	説明
DBSC_EVENTTYPE_ERROR_MSG	0	エラーメッセージのテキストです。
	1	メッセージに関連付けられているメッセージ ID。
DBSC_EVENTTYPE_WARNING_MSG	0	警告メッセージのテキストです。
	1	メッセージに関連付けられているメッセージ ID。

イベント	parm_id 値	説明
DBSC_EVENTTYPE_INFO_MSG	0	情報メッセージのテキストです。
	1	メッセージに関連付けられているメッセージ ID。
DBSC_EVENTTYPE_PROGRESS_INDEX	0	新しい進行状況インデックス値。
DBSC_EVENTTYPE_PROGRESS_TEXT	0	新しい進行状況のテキスト。
DBSC_EVENTTYPE_TITLE	0	新しいウィンドウのタイトル。
DBSC_EVENTTYPE_SYNC_DONE	0	同期の終了コード。0 は成功を示します。
DBSC_EVENTTYPE_ML_CONNECT	0	使用されている通信プロトコル。
	1	使用されているネットワークプロトコルのオプション。
DBSC_EVENTTYPE_DOWNLOAD_COMMITTED	0	コミットされた挿入および更新操作の数。
	1	コミットされた削除操作の数。
DBSC_EVENTTYPE_UPLOAD_SENT	0	アップロードされた挿入操作の数。
	1	アップロードされた更新操作の数。
	2	アップロードされた削除操作の数。

権限

synchronize_results および synchronize_parameters 共有グローバル一時テーブル上の SELECT オブジェクトレベル権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例では、現在の接続に対して完了した最後の同期に関する情報を返します。

```
CALL sp_get_last_synchronize_result();
```

次の例では、sp_get_last_synchronize_result システムプロシージャを呼び出すときに名前付きパラメータを使用し、接続 ID 25 から開始された最後に完了した同期についての情報を返します。

```
CALL sp_get_last_synchronize_result(
     @conn_id=25,
     @complete_only=1);
```

関連情報

SYNCHRONIZE 文 [Mobile Link] [1358 ページ]

1.6.8.122 sp_has_role システムプロシージャ

プロシージャのインボーカに、指定されたシステム権限またはユーザ定義ロールが付与されているかどうかを返します。

構文

```
sp_has_role(
rolename
[, grant_type
[, throw_error ] ]
)
```

パラメータ

rolename

この CHAR(128) パラメータを使用して、チェックするシステム権限、またはユーザ定義ロールの名前を指定します。

grant_type

任意の CHAR(20) パラメータを使用して、チェックする付与のタイプを指定します。考えられる値は、次のとおりです。ADMIN または NO ADMIN (デフォルト)。

ADMIN に設定した場合、sp_has_role プロシージャは呼び出し側のユーザがその権限またはロールの管理権限を持っているかどうかをチェックします。NO ADMIN に設定した場合、sp_has_role プロシージャは呼び出し側のユーザがその権限またはロールを行使する権限を持っているかどうかをチェックします。

throw_error

任意の INTEGER パラメータを使用て、権限チェックの結果を示す値を返すかどうかを指定します。1 を指定した場合は、呼び出し側に指定された権限またはロールが付与されていない場合に、メッセージが返されます。0 を指定した場合は (デフォルト)、そのような場合でもメッセージは返されません。

戻り値

戻り値	説明
1	システム権限またはユーザ定義ロールが呼び出し側のユーザに付与されており、`grant_type` チェックにも合格しています。

戻り値	説明
0 またはパーミッションがありません: このコマンド／プロシージャを実行するためのパーミッションがありません。	システム権限またはユーザ定義ロールが呼び出し側のユーザに付与されておらず、grant_type チェックにも合格していません。throw_error が 1 に設定されている場合は、エラーメッセージによって値が 0 に置き換えられます。
-1	指定されたシステム権限またはユーザ定義ロールがありません。throw_error が 1 に設定されている場合でも、エラーメッセージは表示されません。

備考

throw_error 引数は、ユーザがストアドプロシージャのパーミッションチェックに失敗したときに「パーミッションがありません」というエラーメッセージを返すのに便利です。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

- 次の文は、管理権限のない CREATE ANY PROCEDURE システム権限が付与されているかどうかをチェックします。

```
SELECT sp_has_role( 'CREATE ANY PROCEDURE' );
```

- 次の文は、管理権限のある CREATE ANY PROCEDURE システム権限が付与されているかどうかをチェックし、付与されていない場合のみエラーを返します。

```
SELECT sp_has_role( 'CREATE ANY PROCEDURE', 'ADMIN', 1 );
```

- 次の文は、管理権限のない Role_A が付与されているかどうかをチェックします。

```
SELECT sp_has_role( 'Role_A' );
```

- 次の文は、管理権限のある Role_A が付与されているかどうかをチェックし、付与されていない場合のみエラーを返します。

```
SELECT sp_has_role( 'Role_A', 'ADMIN', 1 );
```

関連情報

sp_sys_priv_role_info システムプロシージャ [1734 ページ]
sp_proc_priv システムプロシージャ [1705 ページ]
sp_displayroles システムプロシージャ [1672 ページ]
sp_objectpermission システムプロシージャ [1698 ページ]

1.6.8.123 sp_http_listeners システムプロシージャ

指定されたデータベースに使用される HTTP および HTTPS 接続リスナーをリストします。

構文

```
sp_http_listeners( database-ID )
```

パラメータ

database-ID HTTP および HTTPS 接続リスナーがサービスを提供しているデータベースの ID。デフォルトは、現在のデータベース ID です。

結果セット

カラム名	データ型	説明
ip_address	VARCHAR(128)	接続リスナーの IP アドレスを返します。
port	INTEGER	接続リスナーのポート番号を返します。
dbname	VARCHAR(255)	接続リスナーがすべてのデータベースでサービス可能な場合は NULL を返しますが、そうでない場合はデータベース名を返します。
uri_prefix	LONG VARCHAR	接続リスナーでサービス可能なすべての URI のプレフィクスを返します。必要に応じて、http:// または https:// 識別子、IP アドレス、ポート番号（オプション）、およびデータベース名が含まれます。

備考

実行されている HTTP および HTTPS 接続リスナーごとに、ローが 1 つずつ結果セットに表示されます。ローは、接続リスナーが指定されたデータベースで Web サービスを実行できる場合にのみ表示されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

他のデータベースのこのシステムプロシージャを実行するには、次のいずれかのシステム権限が必要です。

- SERVER OPERATOR
- MONITOR
- MANAGE LISTENERS

例

次のコマンドを使用してデータベースサーバを起動します。

```
dbeng17 database1.db database2.db -xs http(port=80),http(port=8080;dbn=database1)
```

その後、database1 に接続して `SELECT * FROM sp_http_listeners();` を実行した場合、データベースサーバは次のような結果を返します。

ip_address	port	dbname	uri_prefix
127.0.0.1	80	NULL	http://127.0.0.1/database1/
127.0.0.1	8080	database1	http://127.0.0.1:8080/

database2 に接続して同じ文を実行した場合、データベースサーバは次の結果を返します。

ip_address	port	dbname	uri_prefix
127.0.0.1	80	NULL	http://127.0.0.1/database2/

1.6.8.124 sp_list_directory システムプロシージャ

指定されたディレクトリのファイルとサブディレクトリに関する情報を返します。

構文

```
sp_list_directory(
root_path
[, max_depth ]
)
```

パラメータ

root_path

この LONG NVARCHAR パラメータを使用して、ディレクトリの絶対パスまたは相対パスを指定します。相対パスは、サーバの作業ディレクトリです。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

max_depth

任意の INTEGER パラメータを使用して、トラバースするディレクトリの最大数を指定します。トラバースする `root_path` のすべてのサブディレクトリに、max_depth の値として Null、0、または負の値が指定されます。デフォルトは NULL です。

結果セット

カラム名	カラムの説明
`file_path`	LONG NVARCHAR - 指定されたディレクトリ内のファイルまたはサブディレクトリのパス。相対パスとして `directory_path` が指定されている場合は、返される `file_path` の値が相対値となります。それ以外の場合、`file_path` は絶対値となります。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。
`file_type`	NVARCHAR(1) - `file_path` の値がファイルの場合に F を、`file_path` の値がディレクトリの場合に D を指定します。
`file_size`	UNSIGNED BIGINT - `file_path` の値がディレクトリのときに、ファイルのサイズをバイト単位で指定するか、NULL を指定します。
`owner`	NVARCHAR(128) - ファイルまたはディレクトリの所有者。
`create_date_time`	TIMESTAMP WITH TIME ZONE - ファイルまたはディレクトリが作成された日時。データベースサーバのタイムゾーンで返されます。
`modified_date_time`	TIMESTAMP WITH TIME ZONE - ファイルまたはディレクトリが前回変更された日時。データベースサーバのタイムゾーンで返されます。
`access_date_time`	TIMESTAMP WITH TIME ZONE - ファイルまたはディレクトリが前回アクセスされた日時。データベースサーバのタイムゾーンで返されます。
`permissions`	VARCHAR(10) - ファイルまたはディレクトリのアクセスパーミッションのセット。

備考

なし

権限

READ FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、c:¥sqlany¥samples¥SQLAnywhere ディレクトリのファイルまたはサブディレクトリを一覧表示します。

```
CALL sp_list_directory('c:¥¥sqlany¥¥samples¥¥SQLAnywhere');
```

次の文は、c:¥sqlany¥samples¥SQLAnywhere ディレクトリと c:¥sqlany¥samples¥UltraLite ディレクトリにあるファイルおよびフォルダの合計数を返します。

```
SELECT COUNT(*) FROM sp_list_directory( 'c:¥¥sqlany¥¥samples¥¥SQLAnywhere' )
UNION ALL
SELECT COUNT(*) FROM sp_list_directory( 'c:¥¥sqlany¥¥samples¥¥UltraLite' );
```

c:¥sqlany¥samples¥SQLAnywhere と c:¥sqlany¥samples¥UltraLite の両方のディレクトリに同じファイル名、ファイルタイプ、およびファイルサイズのものが含まれる場合、次の文は 2 つのディレクトリからの結果を組み合わせて、ユニークな結果の合計数を示します。

```
SELECT COUNT(*) FROM
(
    SELECT REPLACE( file_path, 'c:¥¥sqlany¥¥samples¥¥SQLAnywhere', '' ) AS
file_path, file_type, file_size
    FROM sp_list_directory( 'c:¥¥sqlany¥¥samples¥¥SQLAnywhere' )
    UNION
    SELECT REPLACE( file_path, 'c:¥¥sqlany¥¥samples¥¥UltraLite', '' ) AS
file_path, file_type, file_size
    FROM sp_list_directory( 'c:¥¥sqlany¥¥samples¥¥UltraLite' )
) d;
```

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
sp_move_file システムプロシージャ [1696 ページ]

ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.125 sp_list_mutexes_semaphores システムプロシージャ

すべてのテンポラリ/永続ミューテックスおよびセマフォに関する情報を返します (各ミューテックスを保持している接続や、セマフォが待機しているかどうかなど)。

構文

```
sp_list_mutexes_semaphores( [oid] )
```

パラメータ

oid (内部使用のみ) UNSIGNED BIGINT ID パラメータ。デフォルトのパラメータ値 NULL を使用します。

結果セット

カラム名	データ型	カラムの説明
mutex_semaphore_id	UNSIGNED INTEGER	オブジェクトの ID。オブジェクトが永続的な場合、ID 値は SYSMUTEXSEMAPHORE システムビューで見つかった値と同じです。オブジェクトがテンポラリの場合、値は内部で生成されたテンポラリ ID です。
creator	VARCHAR(128)	ミューテックスまたはセマフォの所有者
"name"	VARCHAR(128)	ミューテックスまたはセマフォの名前
"type"	VARCHAR(9)	オブジェクトのタイプ。値 'mutex:T' は、トランザクションスコープミューテックスに使用され、値 'mutex:C' は接続スコープミューテックスに使用されます。また、値 'semaphore' はセマフォに使用されます。
is_temp	CHAR(1)	'Y' または 'N' は、ミューテックスまたはセマフォがテンポラリであるかどうかを示します。

カラム名	データ型	カラムの説明
is_dropped	CHAR(1)	'Y' または 'N' は、ミューテックスまたはセマフォがまだリリースされずに削除されているかどうかを示します。Y は、ミューテックスが削除されたがまだリリースが必要なことを示し、N はミューテックスが削除されてリリースされたことを示します。セマフォの場合、この値は常に N になります。
start_with	UNSIGNED INTEGER	セマフォの初期値。ミューテックスの場合、この値は常に NULL になります。
current_count	UNSIGNED INTEGER	セマフォの現在の値。ミューテックスの場合、この値は常に NULL になります。
currently_owned_by	LONG VARCHAR	現在ミューテックスをロックしている接続の ID を示すカンマ区切りのリスト。
currently_waited_for	LONG VARCHAR	現在セマフォの待機中または現在ミューテックス上でブロックされている接続の ID を示すカンマ区切りのリスト。

備考

なし

権限

そのシステムプロシージャに対する EXECUTE 権限と、MONITOR および UPDATE ANY MUTEX SEMAPHORE システム権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、データベース内のすべてのミューテックスおよびセマフォに関する情報を返します。

```
CALL sp_list_mutexes_semaphores();
```

1.6.8.126 sp_list_secure_feature_key システムプロシージャ

定義されたセキュリティ保護済み機能キーのリストを返します。

構文

sp_list_secure_feature_keys( )

結果セット

カラム名	データ型	説明
name	VARCHAR(128)	セキュリティ保護済み機能キーの名前。
features	LONG VARCHAR	セキュリティ保護済み機能キーによって有効化されたセキュリティ保護済み機能。

備考

このプロシージャを実行するには、接続で MANAGE_KEYS 機能を有効にする必要があります。

このプロシージャは、既存のセキュリティ保護済み機能キーの名前と、各キーによって有効にできる機能のセットを返します。

ユーザが MANAGE_FEATURES 機能と MANAGE_KEYS 機能を有効にしている場合、このプロシージャはすべてのセキュリティ保護済み機能キーのリストを返します。

ユーザが MANAGE_KEYS 機能のみを有効にしている場合、このプロシージャは、現在のユーザが有効にしている機能と同じ機能またはそのサブセットを有効にするキーを返します。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例が機能するには、サーバが次のオプションで起動されていることが必要です。`-sk securefkey`。

この例では、MANAGE_FEATURES 機能と MANAGE_KEYS 機能を含む SYSTEM セキュリティ保護済み機能キーを有効にして、sp_list_secure_feature_keys を使用して、現在のすべてのセキュリティ保護済み機能キーのリストを取得します。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_list_secure_feature_keys( );
```

関連情報

sp_alter_secure_feature_key システムプロシージャ [1657 ページ]
sp_create_secure_feature_key システムプロシージャ [1664 ページ]
sp_drop_secure_feature_key システムプロシージャ [1675 ページ]
sp_use_secure_feature_key システムプロシージャ [1751 ページ]

1.6.8.127 sp_login_environment システムプロシージャ

ユーザがログインするときの接続オプションを設定します。

構文

```
sp_login_environment( )
```

備考

sp_login_environment は、デフォルトで login_procedure データベースオプションによって呼び出されるプロシージャです。

このプロシージャは編集しないでください。ログイン環境を変更するには、異なるプロシージャを指すよう login_procedure オプションを変更します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

1.6.8.128 sp_move_directory システムプロシージャ

この関数は、source_path がポイントするディレクトリを、destination_path がポイントする宛先に移動します。

構文

```
sp_move_directory(
source_path
, destination_path
)
```

パラメータ

source_path

この LONG NVARCHAR パラメータを使用して、移動するディレクトリのパスを指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

destination_path

この LONG NVARCHAR パラメータを使用して、ディレクトリを移動する先のパスを指定します。ディレクトリは存在しない場合には作成されます。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数は、ソースディレクトリのすべてのファイルを指定されたディレクトリに移動し、ソースディレクトリを削除します。

権限

READ FILE および WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、SQLAnywhere ディレクトリをそのファイルとサブディレクトリを含めて c:¥temp¥SQLAnywhere に移動します。

```
SELECT sp_move_directory('c:¥¥sqlany¥¥samples¥¥SQLAnywhere', 'c:¥¥temp¥
¥SQLAnywhere');
```

元のディレクトリは削除されます。

次の文は、SQLAnywhere ディレクトリを元の場所にリストアします。

```
SELECT sp_move_directory('c:¥¥temp¥¥SQLAnywhere', 'c:¥¥sqlany¥¥samples¥
¥SQLAnywhere');
```

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]
sp_list_directory システムプロシージャ [1688 ページ]
sp_move_file システムプロシージャ [1696 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.129 sp_move_file システムプロシージャ

指定されたファイルをコンピュータ上の新しいディレクトリに移動します。

構文

```
sp_move_file(
source_path
, destination_path
)
```

パラメータ

source_path

この LONG NVARCHAR パラメータを使用して、移動するファイルのパスをファイル名を含めて指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して

解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

destination_path

LONG NVARCHAR パラメータを使用して、コンピュータ上の宛先ファイルパスをファイル名を含めて指定します。パスは、絶対パスまたは相対パスのどちらでもかまいません。相対パスは、データベースサーバの現在の作業ディレクトリに対して相対的に解決されます。サンドボックス機能が有効になっている場合、絶対パスと相対パスは、メインデータベースファイルがあるディレクトリを参照します。

戻り値

この機能は、成功の場合は 0 を、失敗の場合は 1 を返します。

備考

この関数は、ディレクトリ間でファイルを移動します。

権限

READ FILE および WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、ファイル `license.txt` を `c:¥temp` ディレクトリに移動して、新しい名前を付けます。

```
SELECT sp_move_file('c:¥¥sqlany¥¥license.txt', 'c:¥¥temp¥¥license.bkp');
```

ファイルは元の場所には存在しません。

次の文は、ファイルを元の名前と場所にリストアします。

```
SELECT sp_move_file('c:¥¥temp¥¥license.bkp', 'c:¥¥sqlany¥¥license.txt');
```

関連情報

sp_copy_directory システムプロシージャ [1660 ページ]
sp_copy_file システムプロシージャ [1662 ページ]
sp_create_directory システムプロシージャ [1663 ページ]
sp_delete_directory システムプロシージャ [1668 ページ]
sp_delete_file システムプロシージャ [1669 ページ]

sp_list_directory システムプロシージャ [1688 ページ]
sp_move_directory システムプロシージャ [1695 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.130 sp_objectpermission システムプロシージャ

指定されたロールまたはユーザ ID に付与されているオブジェクト権限のレポート、または指定されたオブジェクトまたは DB 領域に付与されているオブジェクト権限のレポートを生成します。

構文

```
sp_objectpermission(
object_name
, object_owner
, object_type
)
```

パラメータ

object_name

オブジェクト、DB 領域、ユーザ、またはロールの CHAR(128) 名。この引数が指定されていない場合は、現在のユーザのオブジェクト権限が表示されます。デフォルト値は NULL です。

object_owner

指定されたオブジェクト名のオブジェクト所有者の CHAR(128) 名。指定されたオブジェクト所有者が所有する指定されたオブジェクトのオブジェクト権限が表示されます。デフォルト値は NULL です。

object_type

データベースオブジェクトの CHAR(20) タイプ。値が指定されていない場合、すべてのオブジェクトタイプの権限が返されます。デフォルト値は NULL です。有効な値は次のとおりです。

- dbspace
- function
- materialized view
- procedure
- sequence
- table (カラムレベルのオブジェクト権限も表示されます)
- user
- view

結果セット

カラム名	データ型	説明
grantor	CHAR(128)	grantor のユーザ ID を返します。
grantee	CHAR(128)	grantee のユーザ ID を返します。
object_name	CHAR(128)	オブジェクトの名前を返します。
owner	CHAR(128)	オブジェクト所有者の名前を返します。
object_type	CHAR(20)	オブジェクトのタイプを示します。
column_name	CHAR(128)	カラムの名前を返します。
permission	CHAR(20)	権限の名前を返します。
grantable	CHAR(1)	権限が付与可能であるかどうかを示す値を返します。

備考

すべての引数が任意で、次のレポートを生成することができます。

- 入力がオブジェクト (テーブル、ビュー、プロシージャ、シーケンスなど) の場合、このプロシージャは、オブジェクト上でさまざまなオブジェクト権限を持っているすべてのユーザとロールのリストを表示します。
- 入力がロールまたはユーザの場合、このプロシージャは、ロールまたはユーザに付与されているすべてのオブジェクト権限のリストを表示します。
- 入力が DB 領域の場合、このプロシージャは、指定された DB 領域上で CREATE 権限を持つすべてのユーザとロールのリストを表示します。

sp_objectpermission を実行してユーザまたはロールオブジェクト権限を表示すると、ロール grants を通じて継承されたオブジェクト権限も表示されます。デフォルトでは object_type が NULL で、指定されたオブジェクト名に一致する既存のすべてのオブジェクトタイプに対するオブジェクト権限が表示されます。

標準

ANSI/ISO SQL 標準

標準になし。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

このプロシージャを別のユーザ ID、または別のユーザ ID が所有するオブジェクトに対して呼び出す場合は、MANAGE ANY OBJECT PRIVILEGE システム権限も必要です。

DB 領域に対してこのプロシージャを実行する場合は、MANAGE ANY DBSPACE システム権限も持っている必要があります。

関連する動作

なし

例

次の文は、DIAGNOSTICS システム権限に付与されているオブジェクトレベルの権限を返します。この例の目的のために、結果がトランケートされています。

```
CALL sp_objectpermission( 'SA_DEBUG' );
```

grantor	grantee	object_name	owner	object_type	column_name	permission	grantable
(NULL)	PUBLIC	sa_flush_statistics	dbo	PROCEDURE	(NULL)	EXECUTE	N
(NULL)	PUBLIC	sa_audit_string	dbo	PROCEDURE	(NULL)	EXECUTE	N
(NULL)	PUBLIC	sa_enable_auditing_type	dbo	PROCEDURE	(NULL)	EXECUTE	N
(NULL)	PUBLIC	sa_disable_auditing_type	dbo	PROCEDURE	(NULL)	EXECUTE	N
...	...	...	...	...	...	...	...

SA_DEBUG システムロールがオブジェクトレベルのパーミッションを持っているプロシージャが多数存在することが、結果に示されています。

次の文は、ml_server ユーザに付与されているオブジェクトレベルの権限を返します。

```
CALL sp_objectpermission( 'ml_server' );
```

次の文は、システム DB 領域上のオブジェクトレベルの権限を返します。

```
CALL sp_objectpermission( object_name='system', object_type='dbspace' );
```

関連情報

sp_sys_priv_role_info システムプロシージャ [1734 ページ]
sp_has_role システムプロシージャ [1685 ページ]
sp_proc_priv システムプロシージャ [1705 ページ]
sp_displayroles システムプロシージャ [1672 ページ]

1.6.8.131 sa_parse_json システムプロシージャ

SQL データ型を使用した JSON データの表現を返します。

構文

```
sp_parse_json(
  var
 , "json"
  [, maxlen ]
)
```

パラメータ

var

この LONG VARCHAR パラメータを使用して、作成するローカル変数の名前を指定します。変数のタイプは、実行時に決定されます。変数は存在しない場合には作成されます。

"json"

この LONG NVARCHAR パラメータを使用して、JSON データ構造体の文字列表現を指定します。

maxlen

この INTEGER パラメータを使用して、再帰の制限を指定します。指定された深さより下のノードは処理されません。代わりに、JSON のフラグメントとして返されます。

備考

このプロシージャは、JSON オブジェクトを処理して、処理されたデータを SQL データ型として返します。リターン変数のタイプは、プロシージャが実行されるときに決定されます。ほとんどの場合、リターン変数は ROW または ARRAY SQL のデータ型となります。

sp_parse_json システムプロシージャは、後続のインスタンスが異なる数のノードを返すと、ベースとなる ARRAY または OBJECT の VARCHAR フラグメントを返します。

JSON オブジェクト識別子は、データベースサーバで定義されている識別子ルールに準拠している必要があります。また、データベースサーバでは、基本となる ROW データ型および ARRAY データ型の場合と同じ制限が JSON データ型に課せられます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、いくつかのデータでテーブルを設定して、解析可能な JSON 文字列 (`[{"name":"Frank","age":51},{"name":"Bill","age":22},{"name":"Jackie","age":37}]`) を生成します。次に、JSON 文字列を SQL 配列データ型に解析します。

```
BEGIN
    DECLARE json_data LONG VARCHAR;
    CREATE LOCAL TEMPORARY TABLE test (
        name AS VARCHAR(64),
        age AS INT);
    INSERT INTO test (name, age) VALUES ('Frank',51);
    INSERT INTO test (name, age) VALUES ('Bill',22);
    INSERT INTO test (name, age) VALUES ('Jackie',37);
    SELECT * INTO json_data FROM test FOR JSON RAW;
    CALL sp_parse_json ( 'sql_array', json_data );
    SELECT sql_array [[row_num]] .name AS name, sql_array [[row_num]] .age AS age
        FROM sa_rowgenerator ( 1, 3 );
END;
```

関連情報

複合データ型 ROW および ARRAY [178 ページ]
識別子 [6 ページ]
ROW コンストラクタ [複合] [510 ページ]
ARRAY コンストラクタ [複合] [234 ページ]

1.6.8.132 sp_plancache_contents システムプロシージャ

接続用のプランキャッシュのコンテンツを返します。これには、平均実行時間、最小実行時間、最大実行時間、文のカーソルが構築された回数を示す統計情報が含まれます。

構文

```
sp_plancache_contents( [ connidparm ] )
```

パラメータ

connidparm

このオプションの INTEGER パラメータは、現在のデータベースへの接続の接続 ID です。デフォルトは NULL です。この場合、現在の接続のプランキャッシュに関する情報が返されます。

結果セット

カラム名	データ型	説明
stmt_type	CHAR(1)	文のタイプを示す値。考えられる値は、次のとおりです。 **B** バイパス文 **C** クライアント文 **E** イベント **M** テンポラリプロシージャまたバッチ **P** プロシージャ **T** トリガ
definition_id	UNSIGNED INTEGER	バイパス文の場合、SYSTAB システムビューで見つかった関連する table_id です。 クライアント文の場合、文を一意に識別する ID です。この ID は、文テキストのログが有効な場合に、要求レベルログで対応する文テキストと一緒に出力された値に一致します。 イベント文の場合、SYSEVENT システムビューで見つかった関連する event_id 値です。 テンポラリプロシージャの場合、接続でプロシージャを一意に識別する ID です。 バッチの場合、このフィールドは NULL です。 プロシージャ文の場合、SYSPROCEDURE システムビューで見つかった関連する proc_id 値です。 トリガ文の場合、SYSTRIGGER システムビューで見つかった関連する trigger_id 値です。
definition_position	UNSIGNED INTEGER	定義内における文の位置オフセット。バイパスおよびクライアント文の場合、このフィールドは NULL です。
definition_scope	SMALLINT	文コンテキストのスコープ。バイパスおよびクライアント文の場合、このフィールドは NULL です。

カラム名	データ型	説明
bypass_type	CHAR(1)	バイパス文の場合、バイパス文タイプを示す値。考えられる値は、次のとおりです。 **D** DELETE **I** INSERT **S** SELECT **U** UPDATE
status	CHAR(1)	キャッシュエントリの現在の状態を示す値。考えられる値は、次のとおりです。 **D** この文のキャッシュは無効です。 **P** この文のプランはキャッシュされます。 **R** この文の次の実行時にプランをキャッシュする準備ができています。 **T** この文のプランをキャッシュするかどうかを決定できるようにトレーニングしています。
plan_type	CHAR(1)	内部でのみ使用。
plan_signature	UNSIGNED BIGINT	内部でのみ使用。
PSID	UNSIGNED BIGINT	内部でのみ使用。
build_count	UNSIGNED INTEGER	この文のカーソルが構築された回数。
build_avg_msec	DOUBLE	この文のカーソルの構築に必要な平均回数 (ミリ秒)。
reusable_count	UNSIGNED INTEGER	この文の再利用可能なカーソルの実行回数。
reusable_avg_msec	DOUBLE	この文の再利用可能なカーソルの平均実行経過時間 (ミリ秒)。
reusable_min_msec	DOUBLE	この文の再利用可能なカーソルの最小実行経過時間 (ミリ秒)。
reusable_max_msec	DOUBLE	この文の再利用可能なカーソルの最大実行経過時間 (ミリ秒)。
nonreusable_count	UNSIGNED INTEGER	この文の再利用不可能なカーソルの実行回数。
nonreusable_avg_msec	DOUBLE	この文の再利用不可能なカーソルの平均実行経過時間 (ミリ秒)。
nonreusable_min_msec	DOUBLE	この文の再利用不可能なカーソルの最小実行経過時間 (ミリ秒)。
nonreusable_max_msec	DOUBLE	この文の再利用不可能なカーソルの最大実行経過時間 (ミリ秒)。

備考

接続のプランキャッシュにある現在のエントリを調べるには、このシステムプロシージャを使用します。これは、プランキャッシュのサイズを制御する max_plans_cached オプションの適切な値を選択する際に役立ちます。構築時間と実行時間に関する統計情報を使用しても、プランキャッシュによりパフォーマンスの問題が発生する文を特定することができます。この場合、max_plans_cached オプションを 0 に設定するか、FORCE OPTIMIZATION 句を SQL テキストに追加して文のキャッシュを選択的に無効にすることで、すべての文のプランキャッシュを無効にすることができます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。現在の接続以外の接続のプランキャッシュコンテンツを返すには、MANAGE CACHED PLANS システム権限も必要です。

関連する動作

なし

例

次の文は、現在の接続のプランキャッシュエントリを返します。

```
CALL sp_plancache_contents();
```

1.6.8.133 sp_proc_priv システムプロシージャ

システムプロシージャの実行に必要なシステム権限のリストを返します。

構文

```
sp_proc_priv( [proc_name] )
```

パラメータ

proc_name

権限を返すシステムプロシージャの名前を指定する CHAR(128) パラメータです。proc_name が指定されていない場合、すべてのシステムプロシージャに必要な権限が返されます。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
proc_name	CHAR(128)	システムプロシージャの名前。
privilege	LONG VARCHAR	システムプロシージャを実行するのに必要な権限のリスト。

備考

結果セットで、カンマによって区切られたシステムプロシージャに対する権限の数がリストされている場合、権限のいずれか 1 つで十分であることを意味します。システムプロシージャに対して複数の行が表示される場合は、システムプロシージャを実行するのに各行の権限の 1 つが必要になります。

proc_name がない状態で sp_proc_priv が呼び出された場合、その権限を必要とするすべてのシステムプロシージャに関する権限情報が返されます。権限を必要としないシステムプロシージャは結果に含まれません。

sp_proc_priv は、システムプロシージャがパーミッションチェックに合格することを保証する権限のみ返します。ただし、コンテキストによっては、システムプロシージャがパーミッションチェックに合格できるようにする他の権限が存在する可能性があります。それらの権限は sp_proc_priv によっては返されません。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例は、sa_table_fragmentation システムプロシージャを実行するのに必要な権限を返します。1 つの行のみが返されるため、sa_table_fragmentation を実行するには、カンマ区切りリストの権限のいずれか 1 つで十分です。

```
CALL sp_proc_priv( 'sa_table_fragmentation' );
```

proc_name	privilege
sa_table_fragmentation	MANAGE ANY STATISTICS、MONITOR

次の例は、sa_install_feature システムプロシージャの権限を返します。複数の行が返されるため、sa_install_feature の実行には各行の 1 つの権限が必要になります。

```
CALL sp_proc_priv( 'sa_install_feature' );
```

proc_name	privilege
sa_install_feature	SELECT ANY TABLE
sa_install_feature	MANAGE ANY SPATIAL OBJECT
sa_install_feature	MANAGE ANY OBJECT PRIVILEGE
sa_install_feature	CREATE ANY PROCEDURE、CREATE ANY OBJECT

関連情報

sp_sys_priv_role_info システムプロシージャ [1734 ページ]
sp_has_role システムプロシージャ [1685 ページ]
sp_displayroles システムプロシージャ [1672 ページ]
sp_objectpermission システムプロシージャ [1698 ページ]

1.6.8.134 sp_property_history システムプロシージャ

データベースによって追跡されたすべてのデータベースサーバプロパティの値を返します。

構文

```
sp_property_history( property, min_ticks )
```

パラメータ

property レポートするデータベースサーバプロパティの名前を指定するには、この VARCHAR(255) を使用します。NULL の場合、現在モニタされているすべてのプロパティがレポートされます。デフォルトは NULL です。

min_ticks チック値を指定して、記録されたプロパティ値のうち、指定したチック値以上の値を持つプロパティ値をすべて返します。デフォルトは NULL です。

結果セット

カラム名	データ型	説明
name	VARCHAR(255)	データベースサーバプロパティの名前。

カラム名	データ型	説明
チック	UNSIGNED BIGINT	プロパティ値を時系列に並べ替える単純増加の値。
time_recorded	TIMESTAMP WITH TIME ZONE	この値が記録されたシステム時刻。
time_delta	UNSIGNED INTEGER	システム時刻とは独立した、前回の記録以降のミリ秒数。
value	DOUBLE	データベースサーバプロパティの現在の値。
value_delta	DOUBLE	前回の記録以降にデータベースプロパティ値に加えられた変更。

備考

このシステムプロシージャは、データベースサーバで実行されるデータベース、および -phl データベースサーバオプションにより追跡されるデータベースサーバプロパティの結果を返します。データベースサーバは、コンピュータのシステムクロックいにょり測定されるチックを使用して、プロパティ値が記録される時系列の順序を追跡します。記録された各値には、単純増加するチック値と、GMT で測定された関連するタイムスタンプがあります。

`property-name` が NULL の場合、すべてのデータベースサーバプロパティ値が返されます。

`min_ticks` が NULL の場合、選択されたプロパティ (または、`property-name` が NULL の場合はすべてのプロパティ値) が返されます。

データベースが再起動された場合、別の実行中のデータベースにより現在追跡されているプロパティの履歴データのみ保持されます。

データベースサーバが再起動された場合、プロパティの履歴データと追跡設定は失われます。必要な追跡設定を再入力する必要があります。

次のすべてを満たす場合、データベース固有のプロパティ追跡設定も失われます。

- データベースが再起動された。
- データベースサーバで実行されている他のデータベースがデータベースサーバプロパティを追跡していない。
- データベースサーバプロパティがデータベースサーバレベルで追跡されていない。

ヒント

データベース固有の追跡設定を保持するには、データベースの起動イベントを作成し、これらの設定の保持を模倣します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

MANAGE ANY PROPERTY HISTORY システム権限が必要です。

関連する動作

なし。

例

記録されたデータベースサーバプロパティの値をすべて降順にリストするには、次の文を実行します。

```
SELECT * FROM sp_property_history( )
    ORDER BY ticks desc;
```

各ローがチック/時間ごとにデルタ変更を表示するよう、プロパティ履歴情報をピボットするには、次のクエリを実行します。

```
BEGIN
    DECLARE @props VARCHAR(120) ARRAY;
    SELECT *
        INTO #propdata
        FROM sp_property_history();
    SET @props = ( SELECT ARRAY_AGG( DISTINCT name ORDER BY name ) FROM
#propdata );
    SELECT *
        FROM ( SELECT name, ticks, time_recorded, time_delta, value_delta FROM
#propdata ) mysourcedata
        PIVOT ( SUM( value_delta ) delta FOR NAME IN @props ) mydata
        ORDER BY ticks DESC;
END;
```

各ローがチック/時間ごとに値の変更を表示するよう、プロパティ履歴情報をピボットするには、次のクエリを実行します。

```
BEGIN
    DECLARE @props VARCHAR(120) ARRAY;
    SELECT *
        INTO #propdata
        FROM sp_property_history();
    SET @props = ( SELECT ARRAY_AGG( DISTINCT name ORDER BY name ) FROM
#propdata );
    SELECT *
        FROM ( SELECT name, ticks, time_recorded, value FROM #propdata )
mysourcedata
        PIVOT ( SUM( value ) value FOR NAME IN @props ) mydata
        ORDER BY ticks DESC;
END;
```

関連情報

sa_db_option システムプロシージャ [1482 ページ]
sa_server_option システムプロシージャ [1606 ページ]
PROPERTY_IS_TRACKABLE 関数 [システム] [472 ページ]
sa_server_option システムプロシージャ [1606 ページ]
sa_db_option システムプロシージャ [1482 ページ]

1.6.8.135 sp_remote_columns システムプロシージャ

リモートテーブルにあるカラムとそれらのデータ型の記述のリストを生成します。

構文

```
sp_remote_columns(
@server_name
, @table_name
[, @table_owner
[, @table_qualifier ] ]
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、CREATE SERVER 文で指定されたサーバ名を含む文字列を指定します。

@table_name

この CHAR(128) パラメータを使用して、リモートテーブルの名前を指定します。

@table_owner

任意の CHAR(128) パラメータを使用して、table_name の所有者を指定します。デフォルトは % です。

@table_qualifier

任意の CHAR(128) パラメータを使用して、table_name が格納されているデータベースの名前を指定します。デフォルトは % です。

結果セット

カラム名	データ型	説明
database	CHAR(128)	データベース名。
owner	CHAR(128)	データベース所有者名。
table_name	CHAR(128)	テーブル名。
column_name	CHAR(128)	カラムの名前。
domain_id	SMALLINT	カラムのデータ型を示す INTEGER。
width	INTEGER	このカラムの意味は、データ型によって異なります。character 型の場合、width は文字数を表します。

カラム名	データ型	説明
scale	SMALLINT	このカラムの意味は、データ型によって異なります。NUMERIC データ型の場合、scale とは小数点以下の桁数です。
nullable	SMALLINT	NULL カラム値が許可される場合、値は 1 です。それ以外の場合、値は 0 です。
base_type_str	CHAR(4096)	カラムの物理的な型を表す注釈付きの型文字列。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

CREATE EXISTING TABLE 文を入力していて、カラムリストを指定している場合は、リモートテーブルで使用可能なカラムのリストを取得しておくと役立つ場合があります。sp_remote_columns はリモートテーブルのカラムとそのデータ型の記述のリストを生成します。データベースを指定する場合は、所有者または値 NULL を指定してください。

テーブルがリモートサーバに存在しない場合、プロシージャは空の結果セットを返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

標準

該当なし

例

次の例は、リモート SQL Anywhere データベースサーバ RemoteSA の ULProduct テーブルのカラムに関する情報を返します。テーブルの所有者は DBA です。

```
CALL sp_remote_columns( 'RemoteSA', 'ULProduct', 'DBA', null );
```

次の例は、リモートサーバ RemoteASE を使用する Adaptive Server Enterprise データベースの SYSOBJECTS テーブルのカラムに関する情報を返します。テーブルの所有者は指定されていません。

```
CALL sp_remote_columns( 'RemoteASE', 'sysobjects', null, 'Production' );
```

次の例は、リモートサーバ MyAccessDB を使用する Microsoft Access データベース `c:¥users¥me¥documents¥MyAccesDB.accdb` の Customers テーブルのカラムに関する情報を返します。Microsoft Access データベースにはテーブル所有者がないため、NULL が指定されます。

```
CALL sp_remote_columns( 'MyAccessDB', 'Customers', null, 'c:¥¥users¥¥me¥¥documents
¥¥MyAccesDB.accdb' );
```

関連情報

CREATE SERVER 文 [914 ページ]

1.6.8.136 sp_read_db_pages システムプロシージャ

指定されたページをキャッシュに読み込みます。

構文

```
sp_read_db_pages( dbspace_id [, page_bitmap [, in_background [,
limit_to_cache_size ] ] ] )
```

パラメータ

dbspace_id

この SMALLINT パラメータを使用して、ページの読み込み元の DB 領域を指定します。

page_bitmap この LONG VARBIT パラメータを使用して、キャッシュに読み込むページを指定します。1 を指定すると、そのページが読み込まれます。このパラメータが NULL (デフォルト) の場合、指定された DB 領域のすべてのページがキャッシュに読み込まれます。この値は、sp_db_cache_contents システムプロシージャから返される文字列です。

in_background

この BIT パラメータを使用して、実行中の他の操作への影響を最小限に抑えてページをバックグラウンドでロードするかどうかを指定します。

limit_to_cache_size

この BIT パラメータを使用して、キャッシュに格納できるページの数を制限するかどうかを指定します。キャッシュに収まる数より多くのページが読み込まれる場合にエラーを発生させる場合は、1 (デフォルト) を指定します。

備考

このプロシージャは、指定された DB 領域からキャッシュにページを読み込みます。

このシステムプロシージャを実行すると、まだキャッシュにないページのみが読み込まれます。要求されたページの一部がキャッシュに収まらない場合、エラーが返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。SERVER OPERATOR システム権限が必要です。

関連する動作

なし

例

1. 次の文を実行し、DB 領域を作成します。

```
CREATE DBSPACE mydbs
AS 'C:¥¥mydb¥¥mydbs.db';
```

2. 新しい DB 領域にテーブルを作成し、次の文を実行してデータを追加します。

```
CREATE TABLE mytable( col1 INT, col2 CHAR(128) ) IN mydbs;
```

```
INSERT INTO mytable
    SELECT column_id, column_name
    FROM sys.syscolumn;
```

```
COMMIT;
```

3. 次の文を実行し、新しい DB 領域の ID を特定します。

```
SELECT * FROM SYS.SYSDBSPACE WHERE dbspace_name = 'mydbs';
```

   新しい DB 領域の ID を書き留め、saved_cache_pages カラムが NULL であることに注目します。

4. 次の文を実行することで、新しい DB 領域のどのページが現在キャッシュにあるかを確認します。

```
SELECT * FROM dbo.sp_db_cache_contents( );
```

   上の文は、データベース内のすべての DB 領域の情報を返します。結果を新しい mydbs DB 領域に制限するには、次の文を実行します。ここで、`mydbs-ID` は SYS.SYSDBSPACE から取得された新しい mydbs DB 領域の ID です。

```
SELECT * FROM dbo.sp_db_cache_contents( mydbs-ID );
```

5. 次の文を実行することで、データベースの現在の安定した状態を保存します。

```
ALTER DATABASE SAVE CACHE;
```

6. その後、他の多くのトランザクションが実行され、キャッシュに大幅な変更が加えられた後、保存された安定した状態に戻すことができます。すべての DB 領域の安定した状態のページをすべてキャッシュに読み込むか、特定の DB 領域の安定した状態のページをすべてキャッシュに読み込むこともできます。
   すべての DB 領域の安定した状態のページをすべてキャッシュに読み込むには、次の文を実行します。

```
ALTER DATABASE RESTORE CACHE;
```

   ある DB 領域 (新しい DB 領域 mydbs など) の安定した状態のページをすべて読み込む場合は、次のような一連の文を実行します。

```
CREATE VARIABLE cache_pages LONG VARBIT;
```

```
SELECT saved_cache_pages
    INTO cache_pages
    FROM SYS.SYSDBSPACE
    WHERE dbspace_name='mydbs';
```

```
CALL dbo.sp_read_db_pages( mydbs-ID, cache_pages );
```

関連情報

sp_db_cache_contents システムプロシージャ [1666 ページ]

1.6.8.137 sp_read_etd システムプロシージャ

指定されたイベントトレースデータ (ETD) ファイルを読み込み、ファイルのコンテンツをローのセットとして返します。

構文

```
sp_read_etd(
filename
[,event_names=event-name[,...]]
[,host_names=host-name[,...]]
[,severity_level=integer]
[,record_start=integer]
[,record_end=integer]
[,timestamp_start=timestamp-with-timezone]
[,timestamp_end=timestamp-with-timezone]
[,regex ]=regular-expression[,...]]
)
```

パラメータ

filename

LONG NVARCHAR 値のこの ARRAY は、処理する ETD ファイルのフルパスと名前を指定します。複数のファイルが指定されると、この関数は指定されたすべての ETD ファイルからの結果を集約し、タイムスタンプの昇順でエントリを返します。相対パスは、データベースサーバの作業ディレクトリに対して相対的に解釈されます。ディスクサンドボックス機能を有効にしている場合は、パスが、メインデータベースファイルがあるディレクトリとそのサブディレクトリに制限されます。

event_names

この LONG NVARCHAR は、データが返されるイベントのカンマ区切りリストを提供します。

host_names

この LONG NVARCHAR は、イベントを生成したホストのカンマ区切りリストを提供します。

severity_level

この LONG NVARCHAR は、取得する最大重大度レベルを指定します。NULL は、すべての重大度レベルを返します。重大度レベルは次のとおりです。

重大度レベル	値
ALWAYS	0
CRITICAL	50
ERROR	100
WARING	150
INFORMATION	200
DEBUG	250

record_start

この UNSIGNED INT は、取得する最初のレコード番号を識別します。NULL の値は 1 として扱われます。最初のレコード番号は 1 です。

record_end

この UNSIGNED INT は、取得する最後のレコード番号を識別するために使用します。NULL の値は、最初のレコードから使用可能な最後のレコードまでのすべてのレコードを返すことを意味します。

timestamp_start

この TIMESTAMP WITH TIME ZONE は、最初のタイムスタンプを識別するために使用します。

timestamp_end

この TIMESTAMP WITH TIME ZONE は、最後のタイムスタンプを識別するために使用します。

regex

この LONG NVARCHAR は、文字列値を含むイベントフィールドをフィルタする正規文字列表現を指定するために使用します。フィルタリングは、論理 AND 操作として扱われます。カンマ区切りリストで構成されるパラメータの場合、フィルタリングが論理 OR 操作として扱われます。

結果セット

このプロシージャは、次のようにローで構成される結果セットを返します。

カラム	データ型	説明
timestamp	TIMESTAMP WITH TIME ZONE	イベントが発生した時刻
pid	UNSIGNED INT	イベントを生成したプロセス ID
tid	UNSIGNED INT	イベントを生成したスレッド ID
host_name	LONG NVARCHAR	イベントを生成したホスト
os	LONG NVARCHAR	オペレーティングシステムの名前
プロセッサ	LONG NVARCHAR	プロセッサタイプ
severity	LONG NVARCHAR	イベントの重大度レベル
severity_number	TINYINT	重大度レベル番号
event_name	LONG NVARCHAR	イベント名
event_data	LONG NVARCHAR	イベントのフィールドを記述する XML ドキュメント
record_id	UNSIGNED BIGINT	イベントが ETD ファイルに出現する順序。record_id の先頭は 0 であり、単調に増加していきます。0 は、ETD ファイル内の最初のレコードです。

備考

`event_data` XML ドキュメントの形式は以下のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<event_data num_fields="X">
    <field name="Y" type="YY">Z</field>
     ...
</event_data>
```

X はイベントのフィールド数、Y はフィールド名、YY はフィールドタイプ、Z はそのフィールドのデータです。フィールドタイプの有効な値は、integer (整数のサイズの場合)、string、binary、float、double です。バイナリデータは、base64 エンコードで表示されます。

権限

READ FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

sp_read_etd システムプロシージャを呼び出すには、OPENXML 演算子を使用して、1 つのタイプのイベントの event_data をカラムとして取得します。次の文は、sp_read_etd システムプロシージャのタイムスタンプと、event_data に格納された 2 つの値 (ID および INFO) を 1 つの結果セットを返します。

```
SELECT T1.timestamp, T2.ID, T2.INFO
FROM sp_read_etd( Array ('C:¥¥test¥¥trace1.etd' ), event_names = 'nop' ) AS T1,
     OPENXML ( T1.event_data, '/event_data' )
         WITH (
                ID LONG VARCHAR 'field[@name="id"]/text()',
                INFO LONG VARCHAR 'field[@name="info"]/text()'
               ) AS T2;
```

timestamp	ID	INFO
2013-09-10 14:10:33.089-04:00	1	abc
2013-09-10 14:10:36.054-04:00	2	def

次の文を使用し、OPENXML 演算子と LIST 関数を使用してさまざまなイベントの event_data を 1 つのカラムとして取得します。

```
SELECT
    T1.timestamp, T1.event_name, list( id.FIELD + '=' + id.TEXT ) as data
FROM sp_read_etd( Array( 'C:¥¥test¥¥trace3.etd' ) ) AS T1,
    OPENXML ( T1.event_data, '/event_data/field' )
        WITH ( FIELD LONG VARCHAR '@name', TEXT LONG VARCHAR 'text()' ) AS id,
GROUP BY T1.timestamp, T1.event_name, T1.event_data;
```

timestamp	event_name	data
2013-09-10 12:10:13.059-04:00	EVENT2	id=1,desc=abc
2013-09-10 14:10:46.054-04:00	EVENT1	id=2,location=XYZ,other_information=abc

正規表現を使用して、文字列値を含むイベントフィールドをフィルタします。次の文は、イベントデータカラムに文字列 'abc' を含むすべてのイベントフィールドを返します。

```
SELECT *
FROM sp_read_etd( Array( 'C:¥¥test¥¥trace1.etd' ), regex = '.*abc.*' );
```

1.6.8.138 sp_remote_exported_keys システムプロシージャ

指定されたプライマリテーブルに外部キーを持つテーブルに関する情報を表示します。

構文

```
sp_remote_exported_keys(
@server_name
, @table_name
[, @table_owner
[, @table_qualifier ] ]
```

```
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、リモートテーブルが格納されているサーバを指定します。

@table_name

この CHAR(128) パラメータを使用して、プライマリキーを格納するテーブルを指定します。

@table_owner

任意の CHAR(128) パラメータを使用して、プライマリテーブルの所有者を指定します。デフォルトでは "%" です。

@table_qualifier

任意の CHAR(128) パラメータを使用して、プライマリテーブルを格納するデータベースを指定します。デフォルトでは "%" です。

結果セット

カラム名	データ型	説明
pk_database	CHAR(128)	プライマリキーテーブルがあるデータベース。
pk_owner	CHAR(128)	プライマリキーテーブルの所有者。
pk_table	CHAR(128)	プライマリキーテーブル。
pk_column	CHAR(128)	プライマリキーカラムの名前。
fk_database	CHAR(128)	外部キーテーブルがあるデータベース。
fk_owner	CHAR(128)	外部キーテーブルの所有者。
fk_table	CHAR(128)	外部キーテーブル。
fk_column	CHAR(128)	外部キーカラムの名前。
key_seq	SMALLINT	キーのシーケンス番号。
fk_name	CHAR(128)	外部キーの名前。
pk_name	CHAR(128)	プライマリキーの名前。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

このプロシージャは、特定のプライマリテーブルに外部キーを持つリモートテーブルに関する情報を提供します。sp_remote_exported_keys システムプロシージャの結果セットには、データベース、所有者、テーブル、カラム、プライマリキ

ーと外部キーの両方の名前、外部キーカラムの外部キーシーケンスが含まれます。基本となる ODBC と JDBC 呼び出しのために結果セットが変わる場合もありますが、外部キーのテーブルとカラムに関する情報は常に返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

この例は、リモートサーバ RemoteSA 上の ULEmployee テーブルの外部キーの関係に関する情報を返します。

```
CALL sp_remote_exported_keys( 'RemoteSA', 'ULEmployee', 'DBA' );
```

関連情報

CREATE SERVER 文 [914 ページ]

1.6.8.139 sp_remote_imported_keys システムプロシージャ

指定された外部キーに対応するプライマリキーを持つリモートテーブルに関する情報を提供します。

構文

```
sp_remote_imported_keys(
@server_name
, @table_name
[, @table_owner
[, @table_qualifier ] ]
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、外部キーテーブルが格納されているサーバを指定します。このパラメータの値は必須です。

@table_name

この CHAR(128) パラメータを使用して、外部キーを格納するテーブルを指定します。このパラメータの値は必須です。

@table_owner

任意の CHAR(128) パラメータを使用して、外部キーテーブルの所有者を指定します。デフォルトでは "%" です。

@table_qualifier

任意の CHAR(128) パラメータを使用して、外部キーテーブルを含むデータベースを指定します。デフォルトでは "%" です。

結果セット

カラム名	データ型	説明
pk_database	CHAR(128)	プライマリキーテーブルがあるデータベース。
pk_owner	CHAR(128)	プライマリキーテーブルの所有者。
pk_table	CHAR(128)	プライマリキーテーブル。
pk_column	CHAR(128)	プライマリキーカラムの名前。
fk_database	CHAR(128)	外部キーテーブルがあるデータベース。
fk_owner	CHAR(128)	外部キーテーブルの所有者。
fk_table	CHAR(128)	外部キーテーブル。
fk_column	CHAR(128)	外部キーカラムの名前。
key_seq	SMALLINT	キーのシーケンス番号。
fk_name	CHAR(128)	外部キーの名前。
pk_name	CHAR(128)	プライマリキーの名前。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

外部キーは、対応するプライマリキーを持つ別のテーブル内のローを参照します。このプロシージャを使用すると、特定の外部テーブルに対応するプライマリキーを持つリモートテーブルのリストを取得できます。sp_remote_imported_keys の結果セットには、データベース、所有者、テーブル、カラム、プライマリキーと外部キーの両方の名前、外部キーカラムの外部キーシーケンスが含まれます。基本となる ODBC と JDBC 呼び出しのために結果セットが変わる場合もありますが、プライマリキーのテーブルとカラムに関する情報は常に返されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の例は、リモートサーバ RemoteSA 上の ULOrder テーブルの外部キーに対応するテーブルとプライマリキーを返します。

```
CALL sp_remote_imported_keys( 'RemoteSA', 'ULOrder', 'DBA' );
```

関連情報

CREATE SERVER 文 [914 ページ]

1.6.8.140 sp_remote_pcols システムプロシージャ

リモートテーブルにあるカラムとそれらのデータ型の記述のリストを生成します。

構文

```
sp_remote_pcols(
@server_name
, @sp_name
[, @sp_owner
[, @sp_qualifier ] ]
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、CREATE SERVER 文で指定されたサーバ名を含む文字列を指定します。

@sp_name

この CHAR(128) パラメータを使用して、リモートテーブルの名前を指定します。

@sp_owner

任意の CHAR(128) パラメータを使用して、sp_name の所有者を指定します。デフォルトでは "%" です。

@sp_qualifier

任意の CHAR(128) パラメータを使用して、sp_name が格納されているデータベースの名前を指定します。デフォルトでは "%" です。

結果セット

カラム名	データ型	説明
database	CHAR(128)	データベース名。
owner	CHAR(128)	データベース所有者名。
proc_name	CHAR(128)	ストアドプロシージャ名。
parm_name	CHAR(128)	パラメータ名または結果セットカラム。
parm_mode	CHAR(10)	パラメータのモードまたは結果セットカラム (IN、OUT、INOUT、RESULT)。
domain_id	SMALLINT	パラメータまたは結果セットカラムのデータ型を示す INTEGER。
width	INTEGER	このカラムの意味は、データ型によって異なります。character 型の場合、width は文字数を表します。
scale	SMALLINT	このカラムの意味は、データ型によって異なります。NUMERIC データ型の場合、scale とは小数点以下の桁数です。
nullable	SMALLINT	NULL パラメータ値が許可される場合、値は 1 です。それ以外の場合、値は 0 です。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

CREATE PROCEDURE ...AT 文を入力してパラメータリストを指定した場合、または返される結果セットに関する情報が必要な場合、リモートストアドプロシージャで使用可能なパラメータのリストと結果セットカラムを取得しておくと役立つ場合があります。sp_remote_pcols は、パラメータのリスト、リモートストアドプロシージャの結果セットカラム、およびそれらのデータ型の説明を作成します。データベースを指定する場合は、所有者または値 NULL を指定してください。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

標準

該当なし

例

次の例は、リモート SQL Anywhere データベースサーバ RemoteSA の ULOrderDownload ストアドプロシージャパラメータと結果セットカラムに関する情報を返します。ストアドプロシージャの所有者は DBA です。

```
CALL sp_remote_pcols('RemoteSA', 'ULOrderDownload', 'DBA');
```

次の例は、リモートサーバ RemoteASE を使用する Adaptive Server Enterprise データベース Production の col_name ストアドプロシージャのパラメータと結果セットカラムに関する情報を返します。リモートプロシージャの所有者は指定されていません。

```
CALL sp_remote_pcols( 'RemoteASE', 'col_name', null, 'Production' );
```

関連情報

CREATE SERVER 文 [914 ページ]

1.6.8.141 sp_remote_primary_keys システムプロシージャ

リモートデータアクセスを使用してリモートテーブルについてのプライマリキー情報を提供します。

構文

```
sp_remote_primary_keys(
  @server_name
 , @table_name
  [, @table_owner
  [, @table_qualifier ] ]
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、リモートサーバ名を指定します。

@table_name

この CHAR(128) パラメータを使用して、リモートテーブルの名前を指定します。

@table_owner

任意の CHAR(128) パラメータを使用して、リモートテーブルの所有者を指定します。デフォルトでは "%" です。

@table_qualifier

任意の CHAR(128) パラメータを使用して、リモートデータベースの名前を指定します。デフォルトでは "%" です。

結果セット

カラム名	データ型	説明
database	CHAR(128)	リモートデータベースの名前。
owner	CHAR(128)	テーブルの所有者。
table_name	CHAR(128)	テーブルの名前。
column_name	CHAR(128)	プライマリキーカラムの名前。
key_seq	SMALLINT	プライマリキーのシーケンス番号。
pk_name	CHAR(128)	プライマリキーの名前。

備考

このシステムプロシージャは、リモートデータアクセスを使用してリモートテーブルについてのプライマリキー情報を提供します。

基本となる ODBC 呼び出しに違いがあるため、返される情報は、サーバに指定されたリモートデータアクセスクラスによって、カタログの値やデータベースの値と多少異なります。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

標準

該当なし

関連する動作

なし。

例

次の例は、SQL Anywhere リモートサーバ RemoteSA の DBA が所有するテーブルのプライマリキーに関する情報を返します。

```
CALL sp_remote_primary_keys( 'RemoteSA', null, 'DBA' );
```

Adaptive Server Enterprise サーバ RemoteASE の production データベースにある Fred が所有するすべてのテーブルのプライマリキーのリストを取得します。

```
CALL sp_remote_primary_keys( 'RemoteASE', null, 'Fred', 'production' );
```

関連情報

CREATE SERVER 文 [914 ページ]

1.6.8.142 sp_remote_procedures システムプロシージャ

リモートサーバ上のプロシージャのリストを返します。

構文

```
sp_remote_procedures(
  @server_name
  [, @sp_name
  [, @sp_owner
  [, @sp_qualifier ] ] ]
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、リモートサーバ名を指定します。
@sp_name

この CHAR(128) パラメータを使用して、リモートサーバ名を指定します。デフォルトでは "%" です。
@sp_owner

任意の CHAR(128) パラメータを使用して、リモートストアドプロシージャの所有者を指定します。デフォルトでは "%" です。
@sp_qualifier

任意の CHAR(128) パラメータを使用して、`sp_name` が格納されているデータベースを指定します。デフォルトでは "%" です。

結果セット

カラム名	データ型	説明
database	CHAR(128)	リモートデータベースの名前。
owner	CHAR(128)	ストアドプロシージャの所有者名。
proc_name	CHAR(128)	ストアドプロシージャ名。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

データベースサーバを設定するときに、特定のサーバ上で使用可能なリモートストアドプロシージャのリストを取得しておくと役立つ場合があります。このプロシージャは、サーバ上のストアドプロシージャのリストを返します。

このプロシージャには 4 つのパラメータを指定できます。ストアドプロシージャ、所有者、またはデータベース名を指定すると、ストアドプロシージャのリストはその引数に当てはまるものだけに限定されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例は、RemoteSA という名前の SQL Anywhere リモートサーバの DBA が所有するストアドプロシージャに関する情報を返します。

```
CALL sp_remote_procedures( 'RemoteSA', null, 'DBA' );
```

次の例は、Adaptive Server Enterprise リモートサーバ RemoteASE の production データベースの Fred が所有するストアドプロシージャに関する情報を返します。

```
CALL sp_remote_procedures( 'RemoteASE', null, 'Fred', 'production' );
```

関連情報

CREATE PROCEDURE 文 [891 ページ]
CREATE SERVER 文 [914 ページ]

1.6.8.143 sp_remote_tables システムプロシージャ

サーバ上のテーブルのリストを返します。

構文

```
sp_remote_tables(
  @server_name
  [, @table_name
  [, @table_owner
  [, @table_qualifier
  [, @with_table_type ] ] ] ]
)
```

パラメータ

@server_name

この CHAR(128) パラメータを使用して、リモートサーバ名を指定します。

@table_name

任意の CHAR(128) パラメータを使用して、リモートテーブルの名前を指定します。デフォルトでは "%" です。

@table_owner

任意の CHAR(128) パラメータを使用して、リモートテーブルの所有者を指定します。デフォルトでは "%" です。

@table_qualifier

任意の CHAR(128) パラメータを使用して、table_name が格納されているデータベースを指定します。デフォルトでは "%" です。

@with_table_type

任意の BIT パラメータを使用して、リモートテーブルのタイプを含めるよう指定します。デフォルトは 0 です。結果セットに、テーブルのタイプを一覧するカラムを結果セットに含める場合は 1 を、そうでない場合は 0 を指定します。

結果セット

カラム名	データ型	説明
database	CHAR(128)	リモートデータベースの名前。
owner	CHAR(128)	テーブル所有者の名前。
table_name	CHAR(128)	テーブルの名前。
table_type	CHAR(128)	テーブルタイプを指定します。値は、リモートサーバのタイプによって異なります。たとえば、指定できる値として TABLE、VIEW、SYS、GBL TEMP があります。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

データベースサーバを設定するときに、特定のサーバ上で使用可能なリモートテーブルのリストを取得しておくと役立つ場合があります。このプロシージャは、サーバ上のテーブルのリストを返します。

このプロシージャには 5 つのパラメータを指定できます。テーブル、所有者、またはデータベース名を指定すると、テーブルのリストはその引数に当てはまるものだけに限定されます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

標準

該当なし

例

次の例は、SQL Anywhere リモートサーバ RemoteSA の DBA が所有するテーブルに関する情報を返します。

```
CALL sp_remote_tables( 'RemoteSA', null, 'DBA' );
```

Adaptive Server Enterprise サーバ RemoteASE の production データベースにある Fed が所有するすべてのテーブルのリストを取得します。

```
CALL sp_remote_tables( 'RemoteASE', null, 'Fred', 'production' );
```

サーバ RemoteExcel で参照されている ODBC データソースから、使用可能なすべての Microsoft Excel ワークシートのリストを取得します。

```
CALL sp_remote_tables( 'RemoteExcel' );
```

関連情報

CREATE SERVER 文 [914 ページ]

1.6.8.144 sp_servercaps システムプロシージャ

リモートサーバの機能についての情報を表示します。

構文

```
sp_servercaps( @server_name )
```

パラメータ

@server_name

CREATE SERVER 文で定義されたサーバを指定する CHAR(128) パラメータです。`@server_name` は、CREATE SERVER 文で使用されるサーバと同じサーバ名です。

結果

カラム	タイプ	説明
capid	INTEGER	機能識別子。
capname	CHAR(128)	機能の名前。
capvalue	CHAR(128)	機能の設定。通常は T (true) または F (false)。

備考

このシステムプロシージャを使用するには、サーバを CREATE SERVER 文で定義する必要があります。

このプロシージャは、リモートサーバの機能についての情報を表示します。この機能の情報は、どのくらいの SQL 文をリモートサーバに転送できるかを判断するために使用されます。サーバの機能をリストする ISYSCAPABILITY システムテーブルは、最初のリモートサーバに接続するまでは設定されません。

標準

該当なし

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

リモートサーバ RemoteSA に関する情報を表示します。

```
CALL sp_servercaps( 'RemoteSA' );
```

関連情報

SYSCAPABILITY システムビュー [1794 ページ]
SYSCAPABILITYNAME システムビュー [1794 ページ]
CREATE SERVER 文 [914 ページ]

1.6.8.145 sp_start_listener システムプロシージャ

新しい接続リスナーを起動します。

構文

```
sp_start_listener(
type
, address
[ , options ]
)
```

パラメータ

type

この VARCHAR (12) パラメータを使用して、起動する接続リスナーのタイプを指定します。この値は、**sharedmemory**、**shmem**、**tcpip**、**tcp**、**http**、**https** のいずれかです。

address

この VARCHAR (100) パラメータを使用して、起動する接続リスナーのアドレスを指定します。このアドレスは、ポート番号をコロン (:) で区切った数値の IP アドレス (0.0.0.0:9998 など) か、ポート番号のない IP アドレスです。ポート番号のある IPv6 アドレスの場合、アドレスを括弧で囲み、次にコロンとポート番号を追加します。ポート番号を指定しないと、デフォルトのポート (TCPIP:2638、HTTP:80、HTTPS:443) が使用されます。

TCP/IP と HTTP(S) に対してポート番号を指定する場合、アドレスパラメータは 1 ～ 65535 のポート番号にすることができます。この場合、そのポート番号を使用して、すべての利用可能な IP アドレス上でリスナが起動し、データベースサーバはまるでそのポート番号が -x TCPIP に対する ServerPort (PORT) プロトコルオプションまたは -xs HTTPS(S) データベースサーバオプションとして提供されたかのように動作します。

使用可能なすべての IPv4 アドレスまたは IPv6 アドレスを指定するには、IP アドレス "0.0.0.0" または "(::)" を指定します。

パーソナルデータベースサーバは、ループバック IP アドレス (127.0.0.1 など) のみ受け入れます。

このパラメータは、共有メモリでは無視されます。共有メモリでは、NULL を指定します。

options

ネットワークプロトコルオプションのセミコロン区切りリストを指定するには、この LONG VARCHAR パラメータを使用します。このパラメータは、共有メモリまたは TCP/IP 接続リスナーを使用している場合は無視されます。

> **注記**
>
> `options` パラメータを使用している場合、ServerPort (PORT) プロトコルオプションまたは MyIP (ME) プロトコルオプションを指定することはできません。

備考

新しい接続リスナーは、使用可能なポート番号のうち、次のリストから最初に見つかったポート番号を使用します。

- アドレスパラメータにより指定されたポート。
- デフォルトポート (TCP/IP の場合は 2638、HTTP の場合は 80、HTTPS の場合は 443)。

TCP/IP 接続リスナーは、データベースサーバ起動時に -ec データベースサーバオプションで指定した暗号化設定を使用します。

共有メモリ接続リスナーは、データベースサーバ起動時に -es データベースサーバオプションが指定されたかどうかに関係なく作成できます。sp_start_listener システムプロシージャを使用して共有メモリ接続リスナーを起動すると、データベースサーバとの間で暗号化されない接続が常に許可されます。

権限

MANAGE LISTENERS システム権限が必要です。

> **例**
>
> ローカル接続のみを許可するデータベースサーバが起動しているとします。リモートでデバッグを行うのに都合のよい問題が発生します。ポート 9998 を使用したデータベースサーバへのリモート接続を許可するため、ユーザは共有メモリを使用してデータベースサーバに接続し、次の文を実行します。
>
> ```
> CALL sp_start_listener ( 'tcpip' , '0.0.0.0:9998' );
> ```

関連情報

sp_stop_listener システムプロシージャ [1733 ページ]

1.6.8.146 sp_stop_listener システムプロシージャ

既存の接続リスナーを停止します。

構文

```
sp_stop_listener(
type
, address
[ , force ]
)
```

パラメータ

type

この VARCHAR (12) パラメータを使用して、停止する接続リスナーのタイプを指定します。この値は、**sharedmemory**、**shmem**、**tcpip**、**tcp**、**http**、**https** のいずれかです。

address

この VARCHAR (100) パラメータを使用して、停止する接続リスナーのアドレスを指定します。このアドレスは、ポート番号をコロン (:) で区切った IP アドレスか、ポート番号のない IP アドレスです。ポート番号のある IPv6 アドレスの場合、アドレスを括弧で囲み、次にコロンとポート番号を追加します。ポート番号が指定されていない場合、デフォルトのポート (TCPIP:2638、HTTP:80、HTTPS:443) が使用されます。TCP/IP と HTTP(S) の場合、アドレスパラメータは 1 ～ 65535 のポート番号にすることができます。ポート番号のみを指定した場合、そのポートを使用している指定リスナーがすべて指定されます。

使用可能なすべての IPv4 アドレスまたは IPv6 アドレスを指定するには、IP アドレス "0.0.0.0" または "(::)" を指定します。

パーソナルデータベースサーバは、ループバック IP アドレス (127.0.0.1 など) のみ受け入れます。

このパラメータは、共有メモリでは無視されます。共有メモリでは、NULL を指定します。

force

実行されている最後のネットワークドライバリスナーである場合に接続リスナーを強制的に指定するには、**1** を指定します。デフォルトは 0 です。

備考

sp_stop_listener システムプロシージャは、接続リスナーで起動された新しい接続のみ停止します。既存の接続は変更されません。

次のいずれかに当てはまる場合、`force` パラメータを **1** に設定します。

- 接続リスナーが最後の TCP/IP リスナーであり、共有メモリが有効になっていない場合。
- 接続リスナーが共有メモリであり、実行されている TCP/IP リスナーがない場合。

- 接続リスナーが最後の HTTP リスナーであり、実行されている HTTPS リスナーがない場合。
- 接続リスナーが最後の HTTPS リスナーであり、実行されている HTTP リスナーがない場合。

権限

MANAGE LISTENERS システム権限が必要です。

例

ローカル接続のみを許可するデータベースサーバが起動しているとします。リモートでデバッグを行うのに都合のよい問題が発生します。ポート 9998 を使用したデータベースサーバへのリモート接続を許可するため、ユーザは共有メモリを使用してデータベースサーバに接続し、次の文を実行します。

```
CALL sp_start_listener ( 'tcpip' , '0.0.0.0:9998' );
```

問題が解決されたら、次の文を実行して接続リスナーをシャットダウンします。

```
CALL sp_stop_listener ( 'tcpip' , '0.0.0.0:9998' );
```

関連情報

sp_start_listener システムプロシージャ [1731 ページ]

1.6.8.147 sp_sys_priv_role_info システムプロシージャ

システム権限とシステムロール間の 1 対 1 のマッピングを返します。

構文

```
sp_sys_priv_role_info( )
```

結果セット

カラム	タイプ	説明
sys_priv_name	CHAR(128)	システム権限の名前。
sys_priv_role_name	CHAR(128)	対応するシステムロールの名前。

カラム	タイプ	説明
sys_priv_id	UNSIGNED INTEGER	システムロールの ID 番号。

備考

sp_sys_priv_role_info は、システム権限とシステムロールおよびロール ID 間の完全な 1 対 1 のマッピングをレポートします。このプロシージャは、各システム権限のローを返します。

このプロシージャは主に内部で使用するよう設計されており、付与されているシステムロールを対応するシステム権限にマップしたり、その逆にマップしたりするのに必要なユーティリティが使用します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、対応するシステムロールへのシステム権限のマッピングを返します。

```
CALL sp_sys_priv_role_info();
```

sys_priv_name	sys_priv_role_name	sys_priv_id
VALIDATE ANY OBJECT	SYS_VALIDATE_ANY_OBJECT_ROLE	2147483819
REORGANIZE ANY OBJECT	SYS_REORGANIZE_ANY_OBJECT_ROLE	2147483820
BACKUP DATABASE	SYS_BACKUP_DATABASE_ROLE	2147483821
MANAGE ANY EVENT	SYS_MANAGE_ANY_EVENT_ROLE	2147483822
ALTER DATABASE ...	SYS_ALTER_DATABASE_ROLE	2147483823
SERVER OPERATOR	SYS_SERVER_OPERATOR_ROLE	2147483824
UPGRADE ROLE	SYS_UPGRADE_ROLE_ROLE	2147483825
MANAGE ANY LDAP SERVER	SYS_MANAGE_ANY_LDAP_SERVER_ROLE	2147483827
MANAGE CERTIFICATES	SYS_MANAGE_CERTIFICATES_ROLE	2147483828

sys_priv_name	sys_priv_role_name	sys_priv_id
CREATE ANY SEQUENCE	SYS_CREATE_ANY_SEQUENCE_ROLE	2147483829
...	...	...

関連情報

sp_has_role システムプロシージャ [1685 ページ]
sp_proc_priv システムプロシージャ [1705 ページ]
sp_displayroles システムプロシージャ [1672 ページ]
sp_objectpermission システムプロシージャ [1698 ページ]

1.6.8.148 sp_top_k_statements システムプロシージャ

最大ランタイムを持つ文とプランの組み合わせを、上位から指定した数だけ返します。

構文

```
sp_top_k_statements( [ k ] )
```

パラメータ

k

任意の UNSIGNED INTEGER パラメータを使用して、返されるレコードの数を指定します。デフォルト値は 1000 です。

結果セット

カラム名	データ型	説明
stmt_hash	UNSIGNED BIGINT	文の識別子を返します。

カラム名	データ型	説明
owner_name	CHAR(128)	ストアドプロシージャの所有者名を返します。文がストアドプロシージャ、ユーザ定義の機能、トリガ、またはイベントの一部ではない場合、値は NULL になります。文がストアドプロシージャの一部として実行され、のちにストアドプロシージャが削除された場合も、値は NULL となります。ストアドプロシージャの一部として実行された文は、文のテキストではなくプロシージャに関連するハッシュを取得します。
proc_name	CHAR(128)	ストアドプロシージャの所有者名を返します。文がストアドプロシージャ、ユーザ定義の機能、トリガ、またはイベントの一部ではない場合、値は NULL になります。文がストアドプロシージャの一部として実行され、のちにストアドプロシージャが削除された場合も、値は NULL となります。ストアドプロシージャの一部として実行された文は、文のテキストではなくプロシージャに関連するハッシュを取得します。
reusable_stmt_id	UNSIGNED INTEGER	プロシージャ内の文に割り当てられた一意の識別子を返します (行番号とは限りません)。文がストアドプロシージャ、ユーザ定義の機能、トリガ、またはイベントの一部ではない場合、値は NULL になります。
plan_hash	UNSIGNED BIGINT	プランの識別子を返します。
max_seconds	DOUBLE	現在のプランでの実行時、文で検出された最大ランタイムを返します。
avg_seconds	DOUBLE	現在のプランを使用した文に対する平均ランタイムを返します。
stddev_seconds	DOUBLE	現在のプランを使用した文のランタイムの標準偏差を返します。標準偏差は 2 乗和を使用して算出されます。それまで検出された文の実行が 1 つのみの場合、値は NULL となります。
max_blocking_seconds	DOUBLE	現在のプランを使用している文で検出された最大ブロック時間を返します。
avg_blocking_seconds	DOUBLE	現在のプランを使用している文で検出された平均ブロック時間を返します。
num_exec	UNSIGNED BIGINT	現在のプランを使用して文が実行された回数を返します。
total_num_rows	UNSIGNED BIGINT	現在のプランでのすべての実行において、文によって返された、または修正されたローの合計数を返します。
last_max_time_utc	TIMESTAMP	最大ランタイムが最後に更新された日時を、協定世界時 (UTC) で返します。

カラム名	データ型	説明
last_time_utc	TIMESTAMP	現在のプランで文の統計が最後に更新された日時を、協定世界時 (UTC) で返します。

備考

このシステムプロシージャを使用して、実行に最も時間がかかるのはどの文/プランの組み合わせかを判別します。実行時間の長い文をいくつ返すかを、(k) パラメータで指定します。

> **注記**
>
> 返された文のリストが長い場合、領域の制限により、すべてのデータがキャプチャされない場合があります。

権限

システムプロシージャに関する MONITOR および MANAGE PROFILING 権限が必要です。

関連する動作

なし。

例

次のクエリは、検出された中で最も長いランタイムを持つ上位の文を返します。

```
SELECT *
FROM dbo.sp_top_k_statements( ) TS
OUTER JOIN SYS.GTSYSPERFCACHESTMT PS ON TS.stmt_hash = PS.stmt_hash
ORDER BY TS.stmt_hash;
```

関連情報

sp_find_top_statements システムプロシージャ [1677 ページ]
GTSYSPERFCACHESTMT システムビュー [1792 ページ]
GTSYSPERFCACHEPLAN システムビュー [1791 ページ]

1.6.8.149 sp_trace_event_fields システムプロシージャ

指定されたトレースイベントのフィールドに関する情報を返します。

構文

```
sp_trace_event_fields(
    [ event_name
    [, include_audit_events ] ]
  )
```

パラメータ

event_name

オプションの CHAR(256) パラメータを使用して、トレースイベント名を指定します。デフォルトは NULL です。

include_audit_events 任意の BIT パラメータを使用して、監査イベントを返すかどうかを指定します。このパラメータは、0 (監査イベントを返さない) または 1 (監査イベントを返す) に設定できます。デフォルトでは、監査イベントは返されません (デフォルトは 0)。

結果セット

カラム名	データ型	説明
event_name	CHAR(256)	トレースイベントの名前を返します。システムトレースイベントの名前には、SYS_ というプレフィクスが含まれます。
field_name	CHAR(256)	フィールド名を返します。
field_id	INTEGER	トレースイベントのフィールドの順序を返します。
field_domain	INTEGER	フィールドのドメイン ID を返します。SYSDOMAIN.domain_id に対する外部キーです。
field_description	LONG VARCHAR	フィールドの内容の説明を返します。

備考

event_name が NULL の場合、このプロシージャはすべてのトレースイベントのフィールドを返します。

権限

MANAGE ANY TRACE SESSION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。include_audit_events が設定されていない場合、MANAGE AUDITING システム権限も必要です。

関連する動作

なし。

例

次の文は、データベースのすべてのトレースイベントセッションのフィールドに関する情報を返します。

```
SELECT * FROM sp_trace_event_fields( );
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.6.8.150 sp_trace_event_session_events システムプロシージャ

特定のトレースイベントセッションの一部であるトレースイベントをリストします。

構文

```
sp_trace_event_session_events(
    [ event_name
    [, include_server_sessions
    [, include_audit_events ] ] ]
  )
```

パラメータ

session_name

オプションの CHAR(256) パラメータを使用して、トレースイベントセッションの名前を指定します。デフォルトは NULL です。セッション名が指定されていないか、NULL である場合は、すべてのトレースイベントセッションに関して情報が返されます。

include_server_sessions 任意の BIT パラメータを使用して、エンジンレベルトレースイベントセッションを返すかどうかを指定します。このパラメータは、0 (エンジンレベルトレースイベントセッションを返さない) または 1 (エンジンレベルトレースイベントセッションを返す) に設定できます。デフォルトでは、エンジンレベルトレースイベントセッションは返されません (デフォルトは 0)。

include_audit_events 任意の BIT パラメータを使用して、監査イベントを返すかどうかを指定します。このパラメータは、0 (監査イベントを返さない) または 1 (監査イベントを返す) に設定できます。デフォルトでは、監査イベントは返されません (デフォルトは 0)。

結果セット

カラム名	データ型	説明
session_name	CHAR(256)	トレースイベントセッションの名前を返します。
event_name	CHAR(256)	トレースイベントの名前を返します。

備考

このシステムプロシージャを使用して、トレースイベントセッションとしてログを作成するトレースイベント (システム定義とユーザ定義の両方) を決定します。

権限

MANAGE ANY TRACE SESSION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。include_audit_events が設定されていない場合、MANAGE AUDITING システム権限も必要です。

関連する動作

なし。

例

この分は、現在のデータベースのすべてのイベントトレースセッションの一部であるイベントに関する情報を返します。

```
SELECT * FROM sp_trace_event_session_events( );
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.6.8.151 sp_trace_event_session_target_options システムプロシージャ

トレースイベントセッションのターゲットオプションをリストします。

構文

```
sp_trace_event_session_target_options(
    [ session_name
    [, include_server_sessions
    [, include_audit_events ] ] ]
  )
```

パラメータ

session_name

オプションの CHAR(256) パラメータを使用して、トレースイベントセッションの名前を指定します。デフォルトは NULL です。セッション名が指定されていないか、NULL である場合は、すべてのトレースイベントセッションに関して情報が返されます。

include_server_sessions 任意の BIT パラメータを使用して、エンジンレベルトレースイベントセッションを返すかどうかを指定します。このパラメータは、0 (エンジンレベルトレースイベントセッションを返さない) または 1 (エンジンレベルトレースイベントセッションを返す) に設定できます。デフォルトでは、エンジンレベルトレースイベントセッションは返されません (デフォルトは 0)。

include_audit_events 任意の BIT パラメータを使用して、監査イベントを返すかどうかを指定します。このパラメータは、0 (監査イベントを返さない) または 1 (監査イベントを返す) に設定できます。デフォルトでは、監査イベントは返されません (デフォルトは 0)。

結果セット

カラム名	データ型	説明
session_name	CHAR(256)	セッション名を返します。
target_type	CHAR(256)	ターゲットタイプ (FILE など) を返します。
option_name	CHAR(256)	オプション名を返します。
option_value	LONG VARCHAR	オプションの値を返します。

備考

このプロシージャは、現在のデータベースの 1 つまたはすべてのトレースイベントセッションのオプションに関する情報を返します。

権限

MANAGE ANY TRACE SESSION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。include_audit_events が設定されていない場合、MANAGE AUDITING システム権限も必要です。

関連する動作

なし。

例

次の文は、データベース内のすべてのトレースイベントセッションのターゲットオプションに関する情報を返します。

```
SELECT * FROM sp_trace_event_session_target_options( );
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]

1.6.8.152 sp_trace_event_session_targets システムプロシージャ

トレースセッションのターゲットをリストします。

構文

```
sp_trace_event_session_targets(
    [ session_name
    [, include_server_sessions
    [, include_audit_events ] ] ]
)
```

パラメータ

session_name

オプションの CHAR(256) パラメータを使用して、トレースイベントセッションの名前を指定します。デフォルトは NULL です。セッション名が指定されていない場合または NULL の場合、データベースのすべてのトレースイベントセッションに関する情報が返されます。
include_server_sessions エンジンレベルトレースイベントセッションを返すかどうかを指定する任意の BIT パラメータ。このパラメータは、0 (エンジンレベルトレースイベントセッションを返さない) または 1 (エンジンレベルトレースイベントセッションを返す) に設定できます。デフォルトでは、エンジンレベルトレースイベントセッションは返されません (デフォルトは 0)。
include_audit_events 監査イベントを返すかどうかを指定する任意の BIT パラメータ。このパラメータは、0 (監査イベントを返さない) または 1 (監査イベントを返す) に設定できます。デフォルトでは、監査イベントは返されません (デフォルトは 0)。

結果セット

カラム名	データ型	説明
session_name	CHAR(256)	セッション名を返します。
target_type	CHAR(256)	ターゲットタイプ (ファイルなど) を返します。

備考

ターゲットは、トレースイベントセッション情報のログ (データベースサーバメッセージログなど) が作成される場所となります。

権限

MANAGE ANY TRACE SESSION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です
include_audit_events が設定されていない場合、MANAGE AUDITING システム権限も必要です。

関連する動作

なし。

例

次の文は、データベース内のすべてのトレースイベントセッションに関する情報を返します。

```
SELECT * FROM sp_trace_event_session_targets( );
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]

sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.6.8.153 sp_trace_event_sessions システムプロシージャ

データベースで定義されているトレースイベントセッションのリストを返します。

構文

```
sp_trace_event_sessions( [ session_name ] )
```

パラメータ

session_name

この CHAR(256) パラメータを使用して、情報を取得するトレースイベントセッションを指定します。セッション名が指定されていない場合、データベース内のすべてのトレースイベントセッションに関する情報が返されます。

結果セット

カラム名	データ型	説明
session_name	CHAR(256)	セッションの名前を返します。
description	LONG VARCHAR	セッションの説明を返します。
started	BIT	セッションが開始されている場合は 1 を、それ以外の場合は 0 を返します。
is_temporary	BIT	トレースセッションがテンポラリの場合は 1 を、それ以外の場合は 0 を返します。

備考

ALTER TRACE EVENT SESSION 文の STATE 句を使用して、トレースイベントセッションを手動で開始および停止します。

権限

MANAGE ANY TRACE SESSION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次の文は、現在のデータベースのトレースイベントセッションのリストを返します。

```
SELECT * FROM sp_trace_event_sessions( );
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_events システムプロシージャ [1747 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.6.8.154 sp_trace_events システムプロシージャ

データベース内のトレースイベントに関する情報を返します。

構文

```
sp_trace_events(
    [ event_name
    [, include_audit_events ] ]
)
```

パラメータ

event_name

任意の CHAR(256) パラメータを使用して、トレースイベント名を指定します。デフォルトは NULL です。

include_audit_events 任意の BIT パラメータを使用して、監査イベントを返すかどうかを指定します。このパラメータは、0 (監査イベントを返さない) または 1 (監査イベントを返す) に設定できます。デフォルトでは、監査イベントは返されません (デフォルトは 0)。

結果セット

カラム名	データ型	説明
event_name	CHAR(256)	トレースイベントの名前を返します。システムトレースイベントの名前には、SYS_ というプレフィクスが含まれます。
description	LONG VARCHAR	トレースイベントがキャプチャする内容の説明を返します。
severity	TINYINT	トレースイベントの重要度レベルを返します。
is_system	BIT	システムトレースイベントの場合は 1 を、それ以外の場合は 0 を返します。
is_temporary	BIT	テンポラリトレースイベントの場合は 1 を、それ以外の場合は 0 を返します。

備考

このプロシージャは、データベース内のシステム定義とユーザ定義の両方のトレースイベントに関する情報を返します。`event_name` が指定されていない場合または NULL の場合、指定されたトレーオスイベントまたはすべてのトレースイベントの詳細を返します。

レベル	重要度の値の範囲
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

権限

MANAGE ANY TRACE SESSION システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。include_audit_events が設定されていない場合、MANAGE AUDITING システム権限も必要です。

関連する動作

なし。

例

次の文は、データベース内のユーザ定義のトレースイベントセッションのリストを返します。

```
SELECT * FROM sp_trace_events( )
WHERE is_system = 0;
```

関連情報

CREATE TEMPORARY TRACE EVENT 文 [969 ページ]
CREATE TEMPORARY TRACE EVENT SESSION 文 [971 ページ]
ALTER TRACE EVENT SESSION 文 [728 ページ]
DROP TRACE EVENT 文 [1079 ページ]
DROP TRACE EVENT SESSION 文 [1081 ページ]
NOTIFY TRACE EVENT 文 [1217 ページ]
sp_trace_event_fields システムプロシージャ [1739 ページ]
sp_trace_event_sessions システムプロシージャ [1746 ページ]
sp_trace_event_session_events システムプロシージャ [1740 ページ]
sp_trace_event_session_targets システムプロシージャ [1744 ページ]
sp_trace_event_session_target_options システムプロシージャ [1742 ページ]

1.6.8.155 sp_tsql_environment システムプロシージャ

ユーザが jConnect または Open Client アプリケーションから接続するときの接続オプションを設定します。

構文

```
sp_tsql_environment( )
```

備考

sp_login_environment プロシージャは、login_procedure データベースオプションによって指定されるデフォルトのプロシージャです。新規接続ごとに、login_procedure で指定されたプロシージャが呼び出されます。接続に TDS 通信プロトコルを使用する場合 (つまり、Open Client または jConnect 接続の場合)、今度は sp_login_environment が sp_tsql_environment を呼び出します。

このプロシージャは、デフォルトの Adaptive Server Enterprise の動作との互換性を持つように、データベースオプションを設定します。

新しいプロシージャを作成し login_procedure オプションを変更してそれらの新しいプロシージャをポイントして、デフォルトの動作を変更します。

次は、sp_tsql_environment プロシージャが設定したオプションのリストです。

```
if db_property( 'IQStore' ) = 'Off' then
    -- SQL Anywhere datastore
    SET TEMPORARY OPTION close_on_endtrans='OFF';
end if;
SET TEMPORARY OPTION ansinull='OFF';
SET TEMPORARY OPTION tsql_variables='ON';
SET TEMPORARY OPTION ansi_blanks='ON';
SET TEMPORARY OPTION chained='OFF';
SET TEMPORARY OPTION quoted_identifier='OFF';
SET TEMPORARY OPTION allow_nulls_by_default='OFF';
SET TEMPORARY OPTION on_tsql_error='CONTINUE';
SET TEMPORARY OPTION isolation_level='1';
SET TEMPORARY OPTION date_format='YYYY-MM-DD';
SET TEMPORARY OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';
SET TEMPORARY OPTION time_format='HH:NN:SS.SSS';
SET TEMPORARY OPTION date_order='MDY';
SET TEMPORARY OPTION escape_character='OFF';
```

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし

例

次は、sp_tsql_environment プロシージャを呼び出す例です。

```
CALL sp_tsql_environment();
```

関連情報

sp_login_environment システムプロシージャ [1694 ページ]

1.6.8.156 sp_use_secure_feature_key システムプロシージャ

現在の接続で指定されている保護された機能キーに含まれる機能を有効にします。

構文

```
sp_use_secure_feature_key(
name
, auth_key
)
```

パラメータ

name

有効にするセキュリティ保護済み機能キーの VARCHAR (128) 名。

auth_key

有効化されているセキュリティ保護済み機能キーに対する、大文字と小文字を区別する CHAR (128) 認証コード。認証コードは 6 文字以上である必要があります。

備考

このプロシージャは、現在の接続のみで指定されているセキュリティ保護済み機能キーに含まれるセキュリティ保護済み機能を有効にします。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例が機能するには、サーバが次のオプションで起動されていることが必要です。`-sk securefkey`。

次の例は、現在の接続の MANAGE_KEYS を含む SYSTEM セキュリティ保護済み機能キーを有効にします。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
```

次の例では、MANAGE_KEYS を含むセキュリティ保護済み SYSTEM 機能キーを有効にして、大文字小文字が区別される SecureMySet 認証コードを持つセキュリティ保護済み機能キー MYSET を新たに作成し、その新しいセキュリティ保護済み機能キーを使用します。

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'myset', 'SecureMySet', 'local,remote' );
CALL sp_use_secure_feature_key( 'myset', 'SecureMySet' );
```

新たなセキュリティ保護済み機能キーに切り替えたら、現在の接続はセキュリティ保護済み SYSTEM 機能を使用しなくなります。したがって、MANAGE_KEYS へのアクセスも解除されます。

関連情報

sp_alter_secure_feature_key システムプロシージャ [1657 ページ]
sp_create_secure_feature_key システムプロシージャ [1664 ページ]
sp_drop_secure_feature_key システムプロシージャ [1675 ページ]
sp_list_secure_feature_key システムプロシージャ [1693 ページ]

1.6.8.157 st_geometry_dump システムプロシージャ

ジオメトリを最低レベルのコンポーネントジオメトリに分解します。

構文

```
st_geometry_dump(
geometry
[, "options" ]
)
```

パラメータ

geometry

分解するジオメトリの ST_Geometry 値

"options"

セミコロンで区切ったパラメータと値の VARCHAR(255) 文字列。プロシージャの出力を設定するために使用できます。デフォルトは NULL です。

次の表は、指定できるパラメータのリストです。

パラメータ	デフォルト値	指定可能な値	説明
Format	*Original*	*Original*、*Internal*、または *Mixed*	ジオメトリを返すフォーマット。Original を指定すると、ジオメトリは元のフォーマットで返されます。Internal を指定すると、ジオメトリは正規化フォーマットで返されます。Mixed を指定すると、使用可能な格納フォーマットがどのようなものであっても、1 つのフォーマットにつき 1 ローが返されます。
ExpandPoints	*Yes*	*Yes*、*No*	デフォルトでは、ポイント (ST_LineString や ST_MultiPoint など) を含むジオメトリを分解すると、st_geometry_dump システムプロシージャは構成ポイントを別々のローに出力します。これらの余分なローが生成されないようにするには、ExpandPoints を No に設定します。
MaxDepth	-1	-1、0 以上の任意の数値	デフォルトでは、st_geometry_dump システムプロシージャは、リーフオブジェクトに達するまで、オブジェクト階層の分解を続けます。MaxDepth パラメータを設定して、階層内でのジオメトリの分解レベル数を制限できます。値を 0 に設定すると、ルートジオメトリのみが返されます。値を 1 に設定すると、ジオメトリとその直近の子が返されます。それ以降についても同様です。
SetGeom	*Yes*	*Yes, No*	st_geometry_dump システムプロシージャは、元の型階層内のオブジェクトに関連付けられている ST_Geometry のカラムを返します。このカラムが必要でない場合は、SetGeom パラメータを No に設定して、プロシージャの実行時間と出力サイズを減少させることができます。

パラメータ	デフォルト値	指定可能な値	説明
VALIDATE	*Basic*	*None*、*Basic*、*Full*	デフォルトでは、st_geometry_dump システムプロシージャは、ジオメトリのロード時にデータベースサーバで使用される検証ルールを適用し、ロー内のオブジェクトがこれらのルールに一致する場合は、結果セットの Valid カラムを 1 に設定します。Validate パラメータを None に設定すると、このチェックが無効になり、Full に設定すると、ST_IsValid メソッドによって実行される追加のチェックも適用されます。Full でのチェックは実行に時間がかかります。

結果セット

カラム	データ型	説明
id	UNSIGNED BIGINT	結果内のこのローのユニークな ID。
parent_id	UNSIGNED BIGINT	このオブジェクトの直近の親の ID。
depth	INTEGER	ルートオブジェクトから、このローに関連付けられているオブジェクトまでの深さ。
format	VARCHAR(128)	ジオメトリが元の表現 (Original) であるか、または正規化表現 (Internal) であるか。
valid	BIT	ジオメトリが有効 (1) かどうか (Validate オプションで指定したチェックレベルに基づく)。
geom_type	VARCHAR(128)	ジオメトリの型 (ST_GeometryType によって返される)。
geom	ST_Geometry	ジオメトリの仕様。SetGeom パラメータを No に設定すると、結果セットにジオメトリの仕様は返されません。
xmin	DOUBLE	ジオメトリの x の最小値。
xmax	DOUBLE	ジオメトリの x の最大値。
ymin	DOUBLE	ジオメトリの y の最小値。
ymax	DOUBLE	ジオメトリの y の最大値。
zmin	DOUBLE	ジオメトリの z の最小値。
zmax	DOUBLE	ジオメトリの z の最大値。
mmin	DOUBLE	ジオメトリの m の最小値。

カラム	データ型	説明
mmax	DOUBLE	ジオメトリの m の最大値。
details	LONG VARCHAR	ジオメトリに関する追加の詳細 (オブジェクトが有効でない理由に関する追加情報など)。

備考

st_geometry_dump システムプロシージャは、ジオメトリ階層を分解するときに、階層内のオブジェクト (ルートオブジェクトを含む) ごとに 1 つのローを使用します。階層内の各ジオメトリを検証して、各ジオメトリが有効かどうか、有効でない場合はその理由を調査できます。

st_geometry_dump システムプロシージャの一部の機能は、ST_GeometryN や ST_PointN などの型固有のメソッドを使用して照合できます。

st_geometry_dump システムプロシージャを使用して、無効なジオメトリを修正できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

関連する動作

なし。

例

次の例では、多角形 `'Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))'` をそのコンポーネントジオメトリに分解します。

```
SELECT * FROM st_geometry_dump( 'Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))',
'SetGeom=No' );
```

id	parent_id	depth	format	valid	geom_type	geom	xmin	xmax	ymin	ymax	...
1	1	0	Internal	1	ST_Polygon	`Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))`	0	3	0	3	...

id	parent_id	depth	format	valid	geom_type	geom	xmin	xmax	ymin	ymax	...
2	1	1	Internal	1	ST_LineString	LineString (0 0, 3 0, 3 3, 0 3, 0 0)	0	3	0	3	...
3	2	2	Internal	1	ST_Point	Point (0 0)	0	0	0	0	...
4	2	2	Internal	1	ST_Point	Point (3 0)	3	3	0	0	...
5	2	2	Internal	1	ST_Point	Point (3 3)	3	3	3	3	...
6	2	2	Internal	1	ST_Point	Point (0 3)	0	0	3	3	...
7	2	2	Internal	1	ST_Point	Point (0 0)	0	0	0	0	...

次の例は、st_geometry_dump システムプロシージャを使用して、ジオメトリ内の無効なポイントを見つける方法を示します。この例では、LineString に経度 1200 のポイントが含まれています。このため、結果ではそのポイントと LineString の両方が無効 (valid=0) であるとレポートされています。

```
SET TEMPORARY OPTION st_geometry_on_invalid='Ignore';
CREATE OR REPLACE VARIABLE @geo ST_Geometry;
SET @geo = new ST_LineString( 'LineString(1200 2, 80 10)', 4326 );
SELECT * FROM st_geometry_dump( @geo, 'SetGeom=No' );
```

id	parent_id	depth	format	valid	geom_type	geom	xmin	xmax	ymin	ymax	...	details
1	1	0	Original	0	ST_LineString	(NULL)	80	1,200	2	10	...	値 1200.000000 は座標系 longitude の範囲外です (SRS の境界 [-180.000000, 180.000000] を 50% 以上超過)

id	parent_id	depth	format	valid	geom_type	geom	xmin	xmax	ymin	ymax	...	details
2	1	1	Original	0	ST_LineString	(NULL)	1,200	1,200	2	2	...	値 1200.000000 は座標系 longitude の範囲外です (SRS の境界 [-180.000000, 180.000000] を 50% 以上超過)
3	1	1	Original	1	ST_Point	(NULL)	80	80	10	10	...	

無効なデータが識別されたら、st_geometry_dump システムプロシージャを他の空間メソッドとともに使用して、無効な要素を修正して有効なジオメトリを組み立てることができます。次の例は、経度 1200 の無効なポイントを経度 120.0 に修正する方法を示します。

```
SET TEMPORARY OPTION st_geometry_on_invalid='Ignore';
CREATE OR REPLACE VARIABLE @geo ST_Geometry;
SET @geo = new ST_LineString( 'LineString(1200 2, 80 10)', 4326 );
SELECT ST_LineString::ST_LineStringAggr(
    new ST_Point( IF xmax = 1200 then 120.0 ELSE xmax ENDIF,
                  ymax, 4326 )  ORDER BY id )
FROM  st_geometry_dump( @geo )
WHERE geom_type='ST_Point';
```

1.6.8.158 st_geometry_load_shapefile システムプロシージャ

テーブルを作成し、そこに ESRI シェイプファイルをロードします。

構文

```
st_geometry_load_shapefile(
shp_filename
, srid
, table_name
[, table_owner
[, shp_encoding ] ]
)
```

パラメータ

shp_filename

この VARCHAR(512) パラメータを使用して、ESRI シェイプファイルのロケーションを指定します。ファイル名の拡張子は .shp にする必要があります。また、同じディレクトリにある同じベース名の .shx ファイルと .dbf ファイルを関連付ける必要があります。パスは、クライアントアプリケーションではなく、データベースサーバを基準にした相対パスを指定します。

srid

この INTEGER パラメータを使用して、シェイプファイルからのデータを関連付けるための SRID を指定します。ジオメトリを意味のあるものにするため、指定する SRID は、シェイプファイルで使用されている SRID と同一ではなくても、ジオメトリに対して適切なものにする必要があります。

table_name

この VARCHAR(128) カラムを使用して、シェイプファイルデータを保持するために作成するテーブル名を指定します。

table_owner

任意の VARCHAR(128) カラムを使用して、作成された新しいテーブルの所有者を指定します。デフォルトの所有者は現在のユーザです。

shp_encoding

任意の VARCHAR(50) パラメータを使用して、シェイプファイルを読み込むときに使用するエンコードを指定します。デフォルトのエンコードは ISO-8859-1 です。

備考

st_geometry_load_shapefile システムプロシージャは、最初に、ESRI シェイプファイルで見つかったカラム情報 (名前とデータ型) を使ってテーブル "table_owner"."table_name" を作成します。その後、シェイプファイルからのデータを新しいテーブルにロードします。

このプロシージャは、sa_describe_shapefile システムプロシージャと LOAD TABLE...FORMAT SHAPEFILE 文を利用します。

権限

次に示すように、他の権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

-gl オプションが DBA に設定されている場合

- table_owner である場合は、CREATE TABLE、CREATE ANY TABLE、または CREATE ANY OBJECT のシステム権限を持っていて、かつ、ALTER ANY TABLE、ALTER ANY OBJECT、LOAD ANY TABLE のいずれかのシステム権限を持っていることが必要です。
- table_owner でない場合は、CREATE ANY TABLE または CREATE ANY OBJECT のシステム権限を持っていて、かつ、ALTER ANY TABLE、ALTER ANY OBJECT、LOAD ANY TABLE のいずれかのシステム権限を持っていることが必要です。

-gl オプションが ALL に設定されている場合

- `table_owner` である場合、CREATE TABLE、CREATE ANY TABLE、CREATE ANY OBJECT のいずれかのシステム権限を持っていることが必要です。
- `table_owner` でない場合は、CREATE ANY TABLE または CREATE ANY OBJECT のシステム権限を持っていて、かつ、ALTER ANY TABLE、ALTER ANY OBJECT、LOAD ANY TABLE のいずれかのシステム権限を持っていることが必要です。

-gl オプションが NONE に設定されている場合、データのロードは失敗します。

関連する動作

ロードが失敗しても、作成したファイルはそのまま残ります。

例

次の文は、esri_load テーブルを作成し、架空のシェイプファイルである `c:¥esri¥shapefile.shp` のデータを作成したテーブルにロードし、ロードしたデータを SRID 1000004326 に関連付けます。

```
CALL st_geometry_load_shapefile( 'c:¥¥esri¥¥tgr36069trt00.shp', 1000004326,
'esri_load' );
```

関連情報

sa_describe_shapefile システムプロシージャ [1495 ページ]
LOAD TABLE 文 [1182 ページ]

1.6.8.159 xp_cmdshell システムプロシージャ

プロシージャからオペレーティングシステムコマンドを実行します。

構文

```
xp_cmdshell(
command
[, redir_output | 'no_output' ]
)
```

パラメータ

command

この VARCHAR(8000) パラメータを使用して、システムコマンドを指定します。デフォルトは NULL です。

redir_output

任意の CHAR(254) パラメータを使用して、出力をコマンドウィンドウに表示するかどうかを指定します。デフォルトの動作では、出力がコマンドウィンドウに表示されます。'no_output' を指定した場合、コマンドウィンドウには出力は表示されません。デフォルト値は ' ' です。

戻り値

この関数は INTEGER 終了コードを返します。

備考

xp_cmdshell は、システムコマンドを実行して、制御を元の環境に戻します。xp_cmdshell が返す値は、実行されたシェルプロセスからの終了コードです。子プロセスが起動したときにエラーが発生した場合、戻り値は 2 です。

2 番目のパラメータは、Windows オペレーティングシステムのコマンドラインアプリケーションにのみ影響します。UNIX の場合は、2 番目のパラメータの設定にかかわらず、コマンドウィンドウは表示されません。

sa_enable_auditing_type システムプロシージャと sa_disable_auditing_type システムプロシージャを使用して、xp_cmdshell システムプロシージャの監査を有効および無効にします (xp_cmdshell タイプを使用)。xp_cmdshell の監査が有効になると、xp_cmdshell プロシージャのすべての呼び出しが監査ログに記録されます (プロシージャを実行したユーザ、プロシージャに渡されたパラメータを含む)。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、現在のディレクトリにあるファイルをファイル `c:¥temp.txt` 内にリストします。

```
CALL xp_cmdshell( 'dir > c:¥¥temp.txt' );
```

次の文は、同じ処理を実行しますが、*[コマンド]* ウィンドウは表示されません。

```
CALL xp_cmdshell( 'dir > c:¥¥temp.txt', 'no_output' );
```

関連情報

CALL 文 [756 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.6.8.160 xp_get_mail_error_code システムプロシージャ

最後の SMTP エラーまたは MAPI エラーに関する情報を返します。

構文

```
xp_get_mail_error_code( )
```

戻り値

この関数は、SMTP エラーコードまたは MAPI エラーコードを表す INTEGER 値を返します。

備考

mail プロシージャ (xp_startmail、xp_startsmtp、xp_sendmail、xp_stopmail、xp_stopsmtp) の戻り値が -1 の場合、この関数を使用して SMTP または MAPI のエラーコードを取得します。

mail プロシージャの戻り値が 5、6、または 7 の場合、この関数を使用して最後のソケットエラーのエラー番号を取得します。

MAPI が使用されている場合、返される値は MAPI 関数のリターンコードとなります。SMTP が使用されている場合、返される値は SMTP エラーコード (xp_get_mail_error_text は SMTP エラーテキストを返します) またはエラー番号 (xp_get_mail_error_text は空の文字列を返します) のいずれかとなります。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です。

関連する動作

なし

例

この例では、最後の SMTP エラーコードまたは MAPI エラーコードを取得します。

```
SELECT xp_get_mail_error_code( )
```

この例では、SMTP を使用して、プレーンテキストのメッセージの送信を開始しています。

```
BEGIN
    DECLARE err_smtp INTEGER;
    DECLARE err_code INTEGER;
    DECLARE err_msg  LONG VARCHAR;
    SELECT xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' ) INTO
err_smtp;
    SELECT xp_get_mail_error_code( ), xp_get_mail_error_text( ) INTO err_code,
err_msg;
    SELECT err_smtp, err_code, err_msg;
END;
```

関連情報

MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_get_mail_error_text システムプロシージャ [1762 ページ]
xp_startmail システムプロシージャ [1775 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]

1.6.8.161 xp_get_mail_error_text システムプロシージャ

最後の SMTP エラーまたはステータスメッセージテキストを返します。

構文

```
xp_get_mail_error_text( )
```

戻り値

この関数は、SMTP または MAPI のエラーまたはステータスメッセージテキストを表す LONG VARCHAR 値を返します。使用可能なエラーテキストがない場合、空の文字列または NULL が返されます。

備考

いずれかの mail プロシージャ (xp_startmail、xp_startsmtp、xp_sendmail、xp_stopmail、xp_stopsmtp) のエラーまたはステータスメッセージテキストを取得する場合は、この関数を使用します。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です。

関連する動作

なし

例

この例では、最後の SMTP メッセージテキストまたは MAPI メッセージテキストを取得します。

```
SELECT xp_get_mail_error_text( )
```

この例では、SMTP を使用して、プレーンテキストのメッセージの送信を開始しています。

```
BEGIN
    DECLARE err_smtp INTEGER;
    DECLARE err_code INTEGER;
    DECLARE err_msg  LONG VARCHAR;
    SELECT xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' ) INTO
err_smtp;
    SELECT xp_get_mail_error_code( ), xp_get_mail_error_text( ) INTO err_code,
err_msg;
    SELECT err_smtp, err_code, err_msg;
END;
```

関連情報

MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_get_mail_error_code システムプロシージャ [1761 ページ]
xp_startmail システムプロシージャ [1775 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]

1.6.8.162 xp_getenv システムプロシージャ

環境変数の値を返します。

構文

```
xp_getenv( environment_variable )
```

パラメータ

environment_variable

この VARCHAR(8000) パラメータを使用して、環境変数の所有者を指定します。このパラメータは、データベースの大文字と小文字の区別に関係なく、Windows オペレーティングシステム上では大文字と小文字が区別されず、その他のすべてのオペレーティングシステム上では大文字と小文字が区別されます。デフォルト値は NULL です。

戻り値

この関数は LONG NVARCHAR 値を返します。

備考

指定されている環境変数が NULL または設定されていない場合、NULL が返されます。

権限

SERVER OPERATOR システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

接続に対して GETENV 機能が有効になっている必要があります (-sf サーバオプション)。

関連する動作

なし。

例

次の例は、xp_getenv システムプロシージャを使用して、環境変数 PATH の値を返します。

```
SELECT CAST( xp_getenv( 'PATH' ) AS LONG VARCHAR );
```

次の例は、xp_getenv と sa_split_list のシステムプロシージャを使用して、Windows 環境変数 PATH の値をリストとして返します。UNIX オペレーティングシステムでは、':' を区切り文字として使用します。

```
CALL sa_split_list( CAST( xp_getenv( 'PATH' ) AS LONG VARCHAR ), ';' );
```

次の例は、存在しないことを想定する環境変数 NONEXISTENT を使用します。したがって、クエリを実行すると NULL が返されます。

```
SELECT xp_getenv( 'NONEXISTENT' );
```

1.6.8.163 xp_msver システムプロシージャ

データベースサーバのバージョンと名前についての情報を取り出します。

構文

```
xp_msver( the_option )
```

パラメータ

the_option

この CHAR(254) パラメータを使用して、文字列を指定します。文字列 (string) は次のいずれかでなければなりません。文字列は文字列デリミタで囲みます。

引数	説明
ProductName	製品の名前を返します。ProductName はデフォルトの引数値です。
ProductVersion	バージョン番号とそれに続くビルド番号を返します。フォーマットは次のとおりです。 `17.0.4.1691`
CompanyName	会社の名前を返します。
FileDescription	製品名、オペレーティングシステム名の順に返します。

引数	説明
LegalCopyright	ソフトウェアの著作権を示す文字列を返します。
LegalTrademarks	ソフトウェアの商標を示す文字列を返します。

戻り値

この関数は CHAR(254) 値を返します。

備考

この関数は、製品、会社、バージョンなどの情報を返します。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、バージョンとオペレーティングシステムの説明を要求します。

```
SELECT xp_msver( 'ProductVersion') Version,
   xp_msver( 'FileDescription' ) Description;
```

次に、出力例を示します。Version の値は、システムによって異なります。

Version	Description
17.0.4.1691	SQL Anywhere Windows7

関連情報

システム関数 [215 ページ]

1.6.8.164 xp_read_file システムプロシージャ

ファイルを読み込み、そのファイルの内容を LONG BINARY 変数として返します。

構文

```
xp_read_file(
filename
[, lazy ]
)
```

パラメータ

filename

この LONG VARCHAR パラメータを使用して、内容を返す対象ファイルの名前を指定します。

lazy

この任意の INTEGER パラメータを指定し、その値が 0 以外の場合、ファイルの内容は要求されるまで読み込まれません。読み込みが行われるのは、LONG BINARY 値へのアクセス時のみです。また、その対象はファイル内の要求された部分のみです。デフォルトは 0 または non-lazy です。

戻り値

この関数は、名前付きのファイルの内容を LONG BINARY 値で返します。ファイルが存在しない場合、または読み込むことができない場合、NULL が返されます。

備考

`filename` には、データベースサーバの開始ディレクトリからの相対ファイル名を指定します。

この関数は、ファイルに保存されているドキュメントやイメージ全体をテーブルに挿入するときに役立ちます。ファイルが読み込めない場合は、関数は NULL を返します。

データファイルの文字セットが異なる場合は、CSCONVERT 関数を使用すると文字セットを変換できます。

また、CSCONVERT 関数を使用すると、xp_read_file システムプロシージャを使用するときに発生する文字セット変換の要件に対処できます。

ディスクサンドボックスが有効になっている場合、`filename` で参照されるファイルはアクセス可能な場所に存在している必要があります。

指定されたファイルが存在しないと、この関数は NULL を返します。

権限

READ FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、Products テーブルのカラム Photo にイメージを挿入します。

```
UPDATE Products
SET Photo=xp_read_file( 'c:¥¥sqlany¥¥scripts¥¥adata¥¥HoodedSweatshirt.jpg' )
WHERE Products.ID=600;
```

次の文は、テキストファイルを読み込み、行番号とともに各行を表示します。

```
SELECT * FROM sa_split_list( CAST( xp_read_file( '¥¥Windows¥¥win.ini' ) AS LONG
VARCHAR ), 0x0a );
```

関連情報

CSCONVERT 関数 [文字列] [297 ページ]
xp_write_file システムプロシージャ [1782 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]
CALL 文 [756 ページ]

1.6.8.165 xp_scanf システムプロシージャ

入力文字列とフォーマット文字列から部分文字列を抽出します。

構文

```
xp_scanf(
input_buffer
, format
[ , param1 [, param2 ... ] ]
)
```

パラメータ

input_buffer

この CHAR(254) パラメータを使用して、入力文字列を指定します。

format

この CHAR(254) パラメータを使用して、各 param 引数に対する入力文字列のフォーマットをプレースホルダ (%s) を使用して指定します。最高で 50 のプレースホルダを format 引数に指定できます。また、param 引数と同じプレースホルダ数にする必要があります。%s 文字列フォーマットのみがサポートされます。

> **注記**
>
> %d や %f など、他のフォーマット指定子はサポートされないため、検出された場合、入力文字列のスキャンが停止されます。

param1, param2, ...

これらの CHAR(254) パラメータの中から 1 つ以上のパラメータを使用して、input_buffer から抽出する部分文字列を格納します。最大で 50 のパラメータを指定できます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

備考

> **例**
>
> 次の文は、部分文字列の Hello と World! を入力バッファの Hello World! から抽出します。部分文字列は変数 string1 と string2 に入力された後、選択されます。

```
CREATE VARIABLE string1 CHAR( 254 );
CREATE VARIABLE string2 CHAR( 254 );
CALL xp_scanf( 'Hello World!', '%s %s', string1, string2 );
SELECT string1, string2;
```

次の文は、日付文字列を取得し、年、月、日のコンポーネントに分割する方法を示しています。

```
CREATE VARIABLE ymd LONG VARCHAR;
CREATE VARIABLE year CHAR( 254 );
CREATE VARIABLE month CHAR( 254 );
CREATE VARIABLE day CHAR( 254 );
SET ymd = '2014/11/23'
SET ymd = REPLACE(ymd,'/', ' /');
CALL xp_scanf( ymd, '%s /%s /%s', year, month, day );
SELECT ymd, year, month, day;
```

関連情報

CALL 文 [756 ページ]

1.6.8.166 xp_sendmail システムプロシージャ

xp_startmail または xp_startsmtp でセッションが開始されると、指定された受信者に電子メールメッセージを送信します。このプロシージャには、任意の長さのメッセージを指定できます。

構文

```
xp_sendmail(
  recipient = mail-address
  [, subject = subject ]
  [, cc_recipient = mail-address ]
  [, bcc_recipient = mail-address ]
  [, query = sql-query ]
  [, "message" = message-body ]
  [, attachname = attach-name ]
  [, attach_result = attach-result ]
  [, echo_error = echo-error ]
  [, include_file = filename ]
  [, no_column_header = no-column-header ]
  [, no_output = no-output ]
  [, width = width ]
  [, separator = separator-char ]
  [, dbuser = user-name ]
  [, dbname = db-name ]
  [, type = type ]
  [, include_query = include-query ]
  [, content_type = content-type ]
)
```

パラメータ

一部の引数は固定値を指定し、下記のように Transact-SQL との互換性を保つために使用できます。

recipient

LONG VARCHAR パラメータは、受信者のメールアドレスを指定します。複数の受信者を指定するときは、各電子メールアドレスをセミコロンで区切ります。

subject

LONG VARCHAR パラメータは、メッセージの件名フィールドを指定します。デフォルトは NULL です。

cc_recipient

LONG VARCHAR パラメータは、cc 受信者のメールアドレスを指定します。複数の cc 受信者を指定するときは、各電子メールアドレスをセミコロンで区切ります。デフォルトは NULL です。

bcc_recipient

LONG VARCHAR パラメータは、bcc 受信者のメールアドレスを指定します。複数の bcc 受信者を指定するときは、各電子メールアドレスをセミコロンで区切ります。デフォルトは NULL です。

query

この LONG VARCHAR は Transact-SQL との互換性のために提供されています。SQL Anywhere では使用されません。デフォルトは NULL です。

"message"

LONG VARCHAR パラメータは、メッセージの内容を指定します。デフォルトは NULL です。"message" は予約語であるため、"message" パラメータ名は二重引用符で囲む必要があります。

attachname

この LONG VARCHAR パラメータは Transact-SQL との互換性のために提供されています。SQL Anywhere では使用されません。デフォルトは NULL です。

attach_result

この INTEGER パラメータは Transact-SQL との互換性を保つ目的で提供されています。SQL Anywhere では使用されません。デフォルトは 0 です。

echo_error

この INTEGER パラメータは Transact-SQL との互換性を保つ目的で提供されています。SQL Anywhere では使用されません。デフォルトは 1 です。

include_file

LONG VARCHAR パラメータは、添付ファイルを指定します。デフォルトは NULL です。

no_column_header

この INTEGER パラメータは Transact-SQL との互換性を保つ目的で提供されています。SQL Anywhere では使用されません。デフォルトは 0 です。

no_output

この INTEGER パラメータは Transact-SQL との互換性を保つ目的で提供されています。SQL Anywhere では使用されません。デフォルトは 0 です。

width

この INTEGER パラメータは Transact-SQL との互換性を保つ目的で提供されています。SQL Anywhere では使用されません。デフォルトは 80 です。

separator

この CHAR(1) パラメータは Transact-SQL との互換性のために提供されています。SQL Anywhere では使用されません。デフォルトは CHAR(9) です。

dbuser

この LONG VARCHAR パラメータは Transact-SQL との互換性のために提供されています。SQL Anywhere では使用されません。デフォルトは guest です。

dbname

この LONG VARCHAR パラメータは Transact-SQL との互換性のために提供されています。SQL Anywhere では使用されません。デフォルトは master です。

type

この LONG VARCHAR パラメータは Transact-SQL との互換性のために提供されています。SQL Anywhere では使用されません。デフォルトは NULL です。

include_query

この INTEGER パラメータは Transact-SQL との互換性を保つ目的で提供されています。SQL Anywhere では使用されません。デフォルトは 0 です。

content_type

この LONG VARCHAR パラメータは、"message" パラメータのコンテンツタイプを指定します (たとえば、text/html、ASIS など)。デフォルトは NULL です。content_type の値は検証されないため、無効なコンテンツタイプを設定すると、無効な電子メールや理解できない電子メールが送信されます。

content_type パラメータを ASIS に設定してヘッダを手動で設定します。この操作を行った場合、xp_sendmail プロシージャは、"message" パラメータに渡されたデータが正しく構成されたヘッダ付き電子メールであると想定して、ヘッダをさらに追加することはしません。ASIS を指定した場合は、データを他のパラメータに渡すことで通常なら自動的に指定されるヘッダも含めて、すべてのヘッダを "message" パラメータに手動で設定する必要があります。

戻り値

この関数は INTEGER ステータスコードを返します。

備考

xp_sendmail の引数値は文字列です。各引数の長さは、システムで使用可能なメモリ容量によって制限されます。

content_type 引数は、MIME 電子メールの要件を理解しているユーザが使用します。xp_sendmail では、ASIS が content_type として受け入れられます。content_type が ASIS に設定されると、xp_sendmail はメッセージ本文 ("message") が正しく構成されたヘッダ付き電子メールであると想定し、ヘッダをさらに追加することはしません。ASIS を指定して、複数のコンテンツタイプを含むマルチパートメッセージを送信できます。

include_file パラメータで指定された添付ファイルは application/octet-stream MIME タイプとして base64 エンコードで送信され、データベースサーバに作成されます。

SMTP 電子メールシステムで送信される電子メールは、件名行に 7 ビット ASCII ではない文字が含まれていると、エンコードされます。また、SMS 対応デバイスに送信された電子メールは、件名行に 7 ビット ASCII ではない文字が含まれていると、正しくデコードされないことがあります。

MAPI で電子メールセッションを開始するには xp_startmail を、SMTP で電子メールセッションを開始するには xp_startsmtp を実行する必要があります。

MAPI を使用して電子メールを送信する場合、content_type パラメータはサポートされません。

`message-body` に、998 文字を超える行が含まれている場合、SMTP サーバが ! 文字に加えて改行文字を電子メールの本文に挿入することがあります。このような余分な文字が挿入されないようにするため、`message-body` には 998 文字を超える行を含めないようにしてください。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です

例

この例では、SMTP を使用してプレーンテキストメッセージを送信します。

```
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='This is my subject line',
                  "message"='This text is the body of my email.¥n' );
CALL xp_stopsmtp( );
```

この例では、SMTP を使用して HTML フォーマットのメッセージと添付ファイルを送信します。

```
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='HTML mail example with attachment',
                  "message"='Plain text.<BR><BR><B>Bold text.</B><BR><BR>' ||
                            '<a href="www.sap.com">SAP Home Page</a>',
                  content_type = 'text/html',
                  include_file = '¥¥temp¥¥sendmail2.sql' );
CALL xp_stopsmtp( );
```

この例では、SMTP を使用してインライン HTML フォーマットのメッセージと添付ファイルを送信します。

```
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='Inline HTML mail example with attachment',
                  "message"='Content-Type: text/html;¥nContent-Disposition:
inline; ¥n¥n' ||
                            'Plain text.<BR><BR><B>Bold text.</B><BR><BR>' ||
                            '<a href="www.sap.com">SAP Home Page</a>',
                  content_type = 'ASIS',
                  include_file = '¥¥temp¥¥sendmail3.sql' );
CALL xp_stopsmtp( );
```

この例では、SMTP を使用してインライン HTML フォーマットのメッセージと、署名および 1 つが ZIP ファイルである 2 つの添付ファイルを送信します。

```
BEGIN
DECLARE content LONG VARCHAR;
SET content =
'Content-Type: multipart/mixed; boundary="xxxxx";¥n' ||
'This part of the email should not be shown.  If this ' ||
'is shown then the email client is not MIME compatible¥n¥n' ||
'--xxxxx¥n' ||
'Content-Type: text/html;¥n' ||
'Content-Disposition: inline;¥n¥n' ||
'Plain text.<BR><BR><B>Bold text.</B><BR><BR>' ||
'<a href="www.sap.com">SAP Home Page</a>¥n¥n' ||
xp_read_file( '¥¥temp¥¥johndoe.sig.html' ) ||
'--xxxxx¥n' ||
'Content-Type: application/zip; name="sendmail4.zip"¥n' ||
'Content-Transfer-Encoding: base64¥n' ||
'Content-Disposition: attachment; filename="sendmail4.zip"¥n¥n' ||
base64_encode( xp_read_file( '¥¥temp¥¥sendmail4.zip' ) ) ||
'¥n¥n' ||
'--xxxxx--¥n';
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='Inline HTML mail example with signature and 2
attachments',
                  "message"=content,
                  content_type = 'ASIS',
                  include_file = '¥¥temp¥¥sendmail4.sql' );
```

```
CALL xp_stopsmtp( );
END
```

関連情報

MAPI と SMTP のシステムプロシージャ [1432 ページ]
予約語 [6 ページ]
MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_startmail システムプロシージャ [1775 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]
CALL 文 [756 ページ]

1.6.8.167 xp_sprintf システムプロシージャ

入力文字列セットから結果文字列を構築します。

構文

```
xp_sprintf(
buffer
, format
[ , param1 [, param2 ... ] ]
)
```

パラメータ

buffer

これは、フォーマット済みの結果が挿入される CHAR(254) OUT パラメータです。

format

この CHAR(254) パラメータを使用して、各 `param` 引数に対する結果文字列のフォーマットをプレースホルダ (%s) を使用して指定します。最高で 50 のプレースホルダを `format` 引数に指定できます。また、`param` 引数と同じプレースホルダ数にする必要があります。%s 文字列形式のみがサポートされます。

param1, param2

結果文字列で使用される入力文字列。最高で 50 の CHAR(254) 引数を指定できます。

備考

出力パラメータに配置された結果は、254 文字に切り捨てられます。

権限

そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の文は、文字列 Hello World! を結果の変数に挿入します。

```
CREATE VARIABLE result CHAR( 254 );
CALL xp_sprintf( result, '%s %s', 'Hello', 'World!' );
SELECT result;
```

次の文は、年、月、日を日付文字列にフォーマットします。

```
CREATE VARIABLE result CHAR( 254 );
CALL xp_sprintf( result, '%s/%s/%s', 2014, 11, 23 );
SELECT result;
```

関連情報

CALL 文 [756 ページ]

1.6.8.168 xp_startmail システムプロシージャ

MAPI で電子メールセッションを開始します。

構文

```
xp_startmail(
[ mail_user = mail-login-name
[, mail_password = mail-password ]  ]
)
```

パラメータ

mail_user

この LONG VARCHAR パラメータを使用して、MAPI ログイン名を指定します。デフォルトは NULL です。

mail_password

この LONG VARCHAR パラメータを使用して、MAPI パスワードを指定します。デフォルトは NULL です。

戻り値

この関数は INTEGER ステータスコードを返します。

備考

xp_startmail は、電子メールセッションを開始するシステムプロシージャです。

Microsoft Exchange を使用する場合、`mail-login-name` 引数は Exchange のプロファイル名を指定します。また、プロシージャの呼び出しにパスワードを使用しないでください。

UNIX ではサポートされません。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です

関連情報

MAPI と SMTP のシステムプロシージャ [1432 ページ]
MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]
CALL 文 [756 ページ]

1.6.8.169 xp_startsmtp システムプロシージャ

SMTP で電子メールセッションを開始します。

構文

```
xp_startsmtp(
smtp_sender = email-address
, smtp_server = smtp-server
[, smtp_port = port-number ]
[, timeout = timeout ]
[, smtp_sender_name = username ]
[, smtp_auth_username = auth-username ]
[, smtp_auth_password = auth-password ]
[, trusted_certificates = { public-certificate | * }
[, secure = { 1 | 0 } ]
[, certificate_company = organization ]
[, certificate_unit = organization-unit ]
[, certificate_name = common-name ]
[, skip_certificate_name_check= { 1 | 0 } ]
)
```

パラメータ

smtp_sender

送信者の電子メールアドレスを指定する LONG VARCHAR パラメータです。

smtp_server

使用する SMTP サーバを指定する LONG VARCHAR パラメータです。SMTP サーバ名または IP アドレスで構成されます。

smtp_port

SMTP サーバで接続するポート番号を指定する任意の INTEGER パラメータです。デフォルトは 25 です。

timeout

データサーバからの応答を待機する秒数を指定する任意の INTEGER パラメータです。この秒数が経過すると、xp_sendmail の現在の呼び出しは中止します。デフォルトは 60 秒です。

smtp_sender_name

送信者の電子メールアドレスのエイリアスを指定する任意の LONG VARCHAR パラメータです。たとえば、`email-address` の代わりに JSmith などを指定します。デフォルトは NULL です。

smtp_auth_username

認証を必要とする SMTP サーバに示すユーザ名を指定する任意の LONG VARCHAR パラメータです。デフォルトは NULL です。

smtp_auth_password

認証を要求する SMTP サーバに提供するパスワードを指定する任意の LONG VARCHAR パラメータです。デフォルトは NULL です。

trusted_certificates

オプションの LONG VARCHAR パラメータは、セミコロンで区切られる "キーワード=値" ペアのリストです。デフォルトは NULL です。このパラメータが NULL の場合、標準 SMTP 接続が行われます。使用可能なキーを以下に示します。ファイル、証明書、および cert_name オプションのうち 1 つのみを指定できます。

信頼できる証明書は、サーバの自己署名証明書、パブリックルート証明書、民間認証局に属する証明書のいずれかです。RSA を使用して証明書を作成します。

オペレーティングシステムの証明書ストアの証明書を使用するには、**file=*** を指定します。

TLS 認証と暗号化を用いるセキュア SMTP (SMTPS) 接続をするには、**SMTPS=YES** と指定します。

単一のファイル名を指定することもできます (**trusted_certificates=**file-spec)。

期限切れまたはまだ有効でないルート証明書やデータベースサーバ証明書を受け入れるには、**allow_expired_certs=yes** と指定します。

キー	値
file=	信頼できる証明書を 1 つ以上含むファイルのパスとファイル名。
cert_name=	データベースに保存される証明書の名前。
certificate=	証明書データ。
SMTPS=	*YES* | *NO*
allow_expired_certs=	*YES* | *NO*

secure

このオプションパラメータは、接続がセキュアであるかどうかと、信頼できる証明書を使用するか、オペレーティングシステムの証明書ストアの証明書を使用するかを指定します。デフォルトは NULL です。

表 5: 結果

	secure=NULL	secure=0	secure=1
trusted_certificate=NULL	非セキュア	非セキュア	セキュア。オペレーティングシステム (OS) の証明書ストアを使用します。
trusted_certificate='*'	セキュア。オペレーティングシステム (OS) の証明書ストアを使用します。	エラーを返します。	セキュア。オペレーティングシステム (OS) の証明書ストアを使用します。
trusted_certificate=filename	セキュア。指定された証明書を使用します。	エラーを返します。	セキュア。指定された証明書を使用します。

certificate_company

証明書の組織フィールドがこの値と一致する場合にのみクライアントでサーバ証明書が受け入れられるように指定する任意の LONG VARCHAR パラメータです。trusted_certificates 値が NULL の場合、このパラメータは無視されます。デフォルトは NULL です。

certificate_unit

証明書の組織単位フィールドがこの値と一致する場合にのみクライアントでサーバ証明書が受け入れられるように指定する任意の LONG VARCHAR パラメータです。

certificate_name

証明書の共通名フィールドがこの値と一致する場合にのみクライアントでサーバ証明書が受け入れられるように指定する任意の LONG VARCHAR パラメータです。trusted_certificates 値が NULL の場合、このパラメータは無視されます。デフォルトは NULL です。

skip_certificate_name_check

このオプションの BIT パラメータは、SMTP サーバのホストが SMTP サーバ証明書に対してチェックされるかどうかを制御します。1 を指定するとこのオプションが有効になります。デフォルトは 0 です。このパラメータは、trusted_certificates が NULL のとき、または certificate_company、certificate_unit、または certificate_name といったいずれかのパラメータが指定されているとき、無視されます。

> **注記**
>
> この設定を使用すると、データベースサーバが SMTP サーバが十分に認証されなくなるため、この証明書を 1 に設定することは推奨されません。

戻り値

この機能は INTEGER ステータスコード (SMTP/MAPI リターンコード) を返します。

備考

xp_startsmtp は、SMTP サーバに接続し、指定された電子メールアドレスのメールセッションを開始するシステムプロシージャです。この接続はタイムアウトになります。xp_sendmail を実行する直前に、xp_startsmtp を呼び出す必要があります。

データベースは PLAIN 認証だけでなく、CRAM-MD5 認証もサポートします。smtp_auth_username と smtp_auth_password パラメータとともに xp_startsmtp システムプロシージャを使用する場合、データベースサーバは CRAM-MD5 認証を使用します。SMTP サーバが CRAM-MD5 認証をサポートしない場合、データベースサーバは PLAIN 認証を使用します。SMTP サーバが SMTP 認証機能をサポートしない場合、エラーコード 104 (サーバエラー。応答が理解されませんでした) が返されます。また、-fips データベースサーバオプションを使用してデータベースサーバが開始されると、PLAIN 認証のみが使用されます。

CRAM-MD5 認証は PLAIN 認証より安全ですが、どちらも SMTP サーバへの送信内容を暗号化しません。メールメッセージを含め、SMTP サーバへの送信内容を暗号化するには、セキュア SMTP を使用します。セキュア SMTP は TLS 暗号化機能による暗号化を行うほか、CRAM-MD5 または PLAIN 認証とともに使用できます。

ウィルススキャナは xp_startsmtp に影響を与え、エラーコード 100 を返す可能性があります。McAfee VirusScan バージョン 8.0.0 以降には、電子メールのワームの大量送信を防止するとともに、xp_sendmail の正しい実行を妨げる設定があります。ウィルススキャンソフトウェアで、大量メール送信の保護を回避する処理を指定できる場合、*dbeng17*.exe と *dbsrv17*.exe を指定します。たとえば、McAfee VirusScan では、この 2 つの処理を *Properties* 領域の *Excluded Processes* のリストに追加して、大量メール送信を許可できます。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です

関連情報

MAPI と SMTP のシステムプロシージャ [1432 ページ]
MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_startmail システムプロシージャ [1775 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]
CALL 文 [756 ページ]
CREATE CERTIFICATE 文 [779 ページ]

1.6.8.170 xp_stopmail システムプロシージャ

MAPI 電子メールセッションを閉じます。

構文

```
xp_stopmail( )
```

戻り値

この関数は INTEGER ステータスコードを返します。

備考

xp_stopmail は、電子メールセッションを終了するシステムプロシージャです。

UNIX ではサポートされません。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です

関連情報

MAPI と SMTP のシステムプロシージャ [1432 ページ]
MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_startmail システムプロシージャ [1775 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
xp_stopsmtp システムプロシージャ [1781 ページ]
CALL 文 [756 ページ]

1.6.8.171 xp_stopsmtp システムプロシージャ

SMTP 電子メールセッションを閉じます。

構文

```
xp_stopsmtp( )
```

戻り値

この関数は INTEGER ステータスコードを返します。

備考

xp_stopsmtp は、電子メールセッションを終了するシステムプロシージャです。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する SEND EMAIL 権限が必要です

関連情報

MAPI と SMTP のシステムプロシージャ [1432 ページ]
MAPI システムプロシージャと SMTP システムプロシージャのリターンコード [1434 ページ]
xp_startmail システムプロシージャ [1775 ページ]
xp_stopmail システムプロシージャ [1780 ページ]
xp_sendmail システムプロシージャ [1770 ページ]
xp_startsmtp システムプロシージャ [1777 ページ]
CALL 文 [756 ページ]

1.6.8.172 xp_write_file システムプロシージャ

SQL 文からファイルにデータを書き込みます。

構文

```
xp_write_file(
filename
, file_contents
)
```

パラメータ

filename

この LONG VARCHAR パラメータを使用して、ファイル名を指定します。

file_contents

この LONG BINARY パラメータを使用して、ファイルに書き込む内容を指定します。

戻り値

この関数は INTEGER ステータスコードを返します。

備考

この関数は、`file_contents` をファイル `filename` に書き込みます。成功した場合は 0 を返し、失敗した場合は 0 以外の値を返します。

filename 値には、絶対パスまたは相対パスのプレフィクスを付けることができます。filename に相対パスでプレフィクスを付けた場合、ファイル名はデータベースサーバの作業ディレクトリを基準にします。ファイルがすでに存在する場合は、内容が上書きされます。

この関数は、長いバイナリのデータをファイルにアンロードする場合に便利です。

また、CSCONVERT 関数を使用すると、xp_write_file システムプロシージャを使用するときに発生する文字セット変換の要件に対処できます。

ディスクサンドボックスが有効になっている場合、filename で参照されるファイルはアクセス可能な場所に存在している必要があります。

権限

WRITE FILE システム権限に加え、そのシステムプロシージャに対する EXECUTE 権限が必要です。

例

次の例では、xp_write_file を使用して、データ 123456 を含むファイル accountnum.txt を作成します。

```
CALL xp_write_file( 'accountnum.txt', '123456' );
```

次の例では、サンプルデータベースの Contacts テーブルに問い合わせて、ニュージャージー在住の各連絡先用にテキストファイルを作成します。各テキスト ファイルには、連絡先の姓 (Surname) と名 (GivenName) を連結した名前が付けられ (たとえば Reeves_Scott.txt)、連絡先の住所 (Street)、市 (City)、州 (State) が別々の行に含まれます。

```
SELECT xp_write_file(
Surname || '_' ||  GivenName || '.txt',
Street || '¥n' || City || '¥n' || State )
FROM Contacts WHERE State = 'NJ';
```

次の例では、xp_write_file を使用して、Products テーブル内の全製品について画像ファイル (JPG) を作成します。ID カラムの各値は、Photo カラムの対応する値のコンテンツが含まれるファイルのファイル名になります。

```
SELECT xp_write_file( ID || '.jpg', Photo ) FROM Products;
```

上記の例では、ID は UNIQUE 制約が指定されたローになります。これは、ファイルが以降のローのコンテンツで上書きされないようにするために重要です。また、カラムに格納されているデータに適用できるファイルの拡張子を指定する必要があります。この場合は、Products.Photo に画像データ (JPG) を格納します。

関連情報

CSCONVERT 関数 [文字列] [297 ページ]
xp_read_file システムプロシージャ [1767 ページ]
ディレクトリとファイルのシステムプロシージャ [1434 ページ]

1.7 ビュー

このソフトウェアでは、複数のビューの種類がサポートされています。

このセクションの内容:

1.7.1 システムビュー

カタログには、キーとインデックスの両方からリンクされるシステムテーブルが含まれます。システムテーブルは非表示です。ただし、各テーブルにはシステムビューがあります。一般的に必要とされるジョインを満たすために、システムビューに複数のシステムテーブルのカラムが含まれることもあります。

将来的なカタログとの互換性を保つため、アプリケーションではシステムビューを使用し、変更の可能性がある基本となるシステムテーブルは使用しないでください。

SQL Anywhere では、システム所有者 (ユーザ SYS) がシステムテーブルを所有します。

このセクションの内容:

SYSCAPABILITY システムビュー [1794 ページ]
SYSCAPABILITY システムビューの各ローは、リモートデータベースサーバの機能のステータスを示します。このビューの基本となるシステムテーブルは ISYSCAPABILITY です。

SYSCAPABILITYNAME システムビュー [1794 ページ]
SYSCAPABILITYNAME システムビューの各ローは、SYSCAPABILITY システムビューの各機能 ID の名前を示します。

SYSCERTIFICATE システムビュー [1795 ページ]
SYSCERTIFICATE システムビューの各ローには、テキスト PEM フォーマットの証明書が格納されています。このビューの基本となるシステムテーブルは ISYSCERTIFICATE です。

SYSCHECK システムビュー [1795 ページ]
SYSCHECK システムビューの各ローは、テーブル内の個々の名前付き検査制約の定義です。このビューの基本となるシステムテーブルは ISYSCHECK です。

SYSCOLPERM システムビュー [1795 ページ]
GRANT 文を使うと、テーブルの各カラムに UPDATE、SELECT、または REFERENCES 権限を与えることができます。UPDATE、SELECT、または REFERENCES 権限を持つ各カラムは、SYSCOLPERM システムビューの各ローに記録されます。このビューの基本となるシステムテーブルは ISYSCOLPERM です。

SYSCOLSTAT システムビュー [1796 ページ]
SYSCOLSTAT システムビューには、オプティマイザが使用するカラム統計 (ヒストグラムを含む) が入っています。このビューの内容を取り出すには、sa_get_histogram ストアドプロシージャまたはヒストグラムユーティリティを使用するのが最適です。このビューの基本となるシステムテーブルは ISYSCOLSTAT です。

SYSCONSTRAINT システムビュー [1797 ページ]
SYSCONSTRAINT システムビューの各ローは、そのデータベース内の名前付き制約を記述します。このビューの基本となるシステムテーブルは ISYSCONSTRAINT です。

SYSDBFILE システムビュー [1797 ページ]
SYSDBFILE システムビューの各ローは DB 領域ファイルを記述します。このビューの基本となるシステムテーブルは ISYSDBFILE です。

SYSDBSPACE システムビュー [1798 ページ]
SYSDBSPACE システムビューの各ローは、個々の DB 領域ファイルを記述します。このビューの基本となるシステムテーブルは ISYSDBSPACE です。

SYSDBSPACEPERM システムビュー [1799 ページ]
SYSDBSPACEPERM システムビューの各ローは、DB 領域ファイルの権限に関する記述です。このビューの基本となるシステムテーブルは ISYSDBSPACEPERM です。

SYSDEPENDENCY システムビュー [1799 ページ]
SYSDEPENDENCY システムビューの各ローは、2 つのデータベースオブジェクト間の依存性を示します。このビューの基本となるシステムテーブルは ISYSDEPENDENCY です。

SYSDOMAIN システムビュー [1800 ページ]
SYSDOMAIN システムビューは、組み込みデータ型に関する情報 (ドメインとも呼びます) を記録します。通常の操作ではビューのコンテンツは変化しません。このビューの基本となるシステムテーブルは ISYSDOMAIN です。

SYSEVENT システムビュー [1800 ページ]
SYSEVENT システムビュー内の各ローは、CREATE EVENT で作成された個々のイベントを記述します。このビューの基本となるシステムテーブルは ISYSEVENT です。

SYSEVENTTYPE システムビュー [1802 ページ]

SYSEVENTTYPE システムビューは、CREATE EVENT で参照できるシステムのイベントタイプを定義します。

SYSEXTERNENV システムビュー [1802 ページ]
SYSEXTERNENV システムビューの各ローは、各外部環境を識別して起動するときに必要な情報を記述します。このビューの基本となるシステムテーブルは ISYSEXTERNENV です。

SYSEXTERNENVOBJECT システムビュー [1804 ページ]
SYSEXTERNENVOBJECT システムビューの各ローは、インストールされている個々の外部オブジェクトを記述します。このビューの基本となるシステムテーブルは ISYSEXTERNENVOBJECT です。

SYSEXTERNLOGIN システムビュー [1805 ページ]
SYSEXTERNLOGIN システムビューの各ローは、リモートデータアクセスの個々の外部ログインを記述します。このビューの基本となるシステムテーブルは ISYSEXTERNLOGIN です。

SYSEXTERNLOGINPASSWORD システムビュー [1805 ページ]
SYSEXTERNLOGINPASSWORD システムビューの各ローは、外部ログインのオブジェクト ID とパスワードハッシュ値を示します。このビューの基本となるシステムテーブルは ISYSEXTERNLOGIN です。

SYSFKEY システムビュー [1806 ページ]
SYSFKEY システムビューの各ローは、そのシステム内の個々の外部キー制約を記述します。このビューの基本となるシステムテーブルは ISYSFKEY です。

SYSHISTORY システムビュー [1808 ページ]
SYSHISTORY システムビューの各ローは、データベースの開始、データベースの調整など、データベースに対するシステム操作を記録します。このビューの基本となるシステムテーブルは ISYSHISTORY です。

SYSIDX システムビュー [1810 ページ]
SYSIDX システムビューの各ローはデータベースの個々の論理インデックスを定義します。このビューの基本となるシステムテーブルは ISYSIDX です。

SYSIDXCOL システムビュー [1811 ページ]
SYSIDXCOL システムビューの各ローは、SYSIDX システムビューで記述されているインデックスのカラム 1 つを示します。このビューの基本となるシステムテーブルは ISYSIDXCOL です。

SYSJAR システムビュー [1812 ページ]
SYSJAR システムビューの各ローはデータベースに保存されている JAR ファイルを定義します。このビューの基本となるシステムテーブルは ISYSJAR です。

SYSJARCOMPONENT システムビュー [1813 ページ]
SYSJARCOMPONENT システムビューの各ローは、クラスファイル、マニフェストファイル、およびその他の JAR リソースなどの JAR ファイルコンポーネントを定義します。このビューの基本となるシステムテーブルは ISYSJARCOMPONENT です。

SYSJAVACLASS システムビュー [1814 ページ]
SYSJAVACLASS システムビューの各ローは、データベースに格納されている Java クラス 1 つを示します。このビューの基本となるシステムテーブルは ISYSJAVACLASS です。

SYSLDAPSERVER システムビュー [1814 ページ]
SYSLDAPSERVER システムビューには、データベースで設定されている各 LDAP サーバ設定オブジェクトに対して 1 つのローが含まれます。このビューの基本となるシステムテーブルは ISYSLDAPSERVER です。

SYSLDAPSERVERPASSWORD システムビュー [1816 ページ]
SYSLDAPSERVERPASSWORD システムビューの各ローは、LDAP アクセスアカウントのオブジェクト ID とパスワードハッシュ値を示します。このビューの基本となるシステムテーブルは ISYSLDAPSERVER です。

SYSLOGINMAP システムビュー [1816 ページ]
SYSLOGINMAP システムビューには、統合化ログインまたは Kerberos ログインを使用してデータベースに接続できる各ユーザに 1 つのローが含まれます。そのため、このビューへのアクセスは制限されています。このビューの基本となるシステムテーブルは ISYSLOGINMAP です。

SYSLOGINPOLICY システムビュー [1817 ページ]
このビューの基本となるシステムテーブルは ISYSLOGINPOLICY です。

SYSLOGINPOLICYOPTION システムビュー [1817 ページ]
このビューの基本となるシステムテーブルは ISYSLOGINPOLICYOPTION です。

SYSMIRROROPTION システムビュー [1817 ページ]
このビューの基本となるシステムテーブルは ISYSMIRROROPTION です。

SYSMIRRORSERVER システムビュー [1818 ページ]
このビューの基本となるシステムテーブルは ISYSMIRRORSERVER です。

SYSMIRRORSERVEROPTION システムビュー [1819 ページ]
このビューの基本となるシステムテーブルは ISYSMIRRORSERVEROPTION です。

SYSMUTEXSEMAPHORE システムビュー [1819 ページ]
SYSMUTEXSEMAPHORE システムビューの各ローは、データベースのユーザ定義ミューテックスおよびセマフォに関する情報を示します。このビューの基本となるシステムテーブルは ISYSMUTEXSEMAPHORE です。

SYSMVOPTION システムビュー [1820 ページ]
SYSMVOPTION システムビューの各ローは、マテリアライズドビューまたはテキストインデックスを作成した時点のオプション値に関する記述です。オプションの名前は、SYSMVOPTIONNAME システムビューにあります。このビューの基本となるシステムテーブルは ISYSMVOPTION です。

SYSMVOPTIONNAME システムビュー [1820 ページ]
SYSMVOPTION システムビューの各ローには、マテリアライズドビューまたはテキストインデックスを作成した時点の名前オプション値があります。オプションの値は、SYSMVOPTION システムビューにあります。このビューの基本となるシステムテーブルは ISYSMVOPTIONNAME です。

SYSOBJECT システムビュー [1821 ページ]
SYSOBJECT システムビューの各ローは個々のデータベースオブジェクトを記述します。このビューの基本となるシステムテーブルは ISYSOBJECT です。

SYSODATAPRODUCER システムビュー [1822 ページ]
SYSODATAPRODUCER システムビューの各ローは、OData プロデューサを示します。このビューの基本となるシステムテーブルは ISYSODATAPRODUCER です。

SYSOPTION システムビュー [1822 ページ]
SYSOPTION システムビューには、データベースに格納されている各オプション設定のローのオプションが含まれます。

SYSOPTSTAT システムビュー [1822 ページ]
SYSOPTSTAT システムビューには、ALTER DATABASE CALIBRATE 文によって計算されたコストモデル調整情報が入ります。このビューの内容は内部使用専用であり、sa_get_dtt システムプロシージャによってアクセスするのが最も良い方法です。このビューの基本となるシステムテーブルは ISYSOPTSTAT です。

SYSPHYSIDX システムビュー [1823 ページ]
SYSPHYSIDX システムビューの各ローはデータベースの個々の物理インデックスを定義します。このビューの基本となるシステムテーブルは ISYSPHYSIDX です。

SYSPROCEDURE システムビュー [1824 ページ]
SYSPROCEDURE システムビューの各ローは、そのデータベースに保存されている個々のプロシージャを記述します。このビューの基本となるシステムテーブルは ISYSPROCEDURE です。

SYSPROCPARM システムビュー [1825 ページ]
SYSPROCPARM システムビューの各ローは、データベース内のプロシージャまたは関数の 1 つのパラメータ、結果セットカラムまたは戻り値を示します。このビューの基本となるシステムテーブルは ISYSPROCPARM です。

SYSPROCPERM システムビュー [1827 ページ]
SYSPROCPERM システムビューの各ローは、プロシージャに対する EXECUTE 権限を付与されているユーザを表します。このビューの基本となるシステムテーブルは ISYSPROCPERM です。

SYSPROXYTAB システムビュー [1827 ページ]
SYSPROXYTAB システムビューの各ローは、プロキシテーブルのリモートパラメータを示します。このビューの基本となるシステムテーブルは ISYSPROXYTAB です。

SYSPUBLICATION システムビュー [1828 ページ]
syspublication システムテーブルの各ローは、パブリケーションを示します。このビューの基本となるシステムテーブルは ISYSPUBLICATION です。

SYSREMARK システムビュー [1829 ページ]
SYSREMARK システムビューの各ローは、オブジェクトの注釈 (コメント) を示します。このビューの基本となるシステムテーブルは ISYSREMARK です。

SYSREMOTEOPTION システムビュー [1829 ページ]
SYSREMOTEOPTION システムビューの各ローは、メッセージリンクパラメータの値を示します。このビューの基本となるシステムテーブルは ISYSREMOTEOPTION です。

SYSREMOTEOPTIONTYPE システムビュー [1829 ページ]
SYSREMOTEOPTIONTYPE システムビューの各ローは、個々のメッセージリンクパラメータを示します。このビューの基本となるシステムテーブルは ISYSREMOTEOPTIONTYPE です。

SYSREMOTETYPE システムビュー [1830 ページ]
SYSREMOTETYPE システムビューには、リモートテーブルに関する情報があります。このビューの基本となるシステムテーブルは ISYSREMOTETYPE です。

SYSREMOTEUSER システムビュー [1830 ページ]
SYSREMOTEUSER システムビューの各ローは、REMOTE システム権限を持つユーザ ID (サブスクライバ) を示します。そのユーザに送信されたメッセージと、そのユーザから送信されたメッセージのステータスも合わせて示します。このビューの基本となるシステムテーブルは ISYSREMOTEUSER です。

SYSROLEGRANT システムビュー [1831 ページ]
SYSROLEGRANT システムビューは、ロールのメンバーシップとメンバーシップのタイプに関する情報を格納します。このビューの基礎となるシステムテーブルは、ISYSROLEGRANT です。

SYSROLEGRANTEXT システムビュー [1832 ページ]
SET USER および CHANGE PASSWORD システム権限を付与するときは、被付与者がそれらのシステム権限を付与できるユーザまたはロールのリストを指定できます。

SYSSCHEDULE システムビュー [1833 ページ]
SYSSCHEDULE システムビューの各ローには、CREATE EVENT の SCHEDULE 句で指定された、イベントが発生する時刻が記述されています。このビューの基本となるシステムテーブルは ISYSSCHEDULE です。

SYSSEQUENCE システムビュー [1834 ページ]

SYSSEQUENCE システムビューには、各ユーザ定義シーケンスごとに 1 つのローがあります。このビューの基礎となるシステムテーブルは ISYSSEQUENCE です。

SYSSEQUENCEPERM システムビュー [1835 ページ]
SYSSEQUENCEPERM システムビューには、ユーザまたはグループがシーケンスに対して保持する権限が記録されます。このビューの基礎となるシステムテーブルは ISYSSEQUENCEPERM です。

SYSSERVER システムビュー [1835 ページ]
SYSSERVER システムビューの各ローはリモートサーバを記述します。このビューの基本となるシステムテーブルは ISYSSERVER です。

SYSSOURCE システムビュー [1836 ページ]
SYSSOURCE システムビューの各ローには、SYSOBJECT システムビューにリストされているオブジェクトのソースコード (適用できる場合) が含まれます。このビューの基本となるシステムテーブルは ISYSSOURCE です。

SYSSPATIALREFERENCESYSTEM システムビュー [1836 ページ]
SYSSPATIALREFERENCESYSTEM システムビューの各ローは、データベースに定義されている SRS に関する記述です。このビューの基礎となるシステムテーブルは ISYSSPATIALREFERENCESYSTEM です。

SYSSQLSERVERTYPE システムビュー [1839 ページ]
SYSSQLSERVERTYPE システムビューには、Adaptive Server Enterprise との互換性に関する情報が含まれます。このビューの基本となるシステムテーブルは ISYSSQLSERVERTYPE です。

SYSSUBSCRIPTION システムビュー [1839 ページ]
SYSSUBSCRIPTION システムビューの各ローは、REMOTE システム権限を持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションに関する記述です。このビューの基本となるシステムテーブルは ISYSSUBSCRIPTION です。

SYSSYNC システムビュー [1839 ページ]
SYSSYNC システムビューには、同期に関する情報が含まれます。

SYSSYNCPROFILE システムビュー [1841 ページ]
SYSSYNCPROFILE システムビューには、Mobile Link 同期の同期プロファイルに関する情報が含まれます。

SYSSYNCPROFILE2 システムビュー [1841 ページ]
SYSSYNCPROFILE2 システムビューには、Mobile Link 同期の同期プロファイルに関する情報が含まれます。

SYSSYNCSCRIPT システムビュー [1841 ページ]
SYSSYNCSCRIPT システムビューの各ローは、スクリプト化されたアップロードに関するストアドプロシージャを識別します。このビューは SYSSYNCSCRIPTS ビューとほとんど同じですが、このビューでは値が生の形式で表示されます。

SYSTAB システムビュー [1842 ページ]
SYSTAB システムビューの各ローは、データベース内の 1 つのテーブルまたはビューを示します。ビューの追加情報が SYSVIEW システムビューにあります。このビューの基本となるシステムテーブルは ISYSTAB です。

SYSTABCOL システムビュー [1845 ページ]
SYSTABCOL システムビューには、データベース内の各テーブルとビューに対して 1 つのローがあります。このビューの基本となるシステムテーブルは ISYSTABCOL です。

SYSTABLEPERM システムビュー [1847 ページ]
GRANT 文によってテーブルやビューに対して付与された権限は、SYSTABLEPERM システムビューに格納されます。このビューの各ローは 1 つのテーブル、権限を与えるユーザ ID (grantor)、そして権限を与えられるユーザ ID (grantee) に対応します。このビューの基本となるシステムテーブルは ISYSTABLEPERM です。

SYSTEXTCONFIG システムビュー [1849 ページ]
SYSTEXTCONFIG システムビューの各ローは、全文検索機能で使用するテキスト設定オブジェクト 1 つを示します。このビューの基本となるシステムテーブルは ISYSTEXTCONFIG です。

SYSTEXTIDX システムビュー [1850 ページ]
SYSTEXTIDX システムビューの各ローは、1 つのテキストインデックスを示します。このビューの基本となるシステムテーブルは ISYSTEXTIDX です。

SYSTEXTIDXTAB システムビュー [1851 ページ]
SYSTEXTIDXTAB システムビューの各ローは、テキストインデックスの一部である生成されたテーブルを示します。このビューの基本となるシステムテーブルは ISYSTEXTIDXTAB です。

SYSTIMEZONE システムビュー [1851 ページ]
SYSTIMEZONE システムビューの各ローは、1 つのタイムゾーンを示します。このビューの基本となるシステムテーブルは ISYSTIMEZONE です。

SYSTRIGGER システムビュー [1852 ページ]
SYSTRIGGER システムビューの各ローは、データベース内のトリガ 1 つを示します。このテーブルには、参照トリガアクションを持つ外部キー定義に、自動的に作成されるトリガも含まれます (たとえば、ON DELETE CASCADE)。このビューの基本となるシステムテーブルは ISYSTRIGGER です。

SYSTYPEMAP システムビュー [1855 ページ]
SYSTYPEMAP システムビューには、SYSSQLSERVERTYPE システムビューのエントリの互換性マッピング値があります。このビューの基本となるシステムテーブルは ISYSTYPEMAP です。

SYSUNITOFMEASURE システムビュー [1855 ページ]
SYSUNITOFMEASURE システムビューの各ローは、データベースに定義されている測定単位に関する記述です。SYSUNITOFMEASURE システムビューの基本となるテーブルは、ISYSUNITOFMEASURE システムテーブルです。

SYSUSER システムビュー [1855 ページ]
SYSUSER システムビューの各ローは、そのシステム内の個々のトリガを記述します。

SYSUSERPASSWORD システムビュー [1857 ページ]
SYSUSERPASSWORD システムビューの各ローは、ログインアカウントのオブジェクト ID とパスワードハッシュ値を示します。このビューの基本となるシステムテーブルは ISYSUSER です。

SYSUSERMESSAGE システムビュー [1858 ページ]
SYSUSERMESSAGE システムビューの各ローは、エラー条件に対する個々のユーザ定義メッセージです。このビューの基本となるシステムテーブルは ISYSUSERMESSAGE です。

SYSUSERTYPE システムビュー [1858 ページ]
SYSUSERTYPE システムビューの各ローは、ユーザ定義のデータ型の記述です。このビューの基本となるシステムテーブルは ISYSUSERTYPE です。

SYSDATABASEVARIABLE システムビュー [1859 ページ]
SYSDATABASEVARIABLE システムビューの各ローは、データベース内のデータベーススコープ変数 1 つを示します。このビューの基本となるシステムテーブルは ISYSDATABASEVARIABLE です。

SYSVIEW システムビュー [1860 ページ]
SYSVIEW システムビューの各ローは、データベース内のビューを示します。ビューの追加情報が SYSTAB システムビューにあります。このビューの基本となるシステムテーブルは ISYSVIEW です。

SYSWEBSERVICE システムビュー [1862 ページ]

SYSWEBSERVICE システムビューの各ローは、個々の Web サービスを記述します。このビューの基本となるシステムテーブルは ISYSWEBSERVICE です。

1.7.1.1 ビューと定義の詳細なシステム情報の表示

SQL Central から定義などのシステムビューに関する情報にアクセスします。

コンテキスト

チェックポイントが発生すると、システムビューが更新されます。

手順

1. *SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベースを右クリックし、**所有者フィルタの設定**をクリックします。
3. *SYS* をクリックし、*OK* をクリックします。
4. 左ウィンドウ枠で、**ビュー**をダブルクリックします。
5. 左ウィンドウ枠で SYS が所有するビュー (これはビュー名の後のカッコで囲まれたフレーズ "SYS" によって示されます) をクリックし、右ウィンドウ枠で *SQL* タブをクリックします。
6. **データ**タブをクリックします。

結果

ビュー定義は、**データ**タブに表示されます。

1.7.1.2 GTSYSPERFCACHEPLAN システムビュー

GTSYSPERFCACHEPLAN システムビュー内の各ローには、指定した文の実行プランに対するグラフィカルプラン文字列が含まれます。

カラム名	データ型	説明
stmt_hash	UNSIGNED BIGINT	各クエリに割り当てられた一意の識別子。

カラム名	データ型	説明
plan_hash	UNSIGNED BIGINT	特定の文に対する各プランに割り当てられた ID。
plan_text	XML NOT NULL	クエリ実行プランの XML 表現。

備考

システムビューには、1 つの文に対して複数の実行プランが表示されることがあります。ステートメントパフォーマンスサマリデータが収集されないと、文プランはレポートされません。

文の実行時間が短い (0.005 秒以下) 場合、プランは記録されません。

権限

このビューにアクセスするには、MONITOR システム権限が必要です。

関連情報

GTSYSPERFCACHESTMT システムビュー [1792 ページ]

1.7.1.3 GTSYSPERFCACHESTMT システムビュー

GTSYSPERFCACHESTMT システムビュー内の各ローは、定数が削除された文に対する SQL テキストを表します。

カラム名	データ型	説明
stmt_hash	UNSIGNED BIGINT	各クエリに割り当てられた一意の識別子。
stmt_text	LONG VARCHAR	文の SQL 表現。

備考

ステートメントパフォーマンスサマリ機能は、このビューに格納された SQL 文を使用します。

実行時間の短い文 (0.005 秒以下) に対する SQL は記録されません。

権限

このビューにアクセスするには、MONITOR システム権限が必要です。

関連情報

GTSYSPERFCACHEPLAN システムビュー [1791 ページ]

1.7.1.4 SYSARTICLE システムビュー

SYSARTICLE システムビューの各ローは、パブリケーション内の個々のアーティクルを記述します。このビューの基本となるシステムテーブルは ISYSARTICLE です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	このアーティクルが含まれるパブリケーション。
table_id	UNSIGNED INT	各アーティクルは、単一のテーブルのカラムとローから構成されます。このカラムには、このテーブルの ID が含まれます。
where_expr	LONG VARCHAR	WHERE 句で定義されたローのサブセットを含むアーティクルの場合、このカラムに探索条件が含まれます。
subscribe_by_expr	LONG VARCHAR	SUBSCRIBE BY 式で定義されたローのサブセットを含むアーティクルの場合、このカラムに式が含まれます。
query	CHAR(1)	データベースサーバにアーティクルタイプ情報を示します。
alias	VARCHAR(256)	アーティクルのエイリアス。
schema_change_active	BIT	テーブルとパブリケーションが同期スキーマ変更の一部である場合、1 です。

1.7.1.5 SYSARTICLECOL システムビュー

SYSARTICLECOL システムビューの各ローは、アーティクル内の個々のカラムを示します。このビューの基本となるシステムテーブルは ISYSARTICLECOL です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	カラムが含まれるパブリケーションのユニークな識別子。

カラム名	データ型	説明
table_id	UNSIGNED INT	カラムが属するテーブル。
column_id	UNSIGNED INT	SYSTABCOL システムビューのカラム識別子。

1.7.1.6 SYSCAPABILITY システムビュー

SYSCAPABILITY システムビューの各ローは、リモートデータベースサーバの機能のステータスを示します。このビューの基本となるシステムテーブルは ISYSCAPABILITY です。

カラム名	データ型	説明
capid	INTEGER	SYSCAPABILITYNAME システムビューに表示されている機能 ID。
srvid	UNSIGNED INT	SYSSERVER システムビューに表示されている、機能が適用されるサーバ。
capvalue	CHAR(128)	機能の値。

関連情報

SYSCAPABILITYNAME システムビュー [1794 ページ]

1.7.1.7 SYSCAPABILITYNAME システムビュー

SYSCAPABILITYNAME システムビューの各ローは、SYSCAPABILITY システムビューの各機能 ID の名前を示します。

カラム名	データ型	説明
capid	INTEGER	機能をユニークに識別する番号。
capname	VARCHAR(32000)	機能の名前。

備考

SYSCAPABILITYNAME システムビューは、sa_rowgenerator と次のサーバプロパティの組み合わせを使用して定義されます。

- RemoteCapability
- MaxRemoteCapability

関連情報

SYSCAPABILITY システムビュー [1794 ページ]

1.7.1.8 SYSCERTIFICATE システムビュー

SYSCERTIFICATE システムビューの各ローには、テキスト PEM フォーマットの証明書が格納されています。このビューの基本となるシステムテーブルは ISYSCERTIFICATE です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	証明書の ID。
cert_name	CHAR(128)	証明書の名前。
contents	LONG BINARY	圧縮形式の証明書の内容。
update_time	TIMESTAMP	最後に作成または置換を行ったローカル日時。
update_time_utc	TIMESTAMP WITH TIME ZONE	最後に作成または置換を行った UTC 日時。

1.7.1.9 SYSCHECK システムビュー

SYSCHECK システムビューの各ローは、テーブル内の個々の名前付き検査制約の定義です。このビューの基本となるシステムテーブルは ISYSCHECK です。

カラム名	データ型	説明
check_id	UNSIGNED INT	データベースの制約をユニークに識別する番号。
check_defn	LONG VARCHAR	CHECK 式。

1.7.1.10 SYSCOLPERM システムビュー

GRANT 文を使うと、テーブルの各カラムに UPDATE、SELECT、または REFERENCES 権限を与えることができます。UPDATE、SELECT、または REFERENCES 権限を持つ各カラムは、SYSCOLPERM システムビューの各ローに記録されます。このビューの基本となるシステムテーブルは ISYSCOLPERM です。

カラム名	データ型	説明
table_id	UNSIGNED INT	カラムが属するテーブルのテーブル番号。

カラム名	データ型	説明
grantee	UNSIGNED INT	カラムに権限を与えられたユーザの ID。ユーザ ID が PUBLIC ロールである場合、すべてのユーザがカラムに対する権限を持ちます。
grantor	UNSIGNED INT	権限を付与したユーザの ID。
column_id	UNSIGNED INT	このカラム番号と table_id を使って、権限を付与されたカラムを識別します。
privilege_type	SMALLINT	このカラムの数値は、カラム権限の種類 (16=REFERENCES、1=SELECT、または 8=UPDATE) を示します。
is_grantable	CHAR(1)	カラムの権限が GRANT OPTION を使って付与されているかどうかを示します。

1.7.1.11 SYSCOLSTAT システムビュー

SYSCOLSTAT システムビューには、オプティマイザが使用するカラム統計 (ヒストグラムを含む) が入っています。このビューの内容を取り出すには、sa_get_histogram ストアドプロシージャまたはヒストグラムユーティリティを使用するのが最適です。このビューの基本となるシステムテーブルは ISYSCOLSTAT です。

カラム名	データ型	説明
table_id	UNSIGNED INT	このカラムが属するテーブルまたはマテリアライズドビューをユニークに識別する番号。
column_id	UNSIGNED INT	table_id とともに使用してカラムをユニークに識別する番号。
format_id	SMALLINT	システムでのみ使用。
update_time	TIMESTAMP	このカラムの統計情報が最後に更新されたローカル時刻。
density	FLOAT	カラムの単一の値の平均による選択性の推定。ローに格納されている大きい単一値の選択性は考慮されません。
max_steps	SMALLINT	システムでのみ使用。
actual_steps	SMALLINT	システムでのみ使用。
step_values	LONG BINARY	システムでのみ使用。
frequencies	LONG BINARY	システムでのみ使用。
update_time_utc	TIMESTAMP WITH TIME ZONE	このカラムの統計情報が最後に更新された UTC 時刻。

注記

SQL Anywhere 16 以降を使用して作成されたデータベースでは、不正なアクセスからデータを保護するため、このビューの基本となるシステムテーブルは常に暗号化されます。

関連情報

sa_get_histogram システムプロシージャ [1515 ページ]

1.7.1.12 SYSCONSTRAINT システムビュー

SYSCONSTRAINT システムビューの各ローは、そのデータベース内の名前付き制約を記述します。このビューの基本となるシステムテーブルは ISYSCONSTRAINT です。

カラム名	データ型	説明
constraint_id	UNSIGNED INT	制約のユニークな ID。
constraint_type	CHAR(1)	制約の型。 **C** カラム検査制約 **T** テーブル制約 **P** プライマリキー **F** 外部キー **U** 一意性制約
ref_object_id	UNSIGNED BIGINT	制約の適用対象となるカラム、テーブル、インデックスのオブジェクト ID。
table_object_id	UNSIGNED BIGINT	制約の適用対象となるテーブルのオブジェクト ID。
constraint_name	CHAR(128)	制約名。

1.7.1.13 SYSDBFILE システムビュー

SYSDBFILE システムビューの各ローは DB 領域ファイルを記述します。このビューの基本となるシステムテーブルは ISYSDBFILE です。

カラム名	データ型	説明
dbfile_id	SMALLINT	内部でのみ使用。

カラム名	データ型	説明
dbspace_id	SMALLINT	データベースの各 DB 領域ファイルには、ユニークな番号が割り当てられています。システム DB 領域には、すべてのシステムオブジェクトが含まれ、0 の dbspace_id があります。
dbfile_name	CHAR(128)	DB 領域のユニークなファイル名。これは CREATE TABLE コマンド中で使われます。
file_name	LONG VARCHAR	DB 領域のファイル名。
lob_map	LONG VARBIT	内部でのみ使用。
server_id	UNSIGNED INT	内部でのみ使用。

1.7.1.14 SYSDBSPACE システムビュー

SYSDBSPACE システムビューの各ローは、個々の DB 領域ファイルを記述します。このビューの基本となるシステムテーブルは ISYSDBSPACE です。

カラム名	データ型	説明
dbspace_id	SMALLINT	DB 領域を識別するユニークな番号。システム DB 領域には、すべてのシステムオブジェクトが含まれ、0 の dbspace_id があります。
object_id	UNSIGNED BIGINT	DB 領域のオブジェクト ID。
dbspace_name	CHAR(128)	DB 領域のユニークなファイル名。これは CREATE TABLE コマンド中で使われます。
store_type	TINYINT	内部でのみ使用。
saved_cache_pages	LONG VARBIT	ALTER DATABASE SAVE CACHE 文が最後に実行されたときにキャッシュに存在した DB 領域のページ。文が実行されたことがない場合、値は NULL です。

1.7.1.15 SYSDBSPACEPERM システムビュー

SYSDBSPACEPERM システムビューの各ローは、DB 領域ファイルの権限に関する記述です。このビューの基本となるシステムテーブルは ISYSDBSPACEPERM です。

カラム名	データ型	説明
dbspace_id	SMALLINT	DB 領域を識別するユニークな番号。システム DB 領域には、すべてのシステムオブジェクトが含まれ、0 の dbspace_id があります。
grantee	UNSIGNED INT	権限を取得するユーザのユーザ ID。
privilege_type	SMALLINT	grantee に付与される権限。たとえば、CREATE は DB 領域でオブジェクトを作成する権限を grantee に付与します。

関連情報

GRANT 文 [1131 ページ]

1.7.1.16 SYSDEPENDENCY システムビュー

SYSDEPENDENCY システムビューの各ローは、2 つのデータベースオブジェクト間の依存性を示します。このビューの基本となるシステムテーブルは ISYSDEPENDENCY です。

あるオブジェクトが定義内の別のオブジェクトを参照する場合、この 2 つのデータベースオプション間に依存性が存在します。たとえば、ビューのクエリ指定がテーブルを参照する場合、ビューはテーブルに依存しています。データベースサーバが、テーブル、ビュー、マテリアライズドビュー、カラムの依存性を追跡します。

カラム名	データ型	説明
ref_object_id	UNSIGNED BIGINT	参照オブジェクトのオブジェクト ID。
dep_object_id	UNSIGNED BIGINT	参照オブジェクトの ID。

関連情報

sa_dependent_views システムプロシージャ [1487 ページ]

1.7.1.17 SYSDOMAIN システムビュー

SYSDOMAIN システムビューは、組み込みデータ型に関する情報 (ドメインとも呼びます) を記録します。通常の操作ではビューのコンテンツは変化しません。このビューの基本となるシステムテーブルは ISYSDOMAIN です。

カラム名	データ型	説明
domain_id	SMALLINT	各データ型に割り当てられたユニークな番号。この番号は変更できません。
domain_name	CHAR(128)	CREATE TABLE コマンド中に見られるデータ型の名前 (CHAR または INTEGER など)。
type_id	SMALLINT	ODBC データ型。この値は、Transact-SQL 互換の dbo.SYSTYPES テーブルにある data_type の値に対応します。
"precision"	SMALLINT	このデータ型を使って格納できる有効桁数。数値でないデータ型に対応するカラム値は NULL です。

1.7.1.18 SYSEVENT システムビュー

SYSEVENT システムビュー内の各ローは、CREATE EVENT で作成された個々のイベントを記述します。このビューの基本となるシステムテーブルは ISYSEVENT です。

カラム名	データ型	説明
event_id	UNSIGNED INT	各イベントに割り当てられたユニークな番号。
object_id	UNSIGNED BIGINT	データベースのイベントをユニークに識別するイベントの内部 ID。
creator	UNSIGNED INT	イベントの所有者のユーザ番号。ユーザ名は SYSUSER システムビューで確認できます。
event_name	VARCHAR(128)	イベント名。
enabled	CHAR(1)	イベントが起動できるかどうかを示します。

カラム名	データ型	説明
location	CHAR(1)	イベントを起動するロケーション。 • Y = AT ALL 句と FOR PRIMARY 句が指定される • E = AT CONSOLIDATED 句と FOR PRIMARY 句が指定される • T = AT REMOTE 句と FOR PRIMARY 句が指定される • P = (AT 句は指定されない) FOR PRIMARY 句が指定される • B = AT ALL 句と FOR ALL 句が指定される • D = AT CONSOLIDATED 句と FOR ALL 句が指定される • S = AT REMOTE 句と FOR ALL 句が指定される • M = (AT 句は指定されない) FOR ALL 句が指定される • C = AT CONSOLIDATED (FOR 句は指定されない) • R = AT REMOTE (FOR 句は指定されない) • A = AT ALL 句 (FOR 句は指定されない)
event_type_id	UNSIGNED INT	システムイベントの場合、イベント型は SYSEVENTTYPE システムビューにリストされます。
action	LONG VARCHAR	イベントハンドラ定義。難読化された値は、隠されたイベントを示します。
external_action	LONG VARCHAR	システムでのみ使用。
condition	LONG VARCHAR	イベントハンドラの起動を制御するのに使用される条件。
remarks	LONG VARCHAR	イベントの注釈。このカラムは ISYSREMARK に由来します。
source	LONG VARCHAR	イベントの元のソース。このカラムは ISYSSOURCE に由来します。

関連情報

SYSEVENTTYPE システムビュー [1802 ページ]

1.7.1.19 SYSEVENTTYPE システムビュー

SYSEVENTTYPE システムビューは、CREATE EVENT で参照できるシステムのイベントタイプを定義します。

カラム名	データ型	説明
event_type_id	INT	各イベント型に割り当てられたユニークな番号。
name	VARCHAR(32000)	システムイベント型の名前。
description	VARCHAR(32000)	システムイベント型の説明。

備考

SYSEVENTTYPE システムビューは、sa_rowgenerator と次のサーバプロパティの組み合わせを使用して定義されます。

- EventTypeName
- EventTypeDesc
- MaxEventType

関連情報

SYSEVENT システムビュー [1800 ページ]

1.7.1.20 SYSEXTERNENV システムビュー

SYSEXTERNENV システムビューの各ローは、各外部環境を識別して起動するときに必要な情報を記述します。このビューの基本となるシステムテーブルは ISYSEXTERNENV です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	外部環境のユニークな識別子。
name	CHAR(128)	このカラムは、外部環境または言語の名前を識別します。これは、*java*、*perl*、*php* などのいずれか 1 つです。

カラム名	データ型	説明
scope	CHAR(1)	このカラムは、CONNECTION の場合は C、DATABASE の場合は D のどちらかです。scope カラムは、外部環境が接続ごとに 1 つ起動されるのか、データベースごとに 1 つ起動されるのかを識別します。 接続ごとに 1 つ起動される外部環境 (PERL、PHP、JS、C_ESQL32、C_ESQL64、C_ODBC32、C_ODBC64 など) では、外部環境を使用する接続ごとに外部環境のインスタンスが 1 つあります。接続ごとの場合、外部環境は接続が切断されると終了します。 データベースごとに 1 つ起動される外部環境 (JAVA など) では、外部環境を使用するデータベースごとに外部環境のインスタンスが 1 つあります。データベースごとに 1 つ起動される外部環境は、データベースが停止されると終了します。
support_result_sets	CHAR(1)	このカラムは、結果セットを返すことのできる外部環境を識別します。PERL、PHP、および JS 以外のすべての外部環境が結果セットを返すことができます。
location	LONG VARCHAR	このカラムは、外部環境の実行ファイル/バイナリファイルが置かれているデータベースサーバコンピュータのロケーションを識別します。この句には、実行プログラム/バイナリの名前を指定します。このパスは、完全に修飾されたパスまたは相対パスのどちらでもかまいません。相対パスの場合、実行ファイル/バイナリファイルはデータベースサーバによって検索できるロケーションに置く必要があります。
options	LONG VARCHAR	このカラムは、外部環境に関連付けられている実行ファイルを起動するために、コマンドラインで指定する必要があるオプションを識別します。このカラムは変更しないでください。

カラム名	データ型	説明
user_id	UNSIGNED INT	外部環境が最初に起動すると、データベースへの接続を確立し、外部環境使用のための設定を実行します。デフォルトでは、この接続は DBA ユーザ ID を使用して作成されます。しかし、MANAGE ANY EXTERNAL OBJECT システム権限を持つ別のユーザ ID を外部環境で使用することをデータベース管理者が望む場合は、user_id カラムに別のユーザ ID を指定します。通常、このカラムは NULL であるため、データベースサーバはデフォルトで DBA ユーザ ID を使用します。

1.7.1.21 SYSEXTERNENVOBJECT システムビュー

SYSEXTERNENVOBJECT システムビューの各ローは、インストールされている個々の外部オブジェクトを記述します。このビューの基本となるシステムテーブルは ISYSEXTERNENVOBJECT です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	外部オブジェクトのユニークな識別子。
extenv_id	UNSIGNED BIGINT	外部環境のユニークな識別子 (SYSEXTERNENV.object_id)。
owner	UNSIGNED INT	このカラムは、外部オブジェクトの作成者/所有者を識別します。
name	LONG VARCHAR	このカラムは、INSTALL EXTERNAL OBJECT 文で指定されている外部オブジェクトの名前を識別します。
contents	LONG BINARY	外部オブジェクトの内容。
update_time	TIMESTAMP	このカラムは、オブジェクトが最後に変更 (またはインストール) されたローカル時刻を識別します。
update_time_utc	TIMESTAMP WITH TIME ZONE	このカラムは、オブジェクトが最後に変更 (またはインストール) された UTC 時刻を識別します。

1.7.1.22 SYSEXTERNLOGIN システムビュー

SYSEXTERNLOGIN システムビューの各ローは、リモートデータアクセスの個々の外部ログインを記述します。このビューの基本となるシステムテーブルは ISYSEXTERNLOGIN です。

カラム名	データ型	説明
user_id	UNSIGNED INT	ローカルデータベースでのユーザ ID。
srvid	UNSIGNED INT	SYSSERVER システムビューにリストされているリモートサーバ。
remote_login	VARCHAR(128)	このユーザのリモートサーバのログイン名。
remote_password	CHAR(3)	パスワードが格納されるかどうか。3 つのアスタリスク (***) は、パスワードが格納されることを示します。NULL は、パスワードが格納されないことを示します。パスワードのハッシュを取得するには、SYSEXTERNLOGINPASSWORD システムビューのクエリを実行します。

SYSEXTERNLOGINS システムテーブルに含まれる前のカタログバージョン。このテーブル名は ISYSEXTERNLOGIN ('S' なし) に変更され、このビューの基本となるテーブルになります。

関連情報

SYSEXTERNLOGINPASSWORD システムビュー [1805 ページ]

1.7.1.23 SYSEXTERNLOGINPASSWORD システムビュー

SYSEXTERNLOGINPASSWORD システムビューの各ローは、外部ログインのオブジェクト ID とパスワードハッシュ値を示します。このビューの基本となるシステムテーブルは ISYSEXTERNLOGIN です。

このビューにアクセスするには、SELECT ANY TABLE システム権限と ACCESS USER PASSWORD システム権限が必要です。

カラム	データ型	説明
user_id	UNSIGNED INT	ローカルデータベースでのユーザ ID。
srvid	UNSIGNED INT	SYSSERVER システムビューにリストされているリモートサーバ。
remote_password	VARBINARY(128)	外部ログインのパスワードハッシュ。

関連情報

SYSEXTERNLOGIN システムビュー [1805 ページ]

1.7.1.24 SYSFKEY システムビュー

SYSFKEY システムビューの各ローは、そのシステム内の個々の外部キー制約を記述します。このビューの基本となるシステムテーブルは ISYSFKEY です。

カラム名	データ型	説明
foreign_table_id	UNSIGNED INT	外部テーブルのテーブル番号。
foreign_index_id	UNSIGNED INT	外部キーのインデックス番号。
primary_table_id	UNSIGNED INT	プライマリテーブルのテーブル番号。
primary_index_id	UNSIGNED INT	プライマリキーのインデックス番号。
match_type	TINYINT	制約と一致する型。一致する型には次の型があります。 **0** デフォルトのマッチングを使用 **1** SIMPLE **2** FULL **129** SIMPLE UNIQUE **130** FULL UNIQUE 一致する型の詳細については、CREATE TABLE 文の MATCH 句を参照してください。
check_on_commit	CHAR(1)	COMMIT で外部キーが有効であるかどうかが確認されるまで、INSERT と UPDATE 文を待たせるかどうかを示します。待機する場合、値は 'Y' に、待機しない場合、値は 'N' になります。
nulls	CHAR(1)	外部キーのカラムが NULL 値を許容するかどうかを示します。この設定は外部キーのカラム中の nulls 設定とは独立しています。許容する場合、値は 'Y' に、許容しない場合、値は 'N' になります。

関連情報

CREATE TABLE 文 [952 ページ]

1.7.1.25 SYSHISTORY システムビュー

SYSHISTORY システムビューの各ローは、データベースの開始、データベースの調整など、データベースに対するシステム操作を記録します。このビューの基本となるシステムテーブルは ISYSHISTORY です。

カラム名	データ型	説明
operation	CHAR(128)	データベースファイルに対して実行されるオペレーションのタイプ。operation には次のいずれかの値を指定します。 **INIT** データベースがいつ作成されたかに関する情報。 **UPGRADE** データベースがいつアップグレードされたかに関する情報。 **START** 特定のオペレーティングシステム上で特定のバージョンのデータベースサーバを使用してデータベースがいつ開始されたかに関する情報。 **LAST_START** データベースサーバが最後に開始された時刻に関する情報。LAST_START オペレーションは、LAST_START ローに現在格納されている値とは異なるバージョンのデータベースサーバか異なるオペレーティングシステム、またはその両方でデータベースが開始されたときに START オペレーションに変換されます。 **DTT** DB 領域で実行される *2* 番目から最後のディスク転送時間 (DTT: Disk Transfer Time) の調整操作に関する情報。つまり、ALTER DATABASE CALIBRATE 文か ALTER DATABASE RESTORE DEFAULT CALIBRATION 文のどちらかの最後から 2 番目の実行に関する情報です。 **LAST_DTT** DB 領域で実行される最新の DTT 調整操作に関する情報。つまり、ALTER DATABASE CALIBRATE 文か

カラム名	データ型	説明
		ALTER DATABASE RESTORE DEFAULT CALIBRATION 文のどちらかの最新の実行に関する情報です。 **LAST_BACKUP** 最後のバックアップ (バックアップの日時を含みます)、バックアップの種類、バックアップするファイル、バックアップを実行したデータベースサーバのバージョンに関する情報。
object_id	UNSIGNED INT	DTT と LAST_DTT 以外の操作については、このカラムの値は 0 になります。DTT と LAST_DTT の操作の場合、SYSDBSPACE システムビューで定義されている DB 領域の dbspace_id です。
sub_operation	CHAR(128)	DTT と LAST_DTT 以外の操作については、このカラムの値は、空の単一引用符 ('') のセットになります。DTT と LAST_DTT の操作の場合、このカラムには、DB 領域で実行されるサブ操作の種類も含まれます。次のような値があります。 **DTT_SET** DB 領域の調整が設定されます。 **DTT_UNSET** DB 領域のデフォルト設定が復元されます。
version	CHAR(128)	オペレーションの実行に使用されるデータベースサーバのバージョンとビルド番号。
platform	CHAR(128)	オペレーションが実行されたオペレーティングシステム。
first_time	TIMESTAMP	特定のオペレーティングシステム上の特定のバージョンのソフトウェアでデータベースが最初に開始されたローカル日時。
last_time	TIMESTAMP	特定のオペレーティングシステム上の特定のバージョンのソフトウェアでデータベースが最後に開始されたローカル日時。
details	LONG VARCHAR	このカラムは、データベースサーバの起動に使用されたコマンドラインオプションやデータベースに対して有効になっている機能ビットなどの情報を格納します。この情報は、Sybase 製品の保守契約を結んでいるサポートセンタが使用します。

カラム名	データ型	説明
first_time_utc	TIMESTAMP WITH TIME ZONE	特定のオペレーティングシステム上の特定のバージョンのソフトウェアでデータベースが最初に開始された UTC 日時。
last_time_utc	TIMESTAMP WITH TIME ZONE	特定のオペレーティングシステム上の特定のバージョンのソフトウェアでデータベースが最後に開始された UTC 日時。

関連情報

SYSDBSPACE システムビュー [1798 ページ]

1.7.1.26 SYSIDX システムビュー

SYSIDX システムビューの各ローはデータベースの個々の論理インデックスを定義します。このビューの基本となるシステムテーブルは ISYSIDX です。

カラム名	データ型	説明
table_id	UNSIGNED INT	このインデックスが適用されるテーブルをユニークに識別します。
index_id	UNSIGNED INT	テーブル内のインデックスを識別するユニークな番号。
object_id	UNSIGNED BIGINT	データベースのインデックスをユニークに識別するインデックスの内部 ID。
phys_index_id	UNSIGNED INT	論理インデックスの実装に使用する基本となる物理インデックスを識別します。この値は、テンポラリテーブルまたはリモートテーブル上のインデックスに対しては NULL となります。その他の場合、この値は SYSPHYSIDX システムビューの物理インデックスの object_id に対応します。
dbspace_id	SMALLINT	インデックスを含むファイルの ID。この値は SYSDBSPACE システムビューのエントリに対応します。

カラム名	データ型	説明
index_category	TINYINT	インデックスの型。次のような値があります。 **1** プライマリキー **2** 外部キー **3** セカンダリインデックス (一意性制約を含む) **4** テキストインデックス
"unique"	TINYINT	インデックスがユニークインデックス (1) か、一意性制約 (2) か、予約済み (3) か、ユニークでないインデックス (4) か、またはユニークインデックス WITH NULLS NOT DISTINCT (1) かを示します。
index_name	CHAR(128)	インデックスの名前。
not_enforced	CHAR(1)	内部でのみ使用。
file_id	SMALLINT	廃止予定。このカラムは SYSVIEW に存在しますが、基本となるシステムテーブル ISYSIDX には存在しません。このカラムの内容は dbspace_id と同じであり、互換性のために提供されています。今後は dbspace_id を使用してください。

関連情報

SYSIDXCOL システムビュー [1811 ページ]
SYSPHYSIDX システムビュー [1823 ページ]
SYSDBSPACE システムビュー [1798 ページ]

1.7.1.27 SYSIDXCOL システムビュー

SYSIDXCOL システムビューの各ローは、SYSIDX システムビューで記述されているインデックスのカラム 1 つを示します。このビューの基本となるシステムテーブルは ISYSIDXCOL です。

カラム名	データ型	説明
table_id	UNSIGNED INT	インデックスが適用されるテーブルを識別します。

カラム名	データ型	説明
index_id	UNSIGNED INT	カラムが適用されるインデックスを識別します。table_id と index_id は共同で SYSIDX システムビューにある 1 つのインデックスを識別します。
sequence	SMALLINT	インデックス内の各カラムには、0 から始まるユニークな番号が割り当てられます。これらの番号の順序により、インデックス内のカラムの相対的な重要度が決まります。最も重要なカラムの sequence 番号は 0 になります。
column_id	UNSIGNED INT	インデックスを作成するテーブルのカラムを識別します。table_id と column_id は共同で、SYSCOLUMN システムビューで表される 1 つのカラムを識別します。
"order"	CHAR(1)	インデックス内のカラムが昇順 (A) か、降順 (D) かを示します。テキストインデックスの場合、この値は NULL です。
primary_column_id	UNSIGNED INT	この外部キーカラムに対応するプライマリキーの ID。非外部キーのカラムの場合、値は NULL です。

関連情報

SYSIDX システムビュー [1810 ページ]

1.7.1.28 SYSJAR システムビュー

SYSJAR システムビューの各ローはデータベースに保存されている JAR ファイルを定義します。このビューの基本となるシステムテーブルは ISYSJAR です。

カラム名	データ型	説明
jar_id	INTEGER	JAR ファイルを識別するユニークな番号。
object_id	UNSIGNED BIGINT	データベースのインデックスをユニークに識別する JAR ファイルの内部 ID。
creator	UNSIGNED INT	JAR ファイルの作成者のユーザ番号。INSTALL JAVA 文の AS USER 句で設定できます。
jar_name	LONG VARCHAR	JAR ファイルの名前。
jar_file	LONG VARCHAR	このカラムは使用されなくなり、値は NULL になります。

カラム名	データ型	説明
update_time	TIMESTAMP	JAR ファイルが最後に更新されたローカル時刻。
update_time_utc	TIMESTAMP WITH TIME ZONE	JAR ファイルが最後に更新された UTC 時刻。

関連情報

SYSJARCOMPONENT システムビュー [1813 ページ]
SYSJAVACLASS システムビュー [1814 ページ]

1.7.1.29 SYSJARCOMPONENT システムビュー

SYSJARCOMPONENT システムビューの各ローは、クラスファイル、マニフェストファイル、およびその他の JAR リソースなどの JAR ファイルコンポーネントを定義します。このビューの基本となるシステムテーブルは ISYSJARCOMPONENT です。

カラム名	データ型	説明
component_id	INTEGER	コンポーネントの ID を含むプライマリキー。
jar_id	INTEGER	ローが JAR を示す場合、値は JAR の ID 番号です。それ以外の場合、値は NULL です。
component_name	LONG VARCHAR	コンポーネント名。
component_type	CHAR(1)	このカラムは使用されなくなり、値は NULL になります。
contents	LONG BINARY	JAR ファイルのバイトコード。

関連情報

SYSJAR システムビュー [1812 ページ]
SYSJAVACLASS システムビュー [1814 ページ]

1.7.1.30 SYSJAVACLASS システムビュー

SYSJAVACLASS システムビューの各ローは、データベースに格納されている Java クラス 1 つを示します。このビューの基本となるシステムテーブルは ISYSJAVACLASS です。

カラム名	データ型	説明
class_id	INTEGER	Java クラスのユニークな番号。テーブルのプライマリキーでもあります。
object_id	UNSIGNED BIGINT	データベースのインデックスをユニークに識別する Java クラスの内部 ID。
creator	UNSIGNED INT	クラスの作成者のユーザ番号。INSTALL JAVA 文の AS USER 句で設定できます。
jar_id	INTEGER	クラスがある JAR ファイルの ID。
class_name	LONG VARCHAR	Java クラスの名前。
public	CHAR(1)	クラスがパブリック (Y) かプライベート (N) かを示します。
component_id	INTEGER	SYSJARCOMPONENT システムビューのコンポーネントの ID。
update_time	TIMESTAMP	クラスが最後に更新されたローカル時刻。
update_time_utc	TIMESTAMP WITH TIME ZONE	クラスが最後に更新された UTC 時刻。

関連情報

SYSJAR システムビュー [1812 ページ]
SYSJARCOMPONENT システムビュー [1813 ページ]

1.7.1.31 SYSLDAPSERVER システムビュー

SYSLDAPSERVER システムビューには、データベースで設定されている各 LDAP サーバ設定オブジェクトに対して 1 つのローが含まれます。このビューの基本となるシステムテーブルは ISYSLDAPSERVER です。

LDAP サーバ設定オブジェクトには、外部の LDAP サーバに接続するために必要な設定情報が含まれています。

カラム名	データ型	説明
ldsrv_id	UNSIGNED BIGINT	LDAP サーバのユニークな識別子。この ID は、ログインポリシーによってこの LDAP サーバを参照するために使用されます。このカラムは ISYSOBJECT に対する外部キーです。
ldsrv_name	CHAR(128)	LDAP サーバの名前。

カラム名	データ型	説明
ldsrv_state	CHAR(9)	LDAP サーバのステータス。有効な値は次のとおりです。 • RESET • READY • ACTIVE • FAILED • SUSPENDED
ldsrv_start_tls	TINYINT	有効な値は次のとおりです。1 (ON) または 0 (OFF)。ON の場合は、LDAP over TLS を使用して LDAP サーバに接続します。このプロトコルは、LDAP サーバとの接続や検索に際して暗号化された通信を提供します。
ldsrv_num_retries	TINYINT	失敗またはフェイルオーバー (指定されている場合) を返す前に LDAP サーバで認証を試みる回数。有効な範囲: 1-60
ldsrv_timeout	UNSIGNED INT	接続または検索のタイムアウト値 (単位はミリ秒)。有効な範囲: 1 ～ 3600000 (1 時間)。
ldsrv_last_state_change	TIMESTAMP	最後のステータスの変更が発生した時刻。サーバのローカルタイムゾーンに関係なく、この値は協定世界時 (UTC: Coordinated Universal Time) で格納されます。
ldsrv_search_url	CHAR(1024)	ユーザ ID に基づいてユーザの識別名 (DN) を探すための検索を定義する LDAP URL。
ldsrv_auth_url	CHAR(1024)	ユーザ ID が与えられたユーザの DN を探すための LDAP 検索文字列。
ldsrv_access_dn	CHAR(1024)	他のユーザ ID の DN を検索して取得するために LDAP サーバにアクセスする際に使用される DN。
ldsrv_access_dn_pwd	CHAR(3)	パスワードが格納されるかどうか。3 つのアスタリスク (***) は、パスワードが格納されることを示します。NULL は、パスワードが格納されないことを示します。パスワードのハッシュ値を取得するには、SYSUSERPASSWORD システムビューのクエリを実行します。

関連情報

SYSLDAPSERVERPASSWORD システムビュー [1816 ページ]

1.7.1.32 SYSLDAPSERVERPASSWORD システムビュー

SYSLDAPSERVERPASSWORD システムビューの各ローは、LDAP アクセスアカウントのオブジェクト ID とパスワードハッシュ値を示します。このビューの基本となるシステムテーブルは ISYSLDAPSERVER です。

このビューにアクセスするには、SELECT ANY TABLE システム権限と ACCESS USER PASSWORD システム権限が必要です。

カラム	データ型	説明
ldsrv_id	UNSIGNED BIGINT	LDAP サーバのユニークな識別子。この ID は、ログインポリシーによってこの LDAP サーバを参照するために使用されます。
ldsrv_access_dn_pwd	VARBINARY(1024)	LDAP アクセスアカウントのパスワードハッシュ値。

関連情報

SYSLDAPSERVER システムビュー [1814 ページ]

1.7.1.33 SYSLOGINMAP システムビュー

SYSLOGINMAP システムビューには、統合化ログインまたは Kerberos ログインを使用してデータベースに接続できる各ユーザに 1 つのローが含まれます。そのため、このビューへのアクセスは制限されています。このビューの基本となるシステムテーブルは ISYSLOGINMAP です。

カラム名	データ型	説明
login_mode	TINYINT	ログインの種類: 統合化ログインの場合は 1、Kerberos ログインの場合は 2。
login_id	VARCHAR(1024)	database_uid にマッピングする統合化ログインのユーザプロファイル名または Kerberos 原則。
object_id	UNSIGNED BIGINT	ユニークな識別子。ユーザ ID とデータベースのユーザ ID 間のマッピングごとに 1 つ。
database_uid	UNSIGNED INT	ログイン ID をマッピングするデータベースユーザ ID。

1.7.1.34 SYSLOGINPOLICY システムビュー

このビューの基本となるシステムテーブルは ISYSLOGINPOLICY です。

カラム名	データ型	説明
login_policy_id	UNSIGNED BIGINT	ログインポリシーのユニークな識別子。
login_policy_name	CHAR(128)	ログインポリシーの名前。

関連情報

SYSLOGINPOLICYOPTION システムビュー [1817 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.1.35 SYSLOGINPOLICYOPTION システムビュー

このビューの基本となるシステムテーブルは ISYSLOGINPOLICYOPTION です。

カラム名	データ型	説明
login_policy_id	UNSIGNED BIGINT	ログインポリシーのユニークな識別子。
login_option_name	CHAR(128)	ログインポリシーの名前。
login_option_value	LONG VARCHAR	ログインポリシーの作成時の値。

関連情報

SYSLOGINPOLICY システムビュー [1817 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.1.36 SYSMIRROROPTION システムビュー

このビューの基本となるシステムテーブルは ISYSMIRROROPTION です。

カラム名	データ型	説明
option_name	CHAR(128)	オプションの名前。

カラム名	データ型	説明
option_value	LONG VARCHAR	ミラーが作成された時点のオプション値。 SELECT ANY TABLE システム権限のないユーザの場合、このカラムの値は非表示になります。

関連情報

SYSMIRRORSERVER システムビュー [1818 ページ]
SYSMIRRORSERVEROPTION システムビュー [1819 ページ]
SET MIRROR OPTION 文 [1305 ページ]

1.7.1.37 SYSMIRRORSERVER システムビュー

このビューの基本となるシステムテーブルは ISYSMIRRORSERVER です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	ミラーサーバのユニークな識別子。
server_name	CHAR(128)	サーバの名前。
server_type	CHAR(20)	サーバのタイプ。値は、PRIMARY、MIRROR、ARBITER、PARTNER、または COPY のいずれかです。
parent	UNSIGNED BIGINT	親サーバ。値が NULL の場合、サーバはデータベースミラーリングシステムのプライマリサーバまたはミラーサーバです。 このカラムに値がある場合は、現在のサーバの親サーバの ID です。
alternate_parent	UNSIGNED BIGINT	現在の親が使用できなくなった場合に、代替の親として使用されるサーバの ID です。

関連情報

SYSMIRROROPTION システムビュー [1817 ページ]
SYSMIRRORSERVEROPTION システムビュー [1819 ページ]
CREATE MIRROR SERVER 文 [858 ページ]

1.7.1.38 SYSMIRRORSERVEROPTION システムビュー

このビューの基本となるシステムテーブルは ISYSMIRRORSERVEROPTION です。

カラム名	データ型	説明
server_id	UNSIGNED BIGINT	ミラーサーバのユニークな識別子。
option_name	CHAR(128)	オプションの名前。
option_value	LONG VARCHAR	ミラーが作成された時点のオプション値。

関連情報

SYSMIRROROPTION システムビュー [1817 ページ]
SYSMIRRORSERVER システムビュー [1818 ページ]

1.7.1.39 SYSMUTEXSEMAPHORE システムビュー

SYSMUTEXSEMAPHORE システムビューの各ローは、データベースのユーザ定義ミューテックスおよびセマフォに関する情報を示します。このビューの基本となるシステムテーブルは ISYSMUTEXSEMAPHORE です。

このビューにアクセスするには、SELECT ANY TABLE 権限が必要です。

カラム	データ型	説明
mutex_semaphore_id	UNSIGNED INT	ミューテックスまたはセマフォのユニークな ID。
object_id	UNSIGNED INT	ISYSOBJECT システムテーブルのミューテックスまたはセマフォのオブジェクト ID。
owner	UNSIGNED INT	ミューテックスまたはセマフォの所有者。
name	CHAR(128)	ミューテックスまたはセマフォの名前。
obj_type	CHAR(9)	オブジェクト (ミューテックスまたはセマフォ) のタイプ。
scope	CHAR(11)	ミューテックスまたはセマフォの範囲。ミューテックスの場合、CONNECTION は接続レベルの範囲を示し、TRANSACTION はトランザクションレベルの範囲を示します。セマフォの場合、この値は常に CONNECTION です。

カラム	データ型	説明
start_with	UNSIGNED INT	セマフォの初期カウンタ値。ミューテックスの場合、この値は NULL です。

1.7.1.40 SYSMVOPTION システムビュー

SYSMVOPTION システムビューの各ローは、マテリアライズドビューまたはテキストインデックスを作成した時点のオプション値に関する記述です。オプションの名前は、SYSMVOPTIONNAME システムビューにあります。このビューの基本となるシステムテーブルは ISYSMVOPTION です。

カラム名	データ型	説明
view_object_id	UNSIGNED BIGINT	マテリアライズドビューのオブジェクト ID。
option_id	UNSIGNED INT	データベースのオプションを識別するユニークな番号。オプション名を参照するには、SYSMVOPTIONNAME システムビューを参照してください。
option_value	LONG VARCHAR	マテリアライズドビューが作成された時点のオプション値。

関連情報

SYSMVOPTIONNAME システムビュー [1820 ページ]

1.7.1.41 SYSMVOPTIONNAME システムビュー

SYSMVOPTION システムビューの各ローには、マテリアライズドビューまたはテキストインデックスを作成した時点の名前オプション値があります。オプションの値は、SYSMVOPTION システムビューにあります。このビューの基本となるシステムテーブルは ISYSMVOPTIONNAME です。

カラム名	データ型	説明
option_id	UNSIGNED INT	データベースのオプションをユニークに識別する番号。
option_name	CHAR(128)	オプションの名前。

関連情報

SYSMVOPTION システムビュー [1820 ページ]

1.7.1.42 SYSOBJECT システムビュー

SYSOBJECT システムビューの各ローは個々のデータベースオブジェクトを記述します。このビューの基本となるシステムテーブルは ISYSOBJECT です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	データベースのオブジェクトをユニークに識別するインデックスの内部 ID。
status	TINYINT	オブジェクトのステータス。次のような値があります。 **1 (有効)** オブジェクトは、データベースサーバから使用できます。このステータスは、ENABLED と同等です。つまり、オブジェクトを ENABLE にすると、ステータスは VALID に変更されます。 **2 (無効)** 内部操作後に、オブジェクトを再コンパイルする試みが失敗しました。たとえば、依存するオブジェクトを変更するスキーマの変更後などです。データベースサーバは、文で参照されるオブジェクトを再コンパイルする試みを継続します。 **4 (無効化)** ユーザがオブジェクトを、明示的に無効化しました。たとえば、ALTER TABLE...DISABLE VIEW DEPENDENCIES 文を使用する場合などです。
object_type	TINYINT	オブジェクトの種類。
creation_time	TIMESTAMP	オブジェクトを作成したローカル日時。
object_type_str	CHAR(128)	オブジェクトの種類。
creation_time_utc	TIMESTAMP WITH TIME ZONE	オブジェクトを作成した UTC 日時。

1.7.1.43 SYSODATAPRODUCER システムビュー

SYSODATAPRODUCER システムビューの各ローは、OData プロデューサを示します。このビューの基本となるシステムテーブルは ISYSODATAPRODUCER です。

カラム名	データ型	説明
producer_id	UNSIGNED BIGINT	OData プロデューサの ID。
producer_name	VARCHAR(128)	OData プロデューサの名前。
admin_user	VARCHAR(128)	OData 管理者として機能するデータベースユーザ。
auth_user	VARCHAR(128)	データベースサーバへの接続時にすべてのユーザが使用する認証クレデンシャルを持つデータベースユーザ。
enabled	CHAR(1)	OData プロデューサが有効か無効かを指定します。
model	LONG BINARY	OData プロデューサのサービスモデルを (OSDL に) 指定します。
service_root	LONG VARCHAR	OData サーバで OData サービスのルートを指定します。
using_string	LONG VARCHAR	CREATE ODATA PRODUCER 文の USING 句によって取得される文字列を含みます。

1.7.1.44 SYSOPTION システムビュー

SYSOPTION システムビューには、データベースに格納されている各オプション設定のローのオプションが含まれます。

各ユーザはオプションごとに自分の設定を保存できます。また、PUBLIC ロールに対する設定は、自分の設定を持たないユーザが使うデフォルトの設定になります。このビューの基本となるシステムテーブルは ISYSOPTION です。

カラム名	データ型	説明
user_id	UNSIGNED INT	このオプション設定が適用されるユーザ番号。
option	CHAR(128)	オプションの名前。
"setting"	LONG VARCHAR	現在のオプションの設定。

1.7.1.45 SYSOPTSTAT システムビュー

SYSOPTSTAT システムビューには、ALTER DATABASE CALIBRATE 文によって計算されたコストモデル調整情報が入ります。このビューの内容は内部使用専用であり、sa_get_dtt システムプロシージャによってアクセスするのが最も良い方法です。このビューの基本となるシステムテーブルは ISYSOPTSTAT です。

カラム名	データ型	説明
stat_id	UNSIGNED INT	システムでのみ使用。
group_id	UNSIGNED INT	システムでのみ使用。
format_id	SMALLINT	システムでのみ使用。
data	LONG BINARY	システムでのみ使用。

1.7.1.46 SYSPHYSIDX システムビュー

SYSPHYSIDX システムビューの各ローはデータベースの個々の物理インデックスを定義します。このビューの基本となるシステムテーブルは ISYSPHYSIDX です。

カラム名	データ型	説明
table_id	UNSIGNED INT	インデックスが対応するテーブルのオブジェクト ID。
phys_index_id	UNSIGNED INT	テーブル内の物理インデックスのユニークな番号。
root	INTEGER	データベースファイルにおける物理インデックスのルートページの位置を識別します。
key_value_count	UNSIGNED INT	インデックス内の個別のキー値の数。
leaf_page_count	UNSIGNED INT	リーフインデックスページの数。
depth	UNSIGNED SMALLINT	物理インデックスの深さ (レベル数)。
max_key_distance	UNSIGNED INT	システムでのみ使用。
seq_transitions	UNSIGNED INT	システムでのみ使用。
rand_transitions	UNSIGNED INT	システムでのみ使用。
rand_distance	UNSIGNED INT	システムでのみ使用。
allocation_bitmap	LONG VARBIT	システムでのみ使用。
long_value_bitmap	LONG VARBIT	システムでのみ使用。

関連情報

SYSIDX システムビュー [1810 ページ]
SYSIDXCOL システムビュー [1811 ページ]

1.7.1.47 SYSPROCEDURE システムビュー

SYSPROCEDURE システムビューの各ローは、そのデータベースに保存されている個々のプロシージャを記述します。このビューの基本となるシステムテーブルは ISYSPROCEDURE です。

カラム名	データ型	説明
proc_id	UNSIGNED INT	データベースのプロシージャを一意に識別するプロシージャの内部プロシージャ番号。
creator	UNSIGNED INT	プロシージャの所有者。
object_id	UNSIGNED BIGINT	データベースのプロシージャをユニークに識別するプロシージャの内部 ID。
proc_name	CHAR(128)	プロシージャ名。1 人の作成者が、同じ名前のプロシージャを 2 つ持つことはできません。
proc_defn	LONG VARCHAR	データベースによって解析および構築された後のプロシージャの定義。
remarks	LONG VARCHAR	プロシージャに関する注記。ISYSREMARK システムテーブル内に格納されている値。
replicate	CHAR(1)	内部使用のみ。
srvid	UNSIGNED INT	プロシージャがリモートデータベースサーバ上のプロシージャのプロキシである場合、この値はそのリモートサーバを表します。
source	LONG VARCHAR	保護されたプロシージャのソース。この値は、ISYSSOURCE システムテーブル内に格納されています。preserve_source_format オプションが On に設定されている場合、内容はこのカラムにのみ格納されます。
avg_num_rows	FLOAT	プロシージャが FROM 句に含まれているときに、クエリの最適化に使用するために収集される情報。
avg_cost	FLOAT	プロシージャが FROM 句に含まれているときに、クエリの最適化に使用するために収集される情報。
stats	LONG BINARY	プロシージャが FROM 句に含まれているときに、クエリの最適化に使用するために収集される情報。
dialect	CHAR(1)	Watcom SQL プロシージャと関数に対しては W を、Transact-SQL プロシージャと関数に対しては T を返します。
is_deterministic	CHAR(1)	プロシージャに対して NULL を返します。決定的関数としてマークされている場合は Y を、非決定的関数としてマークされている場合は N を、いずれにもマークされていない場合は U を返します。

カラム名	データ型	説明
is_external	CHAR(1)	関数が外部である場合は Y を、そうでない場合は N を返します。
external_language	VARCHAR(128)	プロシージャが外部ではなく 'native' で、元のインタフェースを使用している場合は NULL を返します。そうでない場合は、プロシージャの言語を含みます。
external_name	VARCHAR(32767)	プロシージャが外部ではない場合は NULL を返します。外部である場合は、プロシージャの外部名を含みます。
sql_security	CHAR(1)	プロシージャが SQL SECURITY INVOKER である場合は I を、プロシージャが SQL SECURITY DEFINER である場合は D を返します。

関連情報

ERROR_LINE 関数 [その他] [340 ページ]
ERROR_STACK_TRACE 関数 [その他] [347 ページ]
sa_error_stack_trace システムプロシージャ [1504 ページ]
sa_stack_trace システムプロシージャ [1633 ページ]
STACK_TRACE 関数 [その他] [539 ページ]

1.7.1.48 SYSPROCPARM システムビュー

SYSPROCPARM システムビューの各ローは、データベース内のプロシージャまたは関数の 1 つのパラメータ、結果セットカラムまたは戻り値を示します。このビューの基本となるシステムテーブルは ISYSPROCPARM です。

カラム名	データ型	説明
proc_id	UNSIGNED INT	このパラメータが属するプロシージャまたは関数をユニークに識別します。
parm_id	SMALLINT	各プロシージャは、パラメータに 1 から順に番号を付けます。パラメータ番号の順は、定義された順になっています。関数の場合、最初のパラメータの内容は関数名であり、その関数の戻り値を表します。

カラム名	データ型	説明
parm_type	SMALLINT	パラメータは、次に示すタイプのいずれかに該当します。 **0** 標準のパラメータ (変数) **1** 結果カラム — 結果セットを返すプロシージャで使用 **2** SQLSTATE エラー値 **3** SQLCODE エラー値 **4** 関数からの戻り値
parm_mode_in	CHAR(1)	このパラメータが、プロシージャまたは関数に値を提供するかどうかを示します (IN または INOUT パラメータ)。
parm_mode_out	CHAR(1)	このパラメータが、プロシージャまたは関数からの値 (IN または INOUT パラメータ) と RESULT 句のカラムのどちらを返すかを示します。
domain_id	SMALLINT	SYSDOMAIN システムビューにリストされたデータ型番号から、パラメータのデータ型を識別します。
width	BIGINT	文字列パラメータでは長さ、数値パラメータでは精度、その他のデータ型では記憶領域のサイズをバイトで示します。
scale	SMALLINT	数値データ型の場合、小数点以下の桁数です。その他のデータ型の場合、このカラムの値は 1 です。
user_type	SMALLINT	パラメータのユーザ型 (適用できる場合)。
parm_name	CHAR(128)	パラメータの名前。
default	LONG VARCHAR	パラメータのデフォルト値。参照情報としてのみ表示されます。
remarks	LONG VARCHAR	常に NULL を返します。ODBC ドライバの旧バージョンを新しいパーソナルデータベースサーバで使用できるようにするために用意されています。
base_type_str	VARCHAR(32767)	パラメータの物理的な型を表す注釈付きの型文字列。

備考

SYSPROCPARM システムビューは、ALTER PROCEDURE...RECOMPILE 文の使用を含む、プロシージャまたは関数の作成または変更時に更新されます。

また、SYSPROCPARM は、古いプロシージャまたは関数が次の条件を満たすときにチェックポイントが実行される場合は常に更新されます。

- プロシージャまたは関数が変更されて以降、参照されている。
- プロシージャが RESULT 句を持つ、またはプロシージャが、RESULT 句を持たない他のプロシージャに対する呼び出しが 10 回ネストされた再帰プロシージャではない。

1.7.1.49 SYSPROCPERM システムビュー

SYSPROCPERM システムビューの各ローは、プロシージャに対する EXECUTE 権限を付与されているユーザを表します。このビューの基本となるシステムテーブルは ISYSPROCPERM です。

カラム名	データ型	説明
proc_id	UNSIGNED INT	プロシージャ番号は、EXECUTE 権限が付与されたプロシージャをユニークに特定します。
grantee	UNSIGNED INT	権限の被付与者のユーザ番号。

1.7.1.50 SYSPROXYTAB システムビュー

SYSPROXYTAB システムビューの各ローは、プロキシテーブルのリモートパラメータを示します。このビューの基本となるシステムテーブルは ISYSPROXYTAB です。

カラム名	データ型	説明
table_object_id	UNSIGNED BIGINT	プロキシテーブルのオブジェクト ID。
existing_obj	CHAR(1)	プロキシテーブルがリモートサーバに以前に存在していたかどうかを示します。
srvid	UNSIGNED INT	プロキシテーブルと関連付けられたリモートサーバのユニークな ID。
remote_location	LONG VARCHAR	リモートサーバに関するプロキシテーブルの位置。
location_escape_char	CHAR(1)	ロケーション区切り文字をエスケープするために使用されるエスケープ文字。

1.7.1.51 SYSPUBLICATION システムビュー

syspublication システムテーブルの各ローは、パブリケーションを示します。このビューの基本となるシステムテーブルは ISYSPUBLICATION です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	パブリケーションをユニークに識別する番号。
object_id	UNSIGNED BIGINT	データベースのパブリケーションをユニークに識別するプロシージャの内部 ID。
creator	UNSIGNED INT	パブリケーションの所有者。
publication_name	CHAR(128)	パブリケーションの名前。
remarks	LONG VARCHAR	パブリケーションに関する注記。ISYSREMARK システムテーブル内に格納されている値。
type	CHAR(1)	このカラムは使用されません。
sync_type	UNSIGNED INT	パブリケーションの同期の種類。次のような値があります。 **0 (ログスキャン)** トランザクションログを使用して、最後のアップロード後に変更されたすべての関連データをアップロードする通常のパブリケーションです。 **1 (スクリプト化されたアップロード)** このパブリケーションの場合、トランザクションログは無視され、アップロードは格納済みプロシージャを使用してユーザが定義します。ストアドプロシージャに関する情報は、ISYSSYNCSCRIPT システムテーブルに格納されます。 **2 (ダウンロード専用)** これはダウンロードのみのパブリケーションです。データはアップロードされません。

関連情報

SYSSYNCSCRIPT システムビュー [1841 ページ]

1.7.1.52 SYSREMARK システムビュー

SYSREMARK システムビューの各ローは、オブジェクトの注釈 (コメント) を示します。このビューの基本となるシステムテーブルは ISYSREMARK です。

カラム	データ型	説明
object_id	UNSIGNED BIGINT	関連する注釈があるオブジェクトの内部 ID。
remarks	LONG VARCHAR	オブジェクトと関連付けられた注釈またはコメント。

1.7.1.53 SYSREMOTEOPTION システムビュー

SYSREMOTEOPTION システムビューの各ローは、メッセージリンクパラメータの値を示します。このビューの基本となるシステムテーブルは ISYSREMOTEOPTION です。

このビューの一部のカラムには、機密データが含まれている可能性があります。SYSREMOTEOPTION2 ビューを使用すると、機密データを含む可能性のあるカラムを除いて、このビューのデータにパブリックアクセスできます。

カラム	データ型	説明
option_id	UNSIGNED INT	メッセージリンクパラメータの ID 番号。
user_id	UNSIGNED INT	パラメータが設定されているユーザ ID。
"setting"	VARCHAR(255)	メッセージリンクパラメータの値。

1.7.1.54 SYSREMOTEOPTIONTYPE システムビュー

SYSREMOTEOPTIONTYPE システムビューの各ローは、個々のメッセージリンクパラメータを示します。このビューの基本となるシステムテーブルは ISYSREMOTEOPTIONTYPE です。

カラム	データ型	説明
option_id	UNSIGNED INT	メッセージリンクパラメータの ID 番号。
type_id	SMALLINT	このパラメータを使用するメッセージタイプの ID 番号。
option	VARCHAR(128)	メッセージリンクパラメータの名前。

基本となるシステムテーブルに関する制約

```
PRIMARY KEY (option_id)
```

```
FOREIGN KEY (type_id) REFERENCES SYS.ISYSREMOTETYPE (type_id)
```

1.7.1.55 SYSREMOTETYPE システムビュー

SYSREMOTETYPE システムビューには、リモートテーブルに関する情報があります。このビューの基本となるシステムテーブルは ISYSREMOTETYPE です。

カラム名	データ型	説明
type_id	SMALLINT	メッセージシステムのうち、このユーザにメッセージを送信するために使用されているものを識別します。
object_id	UNSIGNED BIGINT	データベースのリモートタイプをユニークに識別するインデックスの内部 ID。
type_name	CHAR(128)	メッセージシステムの名前。
publisher_address	LONG VARCHAR	リモートデータベースパブリッシャのアドレス。
remarks	LONG VARCHAR	リモートタイプに関する注釈。ISYSREMARK システムテーブル内に格納されている値。

1.7.1.56 SYSREMOTEUSER システムビュー

SYSREMOTEUSER システムビューの各ローは、REMOTE システム権限を持つユーザ ID (サブスクライバ) を示します。そのユーザに送信されたメッセージと、そのユーザから送信されたメッセージのステータスも合わせて示します。このビューの基本となるシステムテーブルは ISYSREMOTEUSER です。

カラム名	データ型	説明
user_id	UNSIGNED INT	REMOTE 権限を持つユーザのユーザ番号。
consolidate	CHAR(1)	ユーザに CONSOLIDATE 権限 (Y) か REMOTE 権限 (N) が付与されていることを表します。
type_id	SMALLINT	メッセージシステムのうち、このユーザにメッセージを送信するために使用されているものを識別します。
address	LONG VARCHAR	メッセージが送信されるアドレス。アドレスは address_type に対して適切でなければいけません。
frequency	CHAR(1)	メッセージが送信される頻度。
send_time	TIME	次にメッセージがこのユーザへ送信される時刻。
log_send	UNSIGNED BIGINT	メッセージは、log_send が log_sent よりも大きいサブスクライバだけに送信されます。

カラム名	データ型	説明
time_sent	TIMESTAMP	このサブスクライバに、最後にメッセージが送信されたローカル時刻。
log_sent	UNSIGNED BIGINT	最後に送信されたオペレーションのログオフセット。
confirm_sent	UNSIGNED BIGINT	このサブスクライバから最後に確認されたオペレーションに対する、ログオフセット。
send_count	INTEGER	メッセージが送信された回数。
resend_count	INTEGER	メッセージがサブスクライバデータベースに、一度だけ適用されたことを確認するためのカウンタ。
time_received	TIMESTAMP	このサブスクライバから、最後にメッセージが受信されたローカル時刻。
log_received	UNSIGNED BIGINT	現在のデータベースで最後に受信されたオペレーションに対する、サブスクライバのデータベース内のログオフセット。
confirm_received	UNSIGNED BIGINT	最後に確認メッセージが送信されたオペレーションに対する、サブスクライバのデータベース内のログオフセット。
receive_count	INTEGER	メッセージが受信された回数。
rereceive_count	INTEGER	メッセージが現在のデータベースに、一度だけ適用されたことを確認するためのカウンタ。
time_sent_utc	TIMESTAMP WITH TIME ZONE	このサブスクライバに、最後にメッセージが送信された UTC 時刻。
time_received_utc	TIMESTAMP WITH TIME ZONE	このサブスクライバから、最後にメッセージが受信された UTC 時刻。

1.7.1.57 SYSROLEGRANT システムビュー

SYSROLEGRANT システムビューは、ロールのメンバーシップとメンバーシップのタイプに関する情報を格納します。このビューの基礎となるシステムテーブルは、ISYSROLEGRANT です。

カラム名	データ型	説明
grant_id	UNSIGNED INT	各 GRANT 文の識別に使用する ID。
role_id	UNSIGNED INT	ISYSUSER に基づいて、付与されるロールの ID。
grantee	UNISIGNED INT	ISYSUSER に基づいて、ロールを付与されるユーザの ID。

カラム名	データ型	説明
grant_type	TINYINT	3 桁を使用して付与のタイプを表します。右端のビットは、権限が付与されているかどうかを示します。2 桁目は、管理権限が付与されているかどうかを示します。3 桁目は、システム権限が継承可能かどうかを示します。 **001** 権限が付与されていますが、継承可能ではなく、管理権限もありません。DBA と REMOTE DBA を除く、継承不可のレガシー権限にのみ適用。 **101** 権限が付与されており、継承可能ですが、管理権限はありません。 **110** 管理権限のみが付与されています。 **111** 権限が付与されており、継承可能で、管理権限があります。
grant_scope	TINYINT	SET USER および CHANGE PASSWORD によって使用され、付与のスコープを設定します。値は、次のうちの 1 つまたは複数です。 **1** ANY **2** ユーザリスト **4** ロールリスト
grantor	CHAR(128)	付与者の名前。

1.7.1.58 SYSROLEGRANTEXT システムビュー

SET USER および CHANGE PASSWORD システム権限を付与するときは、被付与者がそれらのシステム権限を付与できるユーザまたはロールのリストを指定できます。

SYSROLEGRANTEXT システムビューには、被付与者が SET USER および CHANGE PASSWORD システム権限を付与できるユーザやロールに関する情報が格納されます。

このビューの基礎となるシステムテーブルは ISYSROLEGRANTEXT です。

カラム名	データ型	説明
grant_id	UNSIGNED INT	各 GRANT 文の識別に使用する ID。
user_id	UNSIGNED INT	`user-list` または `role-list` で特定の拡張付与に基づいて指定された user_ids。

備考

`user-list` オプションまたは ANY WITH ROLES `role-list` オプションを使用して、SET USER 権限や CHANGE PASSWORD 権限を付与したり取り消すときは、このビューが拡張構文の値によって更新されます。

1.7.1.59 SYSSCHEDULE システムビュー

SYSSCHEDULE システムビューの各ローには、CREATE EVENT の SCHEDULE 句で指定された、イベントが発生する時刻が記述されています。このビューの基本となるシステムテーブルは ISYSSCHEDULE です。

カラム名	データ型	説明
event_id	UNSIGNED INT	各イベントに割り当てられたユニークな番号。
sched_name	VARCHAR(128)	イベントのスケジュールと関連付けられた名前。
recurring	TINYINT	スケジュールを反復するかどうかを示します。
start_time	TIME	スケジュールの起動時刻。
stop_time	TIME	BETWEEN を指定した場合、スケジュールの停止時刻を示します。
start_date	DATE	イベントの実行がスケジュールされている最初の日付。
days_of_week	TINYINT	イベントがスケジュールされている曜日を示すビットマスク。 • x01 = 日曜日 • x02 = 月曜日 • x04 = 火曜日 • x08 = 水曜日 • x10 = 木曜日 • x20 = 金曜日 • x40 = 土曜日

カラム名	データ型	説明
days_of_month	UNSIGNED INT	イベントがスケジュールされている日付を示すビットマスク。いくつかの例を示します。 • x01 = 1 日 • x02 = 2 日 • x40000000 = 31 日 • x80000000 = その月の最終日
interval_units	CHAR(10)	EVERY で指定される間隔単位。 • HH = 時間 • NN = 分 • SS = 秒
interval_amt	INTEGER	EVERY で指定される期間。

1.7.1.60 SYSSEQUENCE システムビュー

SYSSEQUENCE システムビューには、各ユーザ定義シーケンスごとに 1 つのローがあります。このビューの基礎となるシステムテーブルは ISYSSEQUENCE です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	各シーケンスに割り当てられたユニークな番号。
owner	UNSIGNED INT	シーケンスの所有者。
min_value	BIGINT	シーケンスに許可されている最小値です。
max_value	BIGINT	シーケンスに許可されている最大値です。
increment_by	BIGINT	シーケンスの増分値。
start_with	BIGINT	シーケンスの開始値。
cache	UNSIGNED INT	より速いアクセスのためにメモリに事前割り付けされるシーケンス値の数です。値が 0 の場合は、値が事前割り付けされないことを示します。
cycle	TINYINT	最大値または最小値に達した後に、値の生成を継続するかどうか。
resume_at	BIGINT	ALTER SEQUENCE 文によって指定された RESTART WITH 値です。ALTER RESTART WITH 文が実行されていない場合、値は NULL です。
sequence_name	CHAR(128)	シーケンスの名前。

1.7.1.61 SYSSEQUENCEPERM システムビュー

SYSSEQUENCEPERM システムビューには、ユーザまたはグループがシーケンスに対して保持する権限が記録されます。このビューの基礎となるシステムテーブルは ISYSSEQUENCEPERM です。

カラム名	データ型	説明
sequence_id	UNSIGNED BIGINT	各シーケンスに割り当てられたユニークな番号。
grantee	UNSIGNED INT	シーケンスの変更または削除の権限を持つユーザまたはグループの ID です。
grantor	UNSIGNED INT	シーケンスに対する権限を付与されたユーザの ID です。
privilege_type	SMALLINT	ユーザまたはグループに付与されたシーケンスの権限のタイプです。

1.7.1.62 SYSSERVER システムビュー

SYSSERVER システムビューの各ローはリモートサーバを記述します。このビューの基本となるシステムテーブルは ISYSSERVER です。

> **注記**
>
> SYSSERVERS システムテーブルに含まれる前のカタログバージョン。このテーブル名は ISYSSERVER ('S' なし) に変更され、このビューの基本となるテーブルになります。

カラム名	データ型	説明
srvid	UNSIGNED INT	リモートサーバの識別子。
srvname	VARCHAR(128)	リモートサーバの名前。
srvclass	LONG VARCHAR	CREATE SERVER 文で指定されたサーバクラス。
srvinfo	LONG VARCHAR	サーバ情報。
srvreadonly	CHAR(1)	サーバが読み込み専用かどうか。

1.7.1.63 SYSSOURCE システムビュー

SYSSOURCE システムビューの各ローには、SYSOBJECT システムビューにリストされているオブジェクトのソースコード (適用できる場合) が含まれます。このビューの基本となるシステムテーブルは ISYSSOURCE です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	ソースコードが定義されているオブジェクトの内部 ID。
source	LONG VARCHAR	オブジェクトを作成したときに preserve_source_format データベースオプションが On の場合、このカラムにはオブジェクトの元のソースコードが含まれます。

1.7.1.64 SYSSPATIALREFERENCESYSTEM システムビュー

SYSSPATIALREFERENCESYSTEM システムビューの各ローは、データベースに定義されている SRS に関する記述です。このビューの基礎となるシステムテーブルは ISYSSPATIALREFERENCESYSTEM です。

このビューには、ST_SPATIAL_REFERENCE_SYSTEMS システムビューとは若干異なる量の情報があります。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	システムでのみ使用。
owner	UNSIGNED INT	SRS の所有者。
srs_name	CHAR(128)	SRS の名前。
srs_id	INTEGER	空間参照系の数値識別子 (SRID)。
round_earth	CHAR(1)	SRS タイプが ROUND EARTH (Y) または PLANAR (N) のいずれであるか。
axis_order	CHAR(12)	データベースサーバが緯度と経度に関連してポイントを解釈する方法を記述します (たとえば、ST_Lat メソッドと ST_Long メソッドを使用する場合)。非地理的空間参照系の場合、軸順序は x/y/z/m。地理的空間参照系の場合、デフォルトの軸順序は long/lat/z/m であり、lat/long/z/m もサポートされます。
snap_to_grid	DOUBLE	データベースサーバが計算を実行するときに使用するグリッドのサイズを定義します。
tolerance	DOUBLE	ポイントを比較するときに使用する精度を定義します。
semi_major_axis	DOUBLE	ROUND EARTH SRS の楕円の中心から赤道までの距離です。
semi_minor_axis	DOUBLE	ROUND EARTH SRS の楕円の中心から両極までの距離です。

カラム名	データ型	説明
inv_flattening	DOUBLE	ROUND EARTH SRS の楕円に使用される逆扁平率です。 逆扁平率 (f) は、回転楕円体の極から赤道への扁平度を定義する算術値です。値の範囲は、扁平なし (完全な円) から完全な扁平 (直線) までです。逆扁平率は、次に示すように、値 1/f です。`1/f = (semi_major_axis) / (semi_major_axis - semi_minor_axis)`
min_x	DOUBLE	座標において許される x の最小値です。
max_x	DOUBLE	座標において許される x の最大値です。
min_y	DOUBLE	座標において許される y の最小値です。
max_y	DOUBLE	座標において許される y の最大値です。
min_z	DOUBLE	座標において許される z の最小値です。
max_z	DOUBLE	座標において許される z の最大値です。
min_m	DOUBLE	座標において許される m の最小値です。
max_m	DOUBLE	座標において許される m の最大値です。
organization	LONG VARCHAR	空間参照系によって使用される座標系を作成した組織の名前。
organization_coordsys_id	INTEGER	座標系を作成した組織が座標系に対して指定した ID。

カラム名	データ型	説明
srs_type	CHAR(11)	SQL/MM 標準による定義に従った SRS のタイプ。次のいずれかの値を取ります。 **GEOGRAPHIC** 緯度と経度 (および標高) の軸を持つ、測地参照される座標系に基づく SRS 用です。これらの SRS は、PLANAR タイプまたは ROUND EARTH タイプです。 **PROJECTED** 緯度と経度の軸を持たず、測地参照される座標系に基づく SRS 用です。これらの SRS は PLANAR タイプです。 **ENGINEERING** 測地参照されない座標系に基づく SRS 用です。これらの SRS は PLANAR タイプです。 **GEOCENTRIC** サポートされていません。 **COMPOUND** サポートされていません。 **VERTICAL** サポートされていません。 srs_type が空の場合、タイプは指定されません。
linear_unit_of_measure	UNSIGNED BIGINT	空間参照系によって使用される線形測定単位です。
angular_unit_of_measure	UNSIGNED BIGINT	空間参照系によって使用される角度測定単位です。
count_in_use	UNSIGNED BIGINT	内部でのみ使用。
polygon_format	LONG VARCHAR	多角形のリングの方向。CounterClockwise、ClockWise、または EvenOdd のいずれかです。
storage_format	LONG VARCHAR	データが正規化フォーマット (Internal)、非正規化フォーマット (Original)、または両方 (Mixed) のいずれで格納されるか。
definition	LONG VARCHAR	OGC 標準によって定義されているフォーマットでの、空間参照系の WKT 定義です。
transform_definition	LONG VARCHAR	この SRS から別の SRS にデータを変換するときに使用される変換定義設定です。

1.7.1.65 SYSSQLSERVERTYPE システムビュー

SYSSQLSERVERTYPE システムビューには、Adaptive Server Enterprise との互換性に関する情報が含まれます。このビューの基本となるシステムテーブルは ISYSSQLSERVERTYPE です。

カラム名	データ型	説明
ss_user_type	SMALLINT	Adaptive Server Enterprise ユーザ型。
ss_domain_id	SMALLINT	Adaptive Server Enterprise のドメイン ID。
ss_type_name	VARCHAR(30)	Adaptive Server Enterprise 型名。
primary_sa_domain_id	SMALLINT	対応する SQL Anywhere のプライマリドメイン ID。
primary_sa_user_type	SMALLINT	対応する SQL Anywhere のプライマリユーザタイプ。

1.7.1.66 SYSSUBSCRIPTION システムビュー

SYSSUBSCRIPTION システムビューの各ローは、REMOTE システム権限を持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションに関する記述です。このビューの基本となるシステムテーブルは ISYSSUBSCRIPTION です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	ユーザ ID のサブスクリプションが作成されているパブリケーションの識別子。
user_id	UNSIGNED INT	パブリケーションに対して送信されたユーザの ID。
subscribe_by	CHAR(128)	サブスクリプション用の SUBSCRIBE BY 式がある場合、その値。
created	UNSIGNED BIGINT	トランザクションログ内の、サブスクリプションが作成されたオフセット。
started	UNSIGNED BIGINT	トランザクションログ内の、サブスクリプションが開始されたオフセット。

1.7.1.67 SYSSYNC システムビュー

SYSSYNC システムビューには、同期に関する情報が含まれます。

基本となるテーブル ISYSSYNC の "option" および server_connect カラムには、パスワードなどの機密情報が含まれます。このビューから SELECT を実行するには、SELECT ANY TABLE システム権限と ACCESS USER PASSWORD システム権限が必要です。SYSSYNC2 統合ビューから、機密データのない同じデータにパブリックアクセスできます。

このビューの基本となるシステムテーブルは ISYSSYNC です。

カラム名	データ型	説明
sync_id	UNSIGNED INT	ユニークにローを識別する番号。
type	CHAR(1)	値は常に D です。
publication_id	UNSIGNED INT	SYSPUBLICATION システムビューに入っている publication_id。
progress	UNSIGNED BIGINT	最後に成功したアップロードのログオフセット。
site_name	CHAR(128)	ユーザ名。
option	LONG VARCHAR	同期オプション。
server_connect	LONG VARCHAR	サーバのアドレスまたは URL。
server_conn_type	LONG VARCHAR	同期時に使用する、TCP/IP などの通信プロトコル。
last_download_time	TIMESTAMP	最後にサーバからダウンロードストリームを受信した時刻。
last_upload_time	TIMESTAMP	最後に情報のアップロードが成功した時刻 (サーバで測定)。デフォルトは jan-1-1900 です。
created	UNSIGNED BIGINT	サブスクリプションが作成されたログオフセット。
log_sent	UNSIGNED BIGINT	情報をどこまでアップロードしたかを示すログの進行状況。更新されるこのカラムのエントリに対するアップロード確認の受信は必要ありません。
generation_number	INTEGER	ファイルベースのダウンロードで、このサブスクリプションに対して最後に受信した世代番号。デフォルトは 0 です。
extended_state	VARCHAR(1024)	内部でのみ使用。
script_version	CHAR(128)	CREATE SYNCHRONIZATION SUBSCRIPTION 文、ALTER SYNCHRONIZATION SUBSCRIPTION 文、および START SYNCHRONIZATION SCHEMA CHANGE 文によって使用されるスクリプトバージョンを示します。
subscription_name	CHAR(128)	サブスクリプションの名前。
server_protocol	UNSIGNED BIGINT	内部でのみ使用。同期サーバのバージョンを識別するために内部で使用される値が含まれます。

関連情報

SYSSYNC2 統合ビュー [1882 ページ]

1.7.1.68 SYSSYNCPROFILE システムビュー

SYSSYNCPROFILE システムビューには、Mobile Link 同期の同期プロファイルに関する情報が含まれます。

このビューにアクセスするには、SELECT ANY TABLE システム権限と ACCESS USER PASSWORD システム権限が必要です。

このビューの基本となるシステムテーブルは ISYSSYNCPROFILE です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	同期プロファイルのオブジェクト ID。
profile_name	CHAR(128)	同期プロファイルの名前。
profile_defn	LONG VARCHAR	同期プロファイルの定義。

1.7.1.69 SYSSYNCPROFILE2 システムビュー

SYSSYNCPROFILE2 システムビューには、Mobile Link 同期の同期プロファイルに関する情報が含まれます。

このビューの基本となるシステムテーブルは ISYSSYNCPROFILE です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	同期プロファイルのオブジェクト ID。
profile_name	CHAR(128)	同期プロファイルの名前。

1.7.1.70 SYSSYNCSCRIPT システムビュー

SYSSYNCSCRIPT システムビューの各ローは、スクリプト化されたアップロードに関するストアドプロシージャを識別します。このビューは SYSSYNCSCRIPTS ビューとほとんど同じですが、このビューでは値が生の形式で表示されます。

このビューの基本となるシステムテーブルは ISYSSYNCSCRIPT です。

カラム名	データ型	説明
pub_object_id	UNSIGNED BIGINT	スクリプトが所属するパブリケーションのオブジェクト ID。
table_object_id	UNSIGNED BIGINT	スクリプトの適用対象となるテーブルのオブジェクト ID。

カラム名	データ型	説明
type	UNSIGNED INT	アップロードプロシージャのタイプ。
proc_object_id	UNSIGNED BIGINT	パブリケーションに使用するストアドプロシージャのオブジェクト ID。

関連情報

SYSSYNCSCRIPTS 統合ビュー [1884 ページ]
SYSPROCEDURE システムビュー [1824 ページ]
SYSPUBLICATION システムビュー [1828 ページ]

1.7.1.71 SYSTAB システムビュー

SYSTAB システムビューの各ローは、データベース内の 1 つのテーブルまたはビューを示します。ビューの追加情報が SYSVIEW システムビューにあります。このビューの基本となるシステムテーブルは ISYSTAB です。

カラム名	データ型	説明
table_id	UNSIGNED INT	各テーブルにはユニークな番号 (テーブル番号) が割り当てられます。
dbspace_id	SMALLINT	テーブルを含む DB 領域を示す値。
count	UNSIGNED BIGINT	テーブルまたはマテリアライズドビュー内のロー数。この値は、各チェックポイントが正常処理されているときに更新されます。この数を使用してデータベースアクセスが最適化されます。非マテリアライズドビューの場合、またはリモートテーブルの場合、カウントは常に 0 です。
creator	UNSIGNED INT	このユーザ番号はテーブルまたはビューの所有者を示します。
table_page_count	INTEGER	基本となるテーブルで使用されるメインページの総数。
ext_page_count	INTEGER	このテーブルで使用される拡張ページの総数。

カラム名	データ型	説明
commit_action	INTEGER	グローバルテンポラリテーブルの場合、0 は、テーブルの作成時に ON COMMIT PRESERVE ROWS 句が指定されたことを示します。1 は、テーブルの作成時に ON COMMIT DELETE ROWS 句が指定されたことを示します (テンポラリテーブルのデフォルト動作)。3 は、テーブルの作成時に NOT TRANSACTIONAL 句が指定されたことを示します。非テンポラリテーブルの場合、commit_action は常に 0 です。
share_type	INTEGER	グローバルテンポラリテーブルの場合、4 は、テーブルの作成時に SHARE BY ALL 句が指定されたことを示します。5 は、テーブルの作成時に SHARE BY ALL 句が指定されなかったことを示します。非テンポラリテーブルの場合、share_type は常に 5 です。これは、非テンポラリテーブルの作成時に SHARE BY ALL 句は指定できないためです。
object_id	UNSIGNED BIGINT	テーブルのオブジェクト ID。
last_modified_at	TIMESTAMP	テーブルのデータが最後に修正されたローカル日時。このカラムは、チェックポイント時にのみ更新されます。
table_name	CHAR(128)	テーブルまたはビューの名前。1 人の作成者が、同じ名前のテーブルまたはビューを 2 つ以上持つことはできません。
table_type	TINYINT	テーブルまたはビューのタイプ。次のような値があります。 **1** ベーステーブル **2** マテリアライズドビュー **3** グローバルテンポラリテーブル **4** ローカルテンポラリテーブル **5** テキストインデックスベーステーブル **6** テキストインデックスグローバルテンポラリテーブル **21** ビュー

カラム名	データ型	説明
replicate	CHAR(1)	この値は内部でのみ使用されます。
server_type	TINYINT	基本となるテーブルのデータの場所。次のような値があります。 **1** ローカルサーバ **3** リモートサーバ
tab_page_list	LONG VARBIT	内部でのみ使用。テーブルの情報を含むページセット (ビットマップで表示されます)。
ext_page_list	LONG VARBIT	内部でのみ使用。テーブルに関するローの拡張と大規模なオブジェクト (LOB) ページを含むページセット (ビットマップで表示されます)。
pct_free	UNSIGNED INT	指定されている場合、テーブルの PCT_FREE の指定、それ以外の場合は NULL。
clustered_index_id	UNSIGNED INT	テーブルのクラスタ化されたインデックスの ID。クラスタ化されたインデックスがない場合、このフィールドは NULL です。
encrypted	CHAR(1)	テーブルまたはマテリアライズドビューが暗号化されるかどうか。
last_modified_tsn	UNSIGNED BIGINT	テーブルを修正したトランザクションに割り当てられたシーケンス番号。このカラムは、チェックポイント時にのみ更新されます。
current_schema	UNSIGNED INT	テーブルの現在のスキーマバージョン。
file_id	SMALLINT	廃止予定。このカラムは SYSVIEW に存在しますが、基本となるシステムテーブル ISYSTAB には存在しません。このカラムの内容は dbspace_id と同じであり、互換性のために提供されています。今後は dbspace_id を使用してください。
table_type_str	CHAR(13)	table_type の読み取り可能な値。次のような値があります。 **BASE** ベーステーブル **MAT VIEW** マテリアライズドビュー **GBL TEMP** グローバルテンポラリテーブル **VIEW** ビュー

カラム名	データ型	説明
last_modified_at_utc	TIMESTAMP WITH TIME ZONE	テーブルのデータが最後に修正された UTC 日時。このカラムは、チェックポイント時にのみ更新されます。

関連情報

SYSVIEW システムビュー [1860 ページ]

1.7.1.72 SYSTABCOL システムビュー

SYSTABCOL システムビューには、データベース内の各テーブルとビューに対して 1 つのローがあります。このビューの基本となるシステムテーブルは ISYSTABCOL です。

カラム名	データ型	説明
table_id	UNSIGNED INT	カラムが属するテーブルまたはビューのオブジェクト ID。
column_id	UNSIGNED INT	カラムの ID。各テーブルについて、カラムの番号は 1 から開始されます。 column_id 値によって、SELECT * が使用された場合の結果セットのカラムの順序が決定されます。また、INSERT 文にカラム名のリストが指定されない場合も、この値によってカラム順が決定されます。
domain_id	SMALLINT	カラムのデータ型を、SYSDOMAIN システムビューにリストされているデータ型番号で示します。
nulls	CHAR(1)	NULL 値をカラムに許可するかどうかを指定します。
width	BIGINT	文字列カラムでは長さ、数値カラムでは精度、その他のデータ型では格納サイズをバイトで示します。
scale	SMALLINT	NUMERIC または DECIMAL データ型のカラムについて、小数点以下の桁数。文字列のカラムの場合、1 の値は文字長のセマンティックを指定します。0 は、バイト長のセマンティックを指定します。
object_id	UNSIGNED BIGINT	テーブルカラムのオブジェクト ID。

カラム名	データ型	説明
max_identity	BIGINT	AUTOINCREMENT、IDENTITY、または GLOBAL AUTOINCREMENT カラムの場合、カラムの最大値。
column_name	CHAR(128)	カラム名。
"default"	LONG VARCHAR	カラムのデフォルト値。この値を指定した場合、INSERT 文が値を指定しないときのみ使われます。
user_type	SMALLINT	ユーザ定義のデータ型を使用してカラムが定義される場合、データ型。
column_type	CHAR(1)	カラムのタイプ (C=計算済みカラム、R=他のカラム)。
compressed	TINYINT	このカラムを圧縮形式で格納するかどうか。
collect_stats	TINYINT	システムが自動的にカラムの統計情報を収集および更新するかどうか。
inline_max	SMALLINT	ローに格納する BLOB の最大バイト数。NULL 値は、デフォルトが適用されたこと、またはカラムは文字型またはバイナリ型ではないことを示します。NULL ではない inline_max 値は、CREATE TABLE 文または ALTER TABLE 文を使用してカラムに指定した INLINE 値に対応します。
inline_long	SMALLINT	BLOB サイズが inline_max 値を超えた場合、ローに格納されている BLOB のバイトを複製した数。NULL 値は、デフォルトが適用されたこと、またはカラムは文字型またはバイナリ型ではないことを示します。NULL ではない inline_long 値は、CREATE TABLE 文または ALTER TABLE 文を使用してカラムに指定した PREFIX 値に対応します。
lob_index	TINYINT	内部的なしきい値サイズ (約 8 データベースページ) を超過するカラムの BLOB 値に関するインデックスを構築するかどうか。NULL 値は、デフォルトを適用するか、カラムが BLOB タイプでないことを示します。1 の値は、インデックスを構築することを示します。0 の値は、インデックスを構築しないことを示します。NULL ではない lob_index 値は、CREATE TABLE 文または ALTER TABLE 文を使用してカラムに指定した INDEX または NO INDEX 値に対応します。
base_type_str	VARCHAR(32,767)	カラムの物理的な型を表す注釈付きの型文字列。
nonmaterialized_value	LONG BINARY	内部使用のみ。
start_schema	UNSIGNED INT	このカラムが存在するテーブルスキーマの最初のバージョン。

関連情報

CREATE TABLE 文 [952 ページ]

1.7.1.73 SYSTABLEPERM システムビュー

GRANT 文によってテーブルやビューに対して付与された権限は、SYSTABLEPERM システムビューに格納されます。このビューの各ローは 1 つのテーブル、権限を与えるユーザ ID (grantor)、そして権限を与えられるユーザ ID (grantee) に対応します。このビューの基本となるシステムテーブルは ISYSTABLEPERM です。

カラム名	データ型	説明
stable_id	UNSIGNED INT	権限が適用されるテーブルまたはビューのテーブル番号。
grantee	UNSIGNED INT	権限を受けるユーザ ID のユーザ番号。
grantor	UNSIGNED INT	権限を与えるユーザ ID のユーザ番号。
selectauth	CHAR(1)	SELECT 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
insertauth	CHAR(1)	INSERT 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
deleteauth	CHAR(1)	DELETE 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
updateauth	CHAR(1)	UPDATE 権限が、テーブル内のすべてのカラムに付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
updatecols	CHAR(1)	UPDATE 権限が、基本となるテーブル内の一部のカラムだけに付与されたかどうかを示します。updatecols が "Y" であれば、カラムに UPDATE 権限を与える 1 つまたは複数のローが、SYSCOLPERM システムビューにあります。
alterauth	CHAR(1)	ALTER 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。

カラム名	データ型	説明
referenceauth	CHAR(1)	REFERENCE 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
loadauth	CHAR(1)	LOAD 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
truncateauth	CHAR(1)	TRUNCATE 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
loadauth	CHAR(1)	LOAD 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
truncateauth	CHAR(1)	TRUNCATE 権限が付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。

備考

与えられる権限には型がいくつかあります。それぞれの権限は、次の 3 つのいずれかの値を持ちます。

N

No。grantee はこの権限を grantor から付与されていません。

Y

Yes。grantee はこの権限を grantor から付与されています。

G

grantee はこの権限を受けています。また、grantee は同じ権限を他のユーザに付与できます。

注記

grantee は同じテーブルに関する権限を、他の grantor から受けることがあります。その場合、この情報は、SYSTABLEPERM システムビューの異なるローで見つかります。

基本となるシステムテーブルに関する制約

```
PRIMARY KEY (stable_id, grantee, grantor)
```

```
FOREIGN KEY (stable_id) REFERENCES SYS.ISYSTAB (table_id)
```

```
FOREIGN KEY (grantor) REFERENCES SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)
```

関連情報

GRANT 文 [1131 ページ]

1.7.1.74 SYSTEXTCONFIG システムビュー

SYSTEXTCONFIG システムビューの各ローは、全文検索機能で使用するテキスト設定オブジェクト 1 つを示します。このビューの基本となるシステムテーブルは ISYSTEXTCONFIG です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	テキスト設定オブジェクトのオブジェクト ID。
creator	UNSIGNED INT	テキスト設定オブジェクトの作成者。
term_breaker	TINYINT	文字列を単語に分割するために使用されるアルゴリズム。値は GENERIC の場合は 0、NGRAM の場合は 1 です。GENERIC の場合、英数字以外の文字で区切られた 1 つまたは複数の英数字の文字列は、単語として扱われます。NGRAM は近似一致または単語の区切りにホワイトスペースを使用しないドキュメントに使用します。
stemmer	TINYINT	内部でのみ使用。
min_term_length	TINYINT	1 つの単語に許容される最小文字数。min_term_length より短い単語は無視されます。 MINIMUM TERM LENGTH 設定は、GENERIC 単語区切りの場合にのみ意味を持ちます。NGRAM テキストインデックスの場合、この設定は無視されます。

カラム名	データ型	説明
max_term_length	TINYINT	GENERIC テキストインデックスの場合は、1 つの単語に許容される最大長 (文字数)。max_term_length より長い単語は無視されます。 NGRAM テキストインデックスの場合は、単語が分解される N-gram の長さ。
collation	CHAR(128)	内部でのみ使用。
text_config_name	CHAR(128)	テキスト設定オブジェクトの名前。
prefilter	LONG VARCHAR	外部事前フィルタライブラリの関数名およびライブラリ名です。
postfilter	LONG VARCHAR	内部でのみ使用。
char_stoplist	LONG VARCHAR	CHAR カラムでの全文検索実行時に無視する単語。これらの単語は、テキストインデックスからも省略されます。このカラムは、テキスト設定オブジェクトが default_char から作成されるときに使用されます。
nchar_stoplist	LONG NVARCHAR	NCHAR カラムでの全文検索実行時に無視する単語。これらの単語は、テキストインデックスからも省略されます。このカラムは、テキスト設定オブジェクトが default_nchar から作成されるときに使用されます。
external_term_breaker	LONG VARCHAR	外部単語区切りライブラリの関数名およびライブラリ名です。

1.7.1.75 SYSTEXTIDX システムビュー

SYSTEXTIDX システムビューの各ローは、1 つのテキストインデックスを示します。このビューの基本となるシステムテーブルは ISYSTEXTIDX です。

カラム名	データ型	説明
index_id	UNSIGNED BIGINT	SYSIDX のテキストインデックスのオブジェクト ID。
sequence	UNSIGNED INT	内部でのみ使用。
status	UNSIGNED INT	内部でのみ使用。
text_config	UNSIGNED BIGINT	SYSTEXTCONFIG のテキスト設定オブジェクトのオブジェクト ID。
next_handle	UNSIGNED INT	内部でのみ使用。
last_handle	UNSIGNED INT	内部でのみ使用。

カラム名	データ型	説明
deleted_length	UNSIGNED BIGINT	テキストインデックスの削除されたインデックス付きの値の合計サイズ。
pending_length	UNSIGNED BIGINT	次回のリフレッシュ時にテキストインデックスに追加されるインデックス付きの値の合計サイズ。
refresh_type	TINYINT	再表示タイプ。下記のいずれかです。 **1** MANUAL **2** AUTO **3** IMMEDIATE
refresh_interval	UNSIGNED INT	AUTO REFRESH の間隔 (分単位)。
last_refresh	TIMESTAMP	最後のリフレッシュのローカル時刻。
last_refresh_utc	TIMESTAMP WITH TIME ZONE	最後のリフレッシュの UTC 時刻。

1.7.1.76 SYSTEXTIDXTAB システムビュー

SYSTEXTIDXTAB システムビューの各ローは、テキストインデックスの一部である生成されたテーブルを示します。このビューの基本となるシステムテーブルは ISYSTEXTIDXTAB です。

カラム名	データ型	説明
index_id	UNSIGNED BIGINT	内部でのみ使用。
sequence	UNSIGNED INT	内部でのみ使用。
table_type	UNSIGNED INT	内部でのみ使用。
table_id	UNSIGNED INT	内部でのみ使用。

1.7.1.77 SYSTIMEZONE システムビュー

SYSTIMEZONE システムビューの各ローは、1 つのタイムゾーンを示します。このビューの基本となるシステムテーブルは ISYSTIMEZONE です。

カラム名	データ型	説明
timezone_id	UNSIGNED BIGINT	SYSTIMEZONE のタイムゾーンのオブジェクト ID。
name	VARCHAR(128)	タイムゾーンの名前。

カラム名	データ型	説明
offset	INTEGER	タイムゾーンのオフセット。
dst_offset	INTEGER	夏時間のオフセット。
start_dst_month	INTEGER	夏時間が開始する月。
start_dst_day	VARCHAR(128)	夏時間が開始する日。
start_dst_time	INTEGER	夏時間が開始する時間。
end_dst_month	INTEGER	夏時間が終了する月。
end_dst_day	VARCHAR(128)	夏時間が終了する日。
end_dst_time	INTEGER	夏時間が終了する時間。
alias_id	UNSIGNED BIGINT	内部使用のみ。

備考

このビューの基本となるテーブル ISYSTIMEZONE は、新規データベースの場合、値が入力されていません。ISYSTIMEZONE に値を入力するには、次の文のいずれかを実行します。

CREATE TIME ZONE 文
ALTER TIME ZONE 文

関連情報

ALTER TIME ZONE 文 [726 ページ]
COMMENT 文 [769 ページ]
CREATE TIME ZONE 文 [979 ページ]
DROP TIME ZONE 文 [1078 ページ]

1.7.1.78 SYSTRIGGER システムビュー

SYSTRIGGER システムビューの各ローは、データベース内のトリガ 1 つを示します。このテーブルには、参照トリガアクションを持つ外部キー定義に、自動的に作成されるトリガも含まれます (たとえば、ON DELETE CASCADE)。このビューの基本となるシステムテーブルは ISYSTRIGGER です。

カラム名	データ型	説明
trigger_id	UNSIGNED INT	SYSTRIGGER ビュー内のトリガのユニークな番号です。
table_id	UNSIGNED INT	トリガが所属するテーブルのテーブル ID。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	データベース内のトリガのオブジェクト ID です。
event	CHAR(1)	トリガが起動されるきっかけとなる操作。 **A** INSERT、DELETE **B** INSERT、UPDATE **C** UPDATE COLUMNS **D** DELETE **E** DELETE、UPDATE **I** INSERT **M** INSERT、DELETE、UPDATE **U** UPDATE
trigger_time	CHAR(1)	イベントに関連するトリガが起動されるタイミング。 **A** AFTER (ローレベルのトリガ) **B** BEFORE (ローレベルのトリガ) **I** INSTEAD OF (ローレベルのトリガ) **K** INSTEAD OF (文レベルのトリガ) **R** RESOLVE **S** AFTER (文レベルのトリガ)

カラム名	データ型	説明
trigger_order	SMALLINT	同じタイプ (insert、update、または delete) の複数のトリガが、同じ時刻 (BEFORE または AFTER トリガのみが該当) に起動するように設定されている場合に、トリガが起動される順序。
foreign_table_id	UNSIGNED INT	参照トリガアクション (たとえば ON DELETE CASCADE) を持つ外部キー定義のあるテーブルの ID。foreign_table_id 値は、ISYSIDX.table_id の値を反映します。
foreign_key_id	UNSIGNED INT	foreign_table_id が参照するテーブルの外部キーの ID。foreign_key_id 値は、ISYSIDX.index_id の値を反映します。
referential_action	CHAR(1)	外部キーによって定義される動作。この 1 文字の値は、外部キーを作成したときに指定した動作に対応します。 **C** CASCADE **D** SET DEFAULT **N** SET NULL **R** RESTRICT
trigger_name	CHAR(128)	トリガ名。1 つのテーブルには、同じ名前のトリガは複数存在できません。
trigger_defn	LONG VARCHAR	トリガを作成するのに使ったコマンド。
remarks	LONG VARCHAR	トリガに関する注記。ISYSREMARK システムテーブル内に格納されている値。
source	LONG VARCHAR	トリガの SQL ソース。この値は、ISYSSOURCE システムテーブル内に格納されています。

1.7.1.79 SYSTYPEMAP システムビュー

SYSTYPEMAP システムビューには、SYSSQLSERVERTYPE システムビューのエントリの互換性マッピング値があります。このビューの基本となるシステムテーブルは ISYSTYPEMAP です。

カラム名	データ型	説明
ss_user_type	SMALLINT	Adaptive Server Enterprise ユーザ型を格納します。
sa_domain_id	SMALLINT	対応する SQL Anywhere domain_id を格納します。
sa_user_type	SMALLINT	対応する SQL Anywhere ユーザ型を格納します。
nullable	CHAR(1)	型が NULL 値を許容するかどうか。

1.7.1.80 SYSUNITOFMEASURE システムビュー

SYSUNITOFMEASURE システムビューの各ローは、データベースに定義されている測定単位に関する記述です。SYSUNITOFMEASURE システムビューの基本となるテーブルは、ISYSUNITOFMEASURE システムテーブルです。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	システムでのみ使用。
owner	UNSIGNED INT	測定単位の所有者。
unit_name	CHAR(128)	測定単位の名前。
unit_type	CHAR(7)	角度または線形。
conversion_factor	DOUBLE	測定単位の換算係数。

1.7.1.81 SYSUSER システムビュー

SYSUSER システムビューの各ローは、そのシステム内の個々のトリガを記述します。

スタンドアロンロールも同様にこのビューに格納されますが、これらのロールにとっては user_id、object_id、user_name、user_type カラムのみ意味があります。このビューの基本となるシステムテーブルは ISYSUSER です。

カラム名	データ型	説明
user_id	UNSIGNED INT	ログインポリシーが割り当てられたユーザのユニークな識別子。
object_id	UNSIGNED BIGINT	データベースのユーザのユニークな識別子。
user_name	CHAR(128)	ユーザのログイン名。

カラム名	データ型	説明
password	CHAR(3)	パスワードが格納されるかどうか。3 つのアスタリスク (***) は、パスワードが格納されることを示します。NULL は、パスワードが格納されないことを示します。実際のパスワードのハッシュ値を返すには、SYSUSERPASSWORD システムビューのクエリを実行します。
login_policy_id	UNSIGNED BIGINT	ログインポリシーのユニークな識別子。
expire_password_on_login	TINYINT	次回のログイン時にユーザのパスワードの有効期限が切れるかどうかを示す値。
password_creation_time	TIMESTAMP	ユーザのパスワードが作成されたローカル時刻。
failed_login_attempts	UNSIGNED INT	アカウントがロックされるまでにユーザがログインを失敗できる回数。
last_login_time	TIMESTAMP	ユーザが最後にログインした現地時刻。
user_type	TINYINT	ユーザが通常のユーザなのか、ロールなのか、ロールとして拡張されたユーザなのかを示す値。ユーザ、ロール、または拡張ロールが変更可能 (可変) なのか、または削除可能なのかを示します。考えられる値は、次のとおりです。 **1** 不変のシステムロール。 **5** 可変のシステムロール。 **9** 不変および削除可能なシステムロール。 **12** 可変および削除可能なユーザ。 **13** 可変および削除可能なロール。 **14** 可変および削除可能な、ロールとして拡張されたユーザ。
user_dn	CHAR(1024)	ドメイン内および複数のドメインでユニークな、ユーザの LDAP 識別名 (DN) の ID。DN は LDAP サーバでの認証に使用されます。

カラム名	データ型	説明
user_dn_cached_at	TIMESTAMP	user_dn カラムが最後にキャッシュされた時刻。この値は古い DN をパージするかどうか決めるために使用されます。データベースサーバのローカルのタイムゾーンに関係なく、この値は協定世界時 (UTC: Coordinated Universal Time) で格納されます。
password_creation_time_utc	TIMESTAMP WITH TIME ZONE	ユーザのパスワードが作成された UTC 時刻。
last_login_time_utc	TIMESTAMP WITH TIME ZONE	ユーザが最後にログインした UTC 時刻。
dual_password	CHAR(3)	ユーザの二重パスワードの 2 番目の部分を格納するかどうか。3 つのアスタリスク (***) は、2 番目の部分を格納することを示します。NULL は、2 番目の部分がないことを示します。実際のパスワードのハッシュ値を返すには、SYSUSERPASSWORD システムビューのクエリを実行します。
lock_time	TIMESTAMP	ログイン試行の失敗のためユーザがロックされたタイムスタンプ。

関連情報

SYSUSERPASSWORD システムビュー [1857 ページ]
sp_sys_priv_role_info システムプロシージャ [1734 ページ]
SYSLOGINPOLICY システムビュー [1817 ページ]
SYSLOGINPOLICYOPTION システムビュー [1817 ページ]

1.7.1.82 SYSUSERPASSWORD システムビュー

SYSUSERPASSWORD システムビューの各ローは、ログインアカウントのオブジェクト ID とパスワードハッシュ値を示します。このビューの基本となるシステムテーブルは ISYSUSER です。

このビューにアクセスするには、SELECT ANY TABLE システム権限と ACCESS USER PASSWORD システム権限が必要です。

カラム	データ型	説明
object_id	UNSIGNED BIGINT	ログインアカウントの内部 ID。
password	BINARY(128)	ログインアカウントのパスワードハッシュ値。

カラム	データ型	説明
dual_password	BINARY(128)	二重パスワードの部分のパスワードハッシュ値。

関連情報

SYSUSER システムビュー [1855 ページ]

1.7.1.83 SYSUSERMESSAGE システムビュー

SYSUSERMESSAGE システムビューの各ローは、エラー条件に対する個々のユーザ定義メッセージです。このビューの基本となるシステムテーブルは ISYSUSERMESSAGE です。

SYSUSERMESSAGES システムテーブルに含まれる前のカタログバージョン。このテーブル名は ISYSUSERMESSAGE ('S' なし) に変更され、このビューの基本となるテーブルになります。

カラム名	データ型	説明
error	INTEGER	エラー条件に対するユニークな識別番号。
uid	UNSIGNED INT	メッセージを定義するユーザ番号。
description	VARCHAR(255)	エラー条件に対応するメッセージ。
langid	SMALLINT	予約済み。

1.7.1.84 SYSUSERTYPE システムビュー

SYSUSERTYPE システムビューの各ローは、ユーザ定義のデータ型の記述です。このビューの基本となるシステムテーブルは ISYSUSERTYPE です。

カラム名	データ型	説明
type_id	SMALLINT	ユーザ定義データ型のユニークな識別番号。
creator	UNSIGNED INT	データ型の所有者のユーザ番号。
domain_id	SMALLINT	ユーザ定義データ型の元になるデータ型。SYSDOMAIN システムビューにリストされたデータ型番号で示します。
nulls	CHAR(1)	ユーザ定義データ型が NULL を許可するかどうか。可能な値は Y、N、または U です。値 U は、NULL 入力属性が指定されていないことを示します。

カラム名	データ型	説明
width	BIGINT	文字列カラムでは長さ、数値カラムでは精度、その他のデータ型では記憶領域のサイズをバイト数で示します。
scale	SMALLINT	数値データ型カラムでは小数点以下の桁数、その他のデータ型では 0 を示します。
type_name	CHAR(128)	データ型の名前。
"default"	LONG VARCHAR	データ型のデフォルト値。
"check"	LONG VARCHAR	データ型の CHECK 条件。
base_type_str	VARCHAR(32767)	ユーザタイプの物理的な型を表す注釈付きの型文字列。

1.7.1.85 SYSDATABASEVARIABLE システムビュー

SYSDATABASEVARIABLE システムビューの各ローは、データベース内のデータベーススコープ変数 1 つを示します。このビューの基本となるシステムテーブルは ISYSDATABASEVARIABLE です。

カラム名	データ型	説明
variable_id	UNSIGNED INT	データベース変数の ID。
object_id	UNSIGNED BIGINT	データベースのデータベーススコープ変数をでニークに識別するデータベーススコープ変数の内部 ID。
owner	UNSIGNED INT	データベース変数の所有者。
variable_name	CHAR(128)	データベース変数の名前。
domain_id	SMALLINT	SYSDOMAIN システムビューに表示されているデータ型の ID。
width	UNSIGNED INT	データベース変数が保持できる文字列の長さ、カラムの数値の精度、その他のデータ型で必要な格納サイズをバイトで示します。
scale	SMALLINT	NUMERIC または DECIMAL データ型の変数について、小数点以下の桁数。文字列を含むデータベース変数の場合、1 の値は文字長のセマンティックを指定します。0 は、バイト長のセマンティックを指定します。
user_type	SMALLINT	データベース変数のデータ型。型の値が存在しない場合は NULL。

カラム名	データ型	説明
initial_value	LONG BINARY	データベース変数の初期値。値を指定しない場合、NULL が使用されます。 式によって初期値が設定されると、その式は作成時に評価され、(式ではなく) 結果の定数がこのカラムに保存されます。
base_type_str	VARCHAR(32767)	データベース変数の物理的な型を表す注釈付きの型文字列。
initial_value_string	LONG VARCHAR	データベース変数の初期値の文字列表現。

備考

SET 文などを使用して行ったデータベーススコープ変数の値の更新は、データベースの再起動後は保持されません。また、更新された値はこのビューに反映されません。このビューには、初期/デフォルト値のみが表示されます。

権限

なし。

関連情報

SQL 変数 [118 ページ]
CREATE VARIABLE 文 [992 ページ]

1.7.1.86 SYSVIEW システムビュー

SYSVIEW システムビューの各ローは、データベース内のビューを示します。ビューの追加情報が SYSTAB システムビューにあります。このビューの基本となるシステムテーブルは ISYSVIEW です。

マテリアライズドビューの情報をより読みやすい形式にするには sa_materialized_view_info システムプロシージャを使用します。

カラム名	データ型	説明
view_object_id	UNSIGNED BIGINT	ビューのオブジェクト ID。
view_def	LONG VARCHAR	ビューの定義 (クエリ仕様)。

カラム名	データ型	説明
mv_build_type	TINYINT	内部使用のみ。
mv_refresh_type	TINYINT	ビューに定義された再表示タイプ。可能な値は IMMEDIATE (1) と MANUAL (2) です。
mv_use_in_optimization	TINYINT	クエリの最適化時にマテリアライズドビューを使用できるかどうか (0 = 最適化で使用できません、1 = 最適化で使用できます)。
mv_last_refreshed_at	TIMESTAMP	マテリアライズドビューが最後にリフレッシュされたローカル日時を示します。
mv_known_stale_at	TIMESTAMP	マテリアライズドビューが古くなったローカル時刻。この値は、基本となるいずれかのベーステーブルの変更が検出された時刻に対応します。0 の値は、ビューが新しいか、古いけれどもデータベースサーバから古いと印が付けられていないこと (古くなってからそのビューが使用されていない場合など) を示します。sa_materialized_view_info システムプロシージャを使用して、マテリアライズドビューのステータスを決定します。
mv_last_refreshed_tsn	UNSIGNED BIGINT	マテリアライズドビューをリフレッシュしたトランザクションに割り当てられたシーケンス番号。
mv_last_refreshed_at_utc	TIMESTAMP WITH TIME ZONE	マテリアライズドビューが最後にリフレッシュされた UTC 日時を示します。
mv_known_stale_at_utc	TIMESTAMP WITH TIME ZONE	マテリアライズドビューが古くなった UTC 時刻。この値は、基本となるいずれかのベーステーブルの変更が検出された時刻に対応します。0 の値は、ビューが新しいか、古いけれどもデータベースサーバから古いと印が付けられていないこと (古くなってからそのビューが使用されていない場合など) を示します。sa_materialized_view_info システムプロシージャを使用して、マテリアライズドビューのステータスを決定します。このカラムには、mv_last_refreshed_at が 0 の場合は 0 が含まれ、mv_last_refreshed_at が NULL の場合は NULL が含まれます。
check_option	CHAR(1)	ビューが WITH CHECK OPTION を使用している場合は Y を、使用していない場合は N を返します。

備考

スナップショットアイソレーションによってマテリアライズドビューがリフレッシュされると、mv_last_refreshed_at と mv_last_refreshed_tsn は、マテリアライズドビューの内容の処理中に使用されたローを修正した最初のトランザクションを参照します。

関連情報

sa_materialized_view_info システムプロシージャ [1554 ページ]
SYSTAB システムビュー [1842 ページ]
CREATE MATERIALIZED VIEW 文 [854 ページ]
REFRESH MATERIALIZED VIEW 文 [1252 ページ]
CREATE VIEW 文 [995 ページ]

1.7.1.87 SYSWEBSERVICE システムビュー

SYSWEBSERVICE システムビューの各ローは、個々の Web サービスを記述します。このビューの基本となるシステムテーブルは ISYSWEBSERVICE です。

カラム名	データ型	説明
service_id	UNSIGNED INT	Web サービスをユニークに識別する番号。
object_id	UNSIGNED BIGINT	Web サービスの ID。
service_name	CHAR(128)	Web サービスに割り当てられた名前。
service_type	VARCHAR(40)	サービスのタイプには、RAW、HTTP、XML、SOAP、DISH などがあります。
auth_required	CHAR(1)	すべての要求に有効なユーザ名とパスワードが必要かどうか。
secure_required	CHAR(1)	HTTP などのセキュリティ保護されていない接続を受け入れるのか、または HTTPS などのセキュリティ保護された接続だけを受け入れるのか。
url_path	CHAR(1)	URL の解釈を制御します。
user_id	UNSIGNED INT	認証が有効の場合、サービス使用のパーミッションを持つユーザまたはユーザグループを識別します。認証が無効の場合、要求を処理するときに使用するアカウントを指定します。
parameter	LONG VARCHAR	DISH サービスにインクルードされる SOAP サービスを識別するプレフィクス。
statement	LONG VARCHAR	要求に応答して常に実行される SQL 文。NULL の場合、各要求に含まれる任意の文が代わりに実行されます。DISH 型のサービスでは無視されます。
remarks	LONG VARCHAR	Web サービスに関する注記。ISYSREMARK システムテーブル内に格納されている値。

カラム名	データ型	説明
enabled	CHAR(1)	Web サービスが、現在、有効または無効のいずれであるかを示します (CREATE SERVICE を参照してください)。

1.7.2 統合ビュー

統合ビューは、ユーザから頻繁に要求される形式でデータを表示します。

たとえば、統合ビューにはよく一般に必要とされるジョインが用意されています。統合ビューは、基本となるシステムテーブルの未加工データをそのまま表示するシステムビューとは異なります。たとえば、システムビューのカラムの多くはわかりづらい ID 値ですが、統合ビューは読みやすい名前です。

SQL Anywhere では、システム所有者 (ユーザ SYS) が統合ビューを所有します。

このセクションの内容:

SYSCOLUMNS ビューの各ローは、カタログ内の各テーブルとビューのカラム 1 つを示します。

SYSFOREIGNKEYS 統合ビュー [1873 ページ]
SYSFOREIGNKEYS ビューの各ローは、カタログ内の各テーブルの外部キー 1 つを示します。

SYSINDEXES 統合ビュー [1874 ページ]
SYSINDEXES ビューの各ローは、データベース内のインデックス 1 つを示します。このビューの代わりに、SYSIDX と SYSIDXCOL のシステムビューを使用することもできます。

SYSOPTIONS 統合ビュー [1875 ページ]
SYSOPTIONS ビューの各ローは、SET コマンドを使用して作成されているオプション 1 つを示します。各ユーザはオプションごとに自分の設定を保存できます。また、ユーザ ID PUBLIC に対する設定は、自分の設定を持たないユーザが使うデフォルトの設定になります。

SYSPROCAUTH 統合ビュー [1876 ページ]
SYSPROCAUTH ビューの各ローは、個々のプロシージャに与えられた権限のセットを記述します。代わりに SYSPROCPERM システムビューを使用することもできます。

SYSPROCPARMS 統合ビュー [1876 ページ]
SYSPROCPARMS ビューの各ローは、データベース内のプロシージャに対するパラメータを示します。

SYSPROCS 統合ビュー [1877 ページ]
SYSPROCS ビューには、プロシージャまたはファンクションに記録されているプロシージャ名、ファンクション名、作成者名、コメントが表示されます。

SYSPUBLICATIONS 統合ビュー [1877 ページ]
SYSPUBLICATIONS ビューの各ローは、パブリケーションを示します。

SYSREMOTEOPTION2 統合ビュー [1878 ページ]
SYSREMOTEOPTION および SYSREMOTEOPTIONTYPE システムビューのカラムを一緒に結合して、読みやすい形式で表します。

SYSREMOTEOPTIONS 統合ビュー [1878 ページ]
SYSREMOTEOPTIONS システムビューの各ローは、メッセージリンクパラメータの値を示します。

SYSREMOTETYPES 統合ビュー [1879 ページ]
SYSREMOTETYPES ビューの各ローは、パブリッシャアドレスなど、個々のリモートメッセージタイプに関して記述します。

SYSREMOTEUSERS 統合ビュー [1879 ページ]
SYSREMOTEUSERS ビューの各ローは、REMOTE システム権限を持つユーザ ID (サブスクライバ) を示します。そのユーザに送信した、またはそのユーザから送信されたメッセージのステータスも合わせて示します。

SYSROLEGRANTS 統合ビュー [1880 ページ]
SYSROLEGRANTS システムビューは、SYSROLEGRANT システムビューと同様に、ロールのメンバーシップとメンバーシップのタイプに関する情報を格納します。ただし、SYSROLEGRANTS は (ID だけでなく) ロール名と被付与者名を含みます。このビューの基本となるシステムテーブルは、ISYSROLEGRANT と ISYSUSER です。

SYSSUBSCRIPTIONS 統合ビュー [1881 ページ]
REMOTE システム権限を持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションについて、各ローで示します。

SYSSYNC2 統合ビュー [1882 ページ]
SYSSYNC2 ビューから、機密情報を公開することなく、SYSSYNC システムビューで見つかったデータ (同期に関する情報) にパブリックアクセスできます。

SYSSYNCPUBLICATIONDEFAULTS 統合ビュー [1883 ページ]
SYSSYNCPUBLICATIONDEFAULTS は、同期に関連するパブリケーションに対応したデフォルトの同期設定のビューです。

SYSSYNCS 統合ビュー [1883 ページ]
SYSSYNCS ビューには、同期に関する情報が含まれます。

SYSSYNCSCRIPTS 統合ビュー [1884 ページ]
SYSSYNCSCRIPTS ビューの各ローは、スクリプト化されたアップロードに関するストアドプロシージャを識別します。このビューは SYSSYNCSCRIPT システムビューとほとんど同じですが、このビューでは、生データではなく、人間が読み取ることのできるフォーマットで値が表示されます。

SYSSYNCSUBSCRIPTIONS 統合ビュー [1884 ページ]
SYSSYNCSUBSCRIPTIONS ビューには、同期のサブスクリプションと関連付けられた同期設定が含まれます。

SYSSYNCUSERS 統合ビュー [1885 ページ]
同期ユーザに関連した同期の設定を表示します。

SYSTABAUTH 統合ビュー [1886 ページ]
SYSTABAUTH ビューには、SYSTABLEPERM システムビューの情報が表示されますが、より読みやすい形式です。

SYSTRIGGERS 統合ビュー [1886 ページ]
SYSTRIGGERS ビューの各ローは、データベース内のトリガ 1 つを示します。このテーブルには、参照トリガアクションを持つ外部キー定義に、自動的に作成されるトリガも含まれます (たとえば、ON DELETE CASCADE)。

SYSUSEROPTIONS 統合ビュー [1887 ページ]
SYSUSEROPTIONS ビューには、各ユーザの有効なオプション設定が入っています。ユーザにオプション設定がない場合、このビューには PUBLIC のオプション設定が表示されます。

SYSVIEWS 統合ビュー [1888 ページ]
SYSVIEWS ビューの各ローは、ビュー定義など、個々のビューの内容を示します。

1.7.2.1 ST_GEOMETRY_COLUMNS 統合ビュー

ST_GEOMETRY_COLUMNS システムビューの各ローは、データベースに定義されている空間カラムに関する記述です。

カラム名	データ型	説明
table_catalog	VARCHAR(128)	内部で使用されます。
table_schema	CHAR(128)	空間カラムを含むテーブルが属するスキーマの名前。これはテーブル所有者と同じです。
table_name	CHAR(128)	空間カラムを含むテーブルの名前。
column_name	CHAR(128)	空間カラムの名前。
srs_name	CHAR(128)	空間カラムに関連付けられている SRS の名前。SRS がカラムに関連付けられていない場合、srs_name は NULL です。

カラム名	データ型	説明
srs_id	INTEGER	空間カラムに関連付けられている SRS の SRID。
table_id	UNSIGNED INT	カラムが属するテーブルの数値識別子。
column_id	UNSIGNED INT	カラムの数値識別子。
geometry_type_name	VARCHAR(32767)	カラムに含まれているジオメトリの空間データ型 (たとえば、ST_Point、ST_Geometry など)。

1.7.2.2 ST_SPATIAL_REFERENCE_SYSTEMS 統合ビュー

ST_SPATIAL_REFERENCE_SYSTEMS システムビューの各ローは、データベースに定義されている SRS に関する記述です。このビューには、SYSSPATIALREFERENCESYSTEM システムビューとは若干異なる量の情報があります。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	システムでのみ使用。
owner	UNSIGNED INT	SRS の所有者。
srs_name	CHAR(128)	SRS の名前。
srs_id	INTEGER	空間参照系の数値識別子 (SRID)。

カラム名	データ型	説明
srs_type	CHAR(11)	SQL/MM 標準による定義に従った SRS のタイプ。次のいずれかの値を取ります。 **GEOGRAPHIC** 緯度と経度 (および標高) の軸を持つ、測地参照される座標系に基づく SRS 用です。これらの SRS は、PLANAR タイプまたは ROUND EARTH タイプです。 **PROJECTED** 緯度と経度の軸を持たず、測地参照される座標系に基づく SRS 用です。これらの SRS は PLANAR タイプです。 **ENGINEERING** 測地参照されない座標系に基づく SRS 用です。これらの SRS は PLANAR タイプです。 **GEOCENTRIC** サポートされていません。 **COMPOUND** サポートされていません。 **VERTICAL** サポートされていません。 srs_type が空の場合、タイプは指定されません。
round_earth	CHAR(1)	SRS タイプが ROUND EARTH (Y) または PLANAR (N) のいずれであるか。
axis_order	CHAR(12)	データベースサーバが緯度と経度に関連してポイントを解釈する方法を記述します (たとえば、ST_Lat メソッドと ST_Long メソッドを使用する場合)。非地理的空間参照系の場合、軸順序は x/y/z/m。地理的空間参照系の場合、デフォルトの軸順序は long/lat/z/m であり、lat/long/z/m もサポートされます。
snap_to_grid	DOUBLE	データベースサーバが計算を実行するときに使用するグリッドのサイズを定義します。
tolerance	DOUBLE	ポイントを比較するときに使用する精度を定義します。
semi_major_axis	DOUBLE	ROUND EARTH SRS の楕円の中心から赤道までの距離です。

カラム名	データ型	説明
semi_minor_axis	DOUBLE	ROUND EARTH SRS の楕円の中心から両極までの距離です。
inv_flattening	DOUBLE	ROUND EARTH SRS の楕円に使用される逆扁平率です。これは、次の式によって算出される比率となります。`1/f = (semi-major-axis) / (semi-major-axis - semi-minor-axis)`
min_x	DOUBLE	座標において許される x の最小値です。
max_x	DOUBLE	座標において許される x の最大値です。
min_y	DOUBLE	座標において許される y の最小値です。
max_y	DOUBLE	座標において許される y の最大値です。
min_z	DOUBLE	座標において許される z の最小値です。
max_z	DOUBLE	座標において許される z の最大値です。
min_m	DOUBLE	座標において許される m の最小値です。
max_m	DOUBLE	座標において許される m の最大値です。
min_lat	DOUBLE	座標において許される緯度の最小値です。
max_lat	DOUBLE	座標において許される緯度の最大値です。
min_long	DOUBLE	座標において許される経度の最小値です。
max_long	DOUBLE	座標において許される経度の最大値です。
organization	LONG VARCHAR	空間参照系によって使用される座標系を作成した組織の名前。
organization_coordsys_id	INTEGER	座標系を作成した組織が座標系に対して指定した ID。
linear_unit_of_measure	CHAR(128)	SRS によって使用される線形測定単位。
angular_unit_of_measure	CHAR(128)	SRS によって使用される角度測定単位。
polygon_format	LONG VARCHAR	多角形のリングの方向。CounterClockwise、ClockWise、または EvenOdd のいずれかです。
storage_format	LONG VARCHAR	データが正規化フォーマット (Internal)、非正規化フォーマット (Original)、または両方 (Mixed) のいずれで格納されるか。
definition	LONG VARCHAR	追加の定義設定。
transform_definition	LONG VARCHAR	この SRS から別の SRS にデータを変換するときに使用される変換定義設定です。
description	LONG VARCHAR	SRS の説明。

1.7.2.3 ST_UNITS_OF_MEASURE 統合ビュー

ST_UNITS_OF_MEASURE システムビューの各ローは、データベースに定義されている測定単位に関する記述です。このビューには、SYSUNITOFMEASURE システムビューよりも多い情報があります。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	システムでのみ使用。
owner	UNSIGNED INT	測定単位の所有者。
unit_name	CHAR(128)	測定単位の名前。
unit_type	CHAR(7)	角度または線形。
conversion_factor	DOUBLE	測定単位の換算係数。
description	LONG VARCHAR	測定単位の説明。

1.7.2.4 SYSARTICLECOLS 統合ビュー

SYSARTICLECOLS ビューの各ローは、アーティクル内のカラムを識別します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSARTICLECOLS"
  as select p.publication_name,t.table_name,c.column_name
    from SYS.ISYSARTICLECOL as ac
      join SYS.ISYSPUBLICATION as p on p.publication_id = ac.publication_id
      join SYS.ISYSTAB as t on t.table_id = ac.table_id
      join SYS.ISYSTABCOL as c on c.table_id = ac.table_id
      and c.column_id = ac.column_id
```

関連情報

SYSARTICLECOL システムビュー [1793 ページ]
SYSPUBLICATION システムビュー [1828 ページ]
SYSTAB システムビュー [1842 ページ]
SYSTABCOL システムビュー [1845 ページ]

1.7.2.5 SYSARTICLES 統合ビュー

SYSARTICLES ビューの各ローは、パブリケーション内のアーティクルを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSARTICLES"
  as select u1.user_name as publication_owner,p.publication_name,
    u2.user_name as table_owner,t.table_name,
    a.where_expr,a.subscribe_by_expr,a.alias
    from SYS.ISYSARTICLE as a
      join SYS.ISYSPUBLICATION as p on(a.publication_id = p.publication_id)
      join SYS.ISYSTAB as t on(a.table_id = t.table_id)
      join SYS.ISYSUSER as u1 on(p.creator = u1.user_id)
      join SYS.ISYSUSER as u2 on(t.creator = u2.user_id)
```

関連情報

SYSARTICLE システムビュー [1793 ページ]
SYSPUBLICATION システムビュー [1828 ページ]
SYSTAB システムビュー [1842 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.6 SYSCAPABILITIES 統合ビュー

SYSCAPABILITIES ビューの各ローは、リモートデータベースサーバの機能のステータスを示します。このビューのデータは ISYSCAPABILITY システムテーブルから取得されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCAPABILITIES"
  as select
ISYSCAPABILITY.capid,ISYSCAPABILITY.srvid,property('RemoteCapability',ISYSCAPABILITY
.capid) as capname,ISYSCAPABILITY.capvalue
    from SYS.ISYSCAPABILITY
```

関連情報

SYSCAPABILITY システムビュー [1794 ページ]
SYSCAPABILITYNAME システムビュー [1794 ページ]

1.7.2.7 SYSCATALOG 統合ビュー

SYSCATALOG ビューの各ローは、システムテーブルを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCATALOG"( creator,
  tname,dbspacename,tabletype,ncols,primary_key,"check",
  remarks )
  as select u.user_name,tab.table_name,dbs.dbspace_name,
    if tab.table_type_str = 'BASE' then 'TABLE' else tab.table_type_str endif,
    (select count() from SYS.ISYSTABCOL
      where ISYSTABCOL.table_id = tab.table_id),
    if ix.index_id is null then 'N' else 'Y' endif,
    null,
    rmk.remarks
    from SYS.SYSTAB as tab
      join SYS.ISYSDBSPACE as dbs on(tab.dbspace_id = dbs.dbspace_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator
      left outer join SYS.ISYSIDX as ix on(tab.table_id = ix.table_id and
ix.index_id = 0)
      left outer join SYS.ISYSREMARK as rmk on(tab.object_id = rmk.object_id)
```

関連情報

SYSTAB システムビュー [1842 ページ]
SYSTABCOL システムビュー [1845 ページ]
SYSDBSPACE システムビュー [1798 ページ]
SYSUSER システムビュー [1855 ページ]
SYSIDX システムビュー [1810 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.2.8 SYSCOLAUTH 統合ビュー

SYSCOLAUTH ビューの各ローは、カラムに付与されている一連の権限 (UPDATE、SELECT、または REFERENCES) を示します。

SYSCOLAUTH ビューは、SYSCOLPERM システムビュー内のデータを読みやすい形式で表示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCOLAUTH"( grantor,grantee,creator,tname,colname,
  privilege_type,is_grantable )
  as select u1.user_name,u2.user_name,u3.user_name,tab.table_name,
    col.column_name,cp.privilege_type,cp.is_grantable
    from SYS.ISYSCOLPERM as cp
      join SYS.ISYSUSER as u1 on u1.user_id = cp.grantor
      join SYS.ISYSUSER as u2 on u2.user_id = cp.grantee
```

```
      join SYS.ISYSTAB as tab on tab.table_id = cp.table_id
      join SYS.ISYSUSER as u3 on u3.user_id = tab.creator
      join SYS.ISYSTABCOL as col on col.table_id = cp.table_id
      and col.column_id = cp.column_id
```

関連情報

SYSCOLPERM システムビュー [1795 ページ]
SYSTABCOL システムビュー [1845 ページ]
SYSUSER システムビュー [1855 ページ]
SYSTAB システムビュー [1842 ページ]

1.7.2.9 SYSCOLSTATS 統合ビュー

SYSCOLSTATS ビューには、オプティマイザによって使用されるカラム統計が、ヒストグラムとして格納されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCOLSTATS" AS SELECT u.user_name, t.table_name, c.column_name,
s.format_id,
   dateadd(mi, PROPERTY('TimeZoneAdjustment'), s.update_time) as update_time,
s.density, s.max_steps, s.actual_steps,
   s.step_values, s.frequencies , TODATETIMEOFFSET( s.update_time, 0 ) as
update_time_utc
FROM SYS.ISYSCOLSTAT s
JOIN SYS.ISYSTABCOL c on (s.table_id = c.table_id and s.column_id = c.column_id)
JOIN SYS.ISYSTAB t on (t.table_id = c.table_id)
JOIN SYS.ISYSUSER u on (u.user_id = t.creator)
```

関連情報

SYSCOLSTAT システムビュー [1796 ページ]
SYSTABCOL システムビュー [1845 ページ]
SYSTAB システムビュー [1842 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.10 SYSCOLUMNS 統合ビュー

SYSCOLUMNS ビューの各ローは、カタログ内の各テーブルとビューのカラム 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCOLUMNS"( creator,cname,tname,coltype,nulls,length,
  syslength,in_primary_key,colno,default_value,
  column_kind,remarks )
  as select u.user_name,col.column_name,tab.table_name,dom.domain_name,
    col.nulls,col.width,col.scale,if ixcol.sequence is null then 'N' else 'Y'
endif,col.column_id,
    col."default",col.column_type,rmk.remarks
    from SYS.SYSTABCOL as col
      left outer join SYS.ISYSIDXCOL as ixcol on(col.table_id = ixcol.table_id and
col.column_id = ixcol.column_id and ixcol.index_id = 0)
      join SYS.ISYSTAB as tab on(tab.table_id = col.table_id)
      join SYS.ISYSDOMAIN as dom on(dom.domain_id = col.domain_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator
      left outer join SYS.ISYSREMARK as rmk on(col.object_id = rmk.object_id)
```

関連情報

SYSTABCOL システムビュー [1845 ページ]
SYSIDXCOL システムビュー [1811 ページ]
SYSTAB システムビュー [1842 ページ]
SYSDOMAIN システムビュー [1800 ページ]
SYSUSER システムビュー [1855 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.2.11 SYSFOREIGNKEYS 統合ビュー

SYSFOREIGNKEYS ビューの各ローは、カタログ内の各テーブルの外部キー 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSFOREIGNKEYS"( foreign_creator,
  foreign_tname,
  primary_creator,primary_tname,role,columns )
  as select fk_up.user_name,fk_tab.table_name,pk_up.user_name,
    pk_tab.table_name,ix.index_name,
    (select list(string(fk_col.column_name,' IS ',
      pk_col.column_name)
      order by fkc.table_id,fkc.index_id,fkc."sequence")
      from SYS.ISYSIDXCOL as fkc
        join SYS.ISYSTABCOL as fk_col on(
        fkc.table_id = fk_col.table_id
        and fkc.column_id = fk_col.column_id)
        ,SYS.ISYSTABCOL as pk_col
```

```
      where fkc.table_id = fk.foreign_table_id
      and fkc.index_id = fk.foreign_index_id
      and pk_col.table_id = fk.primary_table_id
      and pk_col.column_id = fkc.primary_column_id)
    from SYS.ISYSFKEY as fk
      join SYS.ISYSTAB as fk_tab on fk_tab.table_id = fk.foreign_table_id
      join SYS.ISYSUSER as fk_up on fk_up.user_id = fk_tab.creator
      join SYS.ISYSTAB as pk_tab on pk_tab.table_id = fk.primary_table_id
      join SYS.ISYSUSER as pk_up on pk_up.user_id = pk_tab.creator
      join SYS.ISYSIDX as ix on ix.table_id = fk.foreign_table_id and ix.index_id =
fk.foreign_index_id
```

関連情報

SYSTAB システムビュー [1842 ページ]
SYSTABCOL システムビュー [1845 ページ]
SYSIDX システムビュー [1810 ページ]
SYSIDXCOL システムビュー [1811 ページ]
SYSFKEY システムビュー [1806 ページ]
SYSUSER システムビュー [1855 ページ]
SYSDOMAIN システムビュー [1800 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.2.12 SYSINDEXES 統合ビュー

SYSINDEXES ビューの各ローは、データベース内のインデックス 1 つを示します。このビューの代わりに、SYSIDX と SYSIDXCOL のシステムビューを使用することもできます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSINDEXES"( icreator,
  iname,fname,creator,tname,indextype,
  colnames,interval,level_num )
  as select u.user_name,idx.index_name,dbs.dbspace_name,u.user_name,
    tab.table_name,
    case idx.index_category
    when 1 then 'Primary Key'
    when 2 then 'Foreign Key'
    when 3 then(
      if idx."unique" = 4 then 'Non-unique'
      else if idx."unique" = 2 then 'UNIQUE constraint'
        else if idx."unique" = 5 then 'UNIQUE NULLS NOT DISTINCT'
          else 'UNIQUE'
          endif
        endif
      endif) when 4 then 'Text Index' end,(select list(string(c.column_name,
      if ixc."order" = 'A' then ' ASC' else ' DESC' endif) order by
      ixc.table_id asc,ixc.index_id asc,ixc.sequence asc)
      from SYS.ISYSIDXCOL as ixc
        join SYS.ISYSTABCOL as c on(
        c.table_id = ixc.table_id
```

```
        and c.column_id = ixc.column_id)
      where ixc.index_id = idx.index_id
      and ixc.table_id = idx.table_id),
    0,0
    from SYS.ISYSTAB as tab
      join SYS.ISYSDBSPACE as dbs on(tab.dbspace_id = dbs.dbspace_id)
      join SYS.ISYSIDX as idx on(idx.table_id = tab.table_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator
```

関連情報

SYSIDX システムビュー [1810 ページ]
SYSTABCOL システムビュー [1845 ページ]
SYSTAB システムビュー [1842 ページ]
SYSDBSPACE システムビュー [1798 ページ]
SYSIDXCOL システムビュー [1811 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.13 SYSOPTIONS 統合ビュー

SYSOPTIONS ビューの各ローは、SET コマンドを使用して作成されているオプション 1 つを示します。各ユーザはオプションごとに自分の設定を保存できます。また、ユーザ ID PUBLIC に対する設定は、自分の設定を持たないユーザが使うデフォルトの設定になります。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSOPTIONS"( user_name,"option",setting )
  as select u.user_name,opt."option",opt.setting
    from SYS.ISYSOPTION as opt
      join SYS.ISYSUSER as u on opt.user_id = u.user_id
```

関連情報

SYSOPTION システムビュー [1822 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.14 SYSPROCAUTH 統合ビュー

SYSPROCAUTH ビューの各ローは、個々のプロシージャに与えられた権限のセットを記述します。代わりに SYSPROCPERM システムビューを使用することもできます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSPROCAUTH"( grantee,
  creator,procname )
  as select u1.user_name,u2.user_name,p.proc_name
    from SYS.ISYSPROCEDURE as p
      join SYS.ISYSPROCPERM as pp on(p.proc_id = pp.proc_id)
      join SYS.ISYSUSER as u1 on u1.user_id = pp.grantee
      join SYS.ISYSUSER as u2 on u2.user_id = p.creator
```

関連情報

SYSPROCEDURE システムビュー [1824 ページ]
SYSPROCPERM システムビュー [1827 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.15 SYSPROCPARMS 統合ビュー

SYSPROCPARMS ビューの各ローは、データベース内のプロシージャに対するパラメータを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSPROCPARMS"( creator,
  procname,parmname,parm_id,parmtype,parmmode,parmdomain,
  length,scale,"default",user_type )
  as select up.user_name,p.proc_name,pp.parm_name,pp.parm_id,pp.parm_type,
    if pp.parm_mode_in = 'Y' and pp.parm_mode_out = 'N' then 'IN'
    else if pp.parm_mode_in = 'N' and pp.parm_mode_out = 'Y' then 'OUT'
      else 'INOUT'
      endif
    endif,dom.domain_name,pp.width,pp.scale,pp."default",ut.type_name
    from SYS.SYSPROCPARM as pp
      join SYS.ISYSPROCEDURE as p on p.proc_id = pp.proc_id
      join SYS.ISYSUSER as up on up.user_id = p.creator
      join SYS.ISYSDOMAIN as dom on dom.domain_id = pp.domain_id
      left outer join SYS.ISYSUSERTYPE as ut on ut.type_id = pp.user_type
```

関連情報

SYSPROCPARM システムビュー [1825 ページ]

SYSPROCEDURE システムビュー [1824 ページ]
SYSUSER システムビュー [1855 ページ]
SYSDOMAIN システムビュー [1800 ページ]
SYSUSERTYPE システムビュー [1858 ページ]

1.7.2.16 SYSPROCS 統合ビュー

SYSPROCS ビューには、プロシージャまたはファンクションに記録されているプロシージャ名、ファンクション名、作成者名、コメントが表示されます。

ビューを構成するテーブルとカラムは、以下の ALTER VIEW 文で示されます。

```
ALTER VIEW "SYS"."SYSPROCS"( creator,
  procname,remarks )
  as select u.user_name,p.proc_name,r.remarks
    from SYS.ISYSPROCEDURE as p
      join SYS.ISYSUSER as u on u.user_id = p.creator
      left outer join SYS.ISYSREMARK as r on(p.object_id = r.object_id)
```

関連情報

SYSPROCEDURE システムビュー [1824 ページ]
SYSUSER システムビュー [1855 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.2.17 SYSPUBLICATIONS 統合ビュー

SYSPUBLICATIONS ビューの各ローは、パブリケーションを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSPUBLICATIONS"
  as select u.user_name as creator,
    p.publication_name,
    r.remarks,
    p.type,
    case p.sync_type
    when 0 then 'logscan'
    when 1 then 'scripted upload'
    when 2 then 'download only'
    else 'invalid'
    end as sync_type
    from SYS.ISYSPUBLICATION as p
      join SYS.ISYSUSER as u on u.user_id = p.creator
      left outer join SYS.ISYSREMARK as r on(p.object_id = r.object_id)
```

関連情報

SYSPUBLICATION システムビュー [1828 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.2.18 SYSREMOTEOPTION2 統合ビュー

SYSREMOTEOPTION および SYSREMOTEOPTIONTYPE システムビューのカラムを一緒に結合して、読みやすい形式で表します。

SELECT ANY TABLE システム権限のないユーザの場合、設定されているカラムの値は非表示になります。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTEOPTION2"
  as select ISYSREMOTEOPTION.option_id,
    ISYSREMOTEOPTION.user_id,
    SYS.HIDE_FROM_NON_DBA(ISYSREMOTEOPTION.setting) as setting
    from SYS.ISYSREMOTEOPTION
```

関連情報

SYSREMOTEOPTION システムビュー [1829 ページ]

1.7.2.19 SYSREMOTEOPTIONS 統合ビュー

SYSREMOTEOPTIONS システムビューの各ローは、メッセージリンクパラメータの値を示します。

SELECT ANY TABLE システム権限のないユーザの場合、設定されているカラムの値は非表示になります。
SYSREMOTEOPTION2 ビューは、公開しても問題のないデータへのパブリックアクセスを提供します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTEOPTIONS"
  as select srt.type_name,
    sup.user_name,
    srot."option",
    SYS.HIDE_FROM_NON_DBA(sro.setting) as setting
    from SYS.ISYSREMOTETYPE as srt
      ,SYS.ISYSREMOTEOPTIONTYPE as srot
      ,SYS.ISYSREMOTEOPTION as sro
      ,SYS.ISYSUSER as sup
    where srt.type_id = srot.type_id
    and srot.option_id = sro.option_id
    and sro.user_id = sup.user_id
```

関連情報

SYSREMOTETYPE システムビュー [1830 ページ]
SYSREMOTEOPTIONTYPE システムビュー [1829 ページ]
SYSREMOTEOPTION システムビュー [1829 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.20 SYSREMOTETYPES 統合ビュー

SYSREMOTETYPES ビューの各ローは、パブリッシャアドレスなど、個々のリモートメッセージタイプに関して記述します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTETYPES"
  as select rt.type_id,rt.type_name,rt.publisher_address,rm.remarks
    from SYS.ISYSREMOTETYPE as rt
      left outer join SYS.ISYSREMARK as rm on(rt.object_id = rm.object_id)
```

関連情報

SYSREMOTETYPE システムビュー [1830 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.2.21 SYSREMOTEUSERS 統合ビュー

SYSREMOTEUSERS ビューの各ローは、REMOTE システム権限を持つユーザ ID (サブスクライバ) を示します。そのユーザに送信した、またはそのユーザから送信されたメッセージのステータスも合わせて示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTEUSERS" AS SELECT u.user_name, r.consolidate,
t.type_name, r.address, r.frequency, r.send_time,
(if r.frequency = 'A' then NULL
else if r.frequency = 'P' then
if r.time_sent IS NULL then CURRENT TIMESTAMP
else (select min( minutes( dateadd(mi, PROPERTY('TimeZoneAdjustment'),
a.time_sent),
60*hour(a.send_time) + minute( seconds( a.send_time, 59 ) ) ) )
FROM SYS.ISYSREMOTEUSER a WHERE a.frequency = 'P' AND a.send_time = r.send_time )
endif
else if CURRENT DATE + r.send_time > coalesce( dateadd(mi,
PROPERTY('TimeZoneAdjustment'), r.time_sent), CURRENT TIMESTAMP)
then CURRENT DATE + r.send_time
else CURRENT DATE + r.send_time + 1
```

```
endif   endif  endif) as next_send, r.log_send ,
dateadd(mi, PROPERTY('TimeZoneAdjustment'), r.time_sent)
as time_sent , r.log_sent, r.confirm_sent, r.send_count, r.resend_count,
dateadd(mi, PROPERTY('TimeZoneAdjustment'), r.time_received) as time_received ,
r.log_received, r.confirm_received, r.receive_count, r.rereceive_count ,
TODATETIMEOFFSET( r.time_sent, 0 ) as time_sent_utc ,
TODATETIMEOFFSET( r.time_received, 0 ) as time_received_utc
FROM SYS.ISYSREMOTEUSER r JOIN SYS.ISYSUSER u ON ( u.user_id = r.user_id ) JOIN
SYS.ISYSREMOTETYPE t ON ( t.type_id = r.type_id )
```

関連情報

SYSREMOTEUSER システムビュー [1830 ページ]
SYSUSER システムビュー [1855 ページ]
SYSREMOTETYPE システムビュー [1830 ページ]

1.7.2.22 SYSROLEGRANTS 統合ビュー

SYSROLEGRANTS システムビューは、SYSROLEGRANT システムビューと同様に、ロールのメンバーシップとメンバーシップのタイプに関する情報を格納します。ただし、SYSROLEGRANTS は (ID だけでなく) ロール名と被付与者名を含みます。このビューの基本となるシステムテーブルは、ISYSROLEGRANT と ISYSUSER です。

カラム名	データ型	説明
grant_id	UNSIGNED INT	各 GRANT 文の識別に使用する ID。
role_id	UNSIGNED INT	ISYSUSER に基づいて、付与されるロールの ID。
role_name	CHAR(128)	ロールの名前。
grantee	UNISIGNED INT	ISYSUSER に基づいて、ロールを付与されるユーザの ID。
grantee_name	CHAR(128)	被付与者の名前。

カラム名	データ型	説明
grant_type	TINYINT	3 ビットを使用して付与のタイプを表します。右端のビットは、権限が付与されているかどうかを示します。2 桁目は、管理権限が付与されているかどうかを示します。3 桁目は、システム権限が継承可能かどうかを示します。 **001** 権限が付与されていますが、継承可能ではなく、管理権限もありません。DBA と REMOTE DBA を除く、継承不可のレガシー権限にのみ適用。 **101** 権限が付与されており、継承可能ですが、管理権限はありません。 **110** 管理権限のみが付与されています。 **111** 権限が付与されており、継承可能で、管理権限があります。
grant_scope	TINYINT	SET USER および CHANGE PASSWORD によって使用され、付与のスコープを設定します。値は、次のうちの 1 つまたは複数です。 **1** ANY **2** ユーザリスト **4** ロールリスト
grantor	CHAR(128)	付与者の名前。

1.7.2.23 SYSSUBSCRIPTIONS 統合ビュー

REMOTE システム権限を持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションについて、各ローで示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSUBSCRIPTIONS"
  as select p.publication_name,u.user_name,s.subscribe_by,s.created,
    s.started
```

```
    from SYS.ISYSSUBSCRIPTION as s
      join SYS.ISYSPUBLICATION as p on(p.publication_id = s.publication_id)
      join SYS.ISYSUSER as u on u.user_id = s.user_id
```

関連情報

SYSSUBSCRIPTION システムビュー [1839 ページ]
SYSPUBLICATION システムビュー [1828 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.24 SYSSYNC2 統合ビュー

SYSSYNC2 ビューから、機密情報を公開することなく、SYSSYNC システムビューで見つかったデータ (同期に関する情報) にパブリックアクセスできます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

server_connect とオプションカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。

```
ALTER VIEW "SYS"."SYSSYNC2"
  as select "ISYSSYNC"."sync_id",
    "ISYSSYNC"."type",
    "ISYSSYNC"."publication_id",
    "ISYSSYNC"."progress",
    "ISYSSYNC"."site_name",
    if "ISYSSYNC"."option" is null then null else '***' endif as "option",
    if "ISYSSYNC"."server_connect" is null then null else '***' endif as
"server_connect",
    "ISYSSYNC"."server_conn_type",
    "ISYSSYNC"."last_download_time",
    "ISYSSYNC"."last_upload_time",
    "ISYSSYNC"."created",
    "ISYSSYNC"."log_sent",
    "ISYSSYNC"."generation_number",
    "ISYSSYNC"."extended_state",
    "ISYSSYNC"."script_version",
    "ISYSSYNC"."subscription_name",
    "ISYSSYNC"."server_protocol"
    from "SYS"."ISYSSYNC"
```

関連情報

SYSSYNC システムビュー [1839 ページ]

1.7.2.25 SYSSYNCPUBLICATIONDEFAULTS 統合ビュー

SYSSYNCPUBLICATIONDEFAULTS は、同期に関連するパブリケーションに対応したデフォルトの同期設定のビューです。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。

server_connect とオプションカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。

```
ALTER VIEW "SYS"."SYSSYNCPUBLICATIONDEFAULTS"
  as select "s"."sync_id",
    "p"."publication_name",
    "s"."option",
    "s"."server_connect",
    "s"."server_conn_type"
    from "SYS"."SYSSYNC2" as "s" join "SYS"."SYSPUBLICATION" as "p"
on("p"."publication_id" = "s"."publication_id") where
    "s"."site_name" is null
```

関連情報

SYSSYNC2 統合ビュー [1882 ページ]
SYSPUBLICATION システムビュー [1828 ページ]

1.7.2.26 SYSSYNCS 統合ビュー

SYSSYNCS ビューには、同期に関する情報が含まれます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

server_connect とオプションカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。

この連結ビューの基本となるビューは、SYSSYNC2 です。

```
ALTER VIEW "SYS"."SYSSYNCS"
  as select "p"."publication_name","s"."progress","s"."site_name",
    "s"."option",
    "s"."server_connect",
    "s"."server_conn_type","s"."last_download_time",
    "s"."last_upload_time","s"."created","s"."log_sent","s"."generation_number",
    "s"."extended_state"
    from "SYS"."SYSSYNC2" as "s"
      left outer join "SYS"."SYSPUBLICATION" as "p"
      on "p"."publication_id" = "s"."publication_id"
```

関連情報

SYSSYNC2 統合ビュー [1882 ページ]
SYSPUBLICATION システムビュー [1828 ページ]

1.7.2.27 SYSSYNCSCRIPTS 統合ビュー

SYSSYNCSCRIPTS ビューの各ローは、スクリプト化されたアップロードに関するストアドプロシージャを識別します。このビューは SYSSYNCSCRIPT システムビューとほとんど同じですが、このビューでは、生データではなく、人間が読み取ることのできるフォーマットで値が表示されます。

```
ALTER VIEW "SYS"."SYSSYNCSCRIPTS"
  as select p.publication_name,
    t.table_name,
    case s.type
    when 0 then 'upload insert'
    when 1 then 'upload delete'
    when 2 then 'upload update'
    else 'unknown'
    end as type,
    c.proc_name
    from SYS.ISYSSYNCSCRIPT as s
      join SYS.ISYSPUBLICATION as p on p.object_id = s.pub_object_id
      join SYS.ISYSTAB as t on t.object_id = s.table_object_id
      join SYS.ISYSPROCEDURE as c on c.object_id = s.proc_object_id
```

関連情報

SYSSYNCSCRIPT システムビュー [1841 ページ]
SYSPUBLICATION システムビュー [1828 ページ]
SYSTAB システムビュー [1842 ページ]
SYSPROCEDURE システムビュー [1824 ページ]

1.7.2.28 SYSSYNCSUBSCRIPTIONS 統合ビュー

SYSSYNCSUBSCRIPTIONS ビューには、同期のサブスクリプションと関連付けられた同期設定が含まれます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。

server_connect とオプションカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。

```
ALTER VIEW "SYS"."SYSSYNCSUBSCRIPTIONS"
  as select "s"."sync_id",
    "p"."publication_name",
    "s"."progress",
```

```
    "s"."site_name",
    "s"."option",
    "s"."server_connect",
    "s"."server_conn_type",
    "s"."last_download_time",
    "s"."last_upload_time",
    "s"."created",
    "s"."log_sent",
    "s"."generation_number",
    "s"."extended_state"
    from "SYS"."SYSSYNC2" as "s" join "SYS"."SYSPUBLICATION" as "p"
on("p"."publication_id" = "s"."publication_id")
    where "s"."publication_id" is not null and
    "s"."site_name" is not null and exists
    (select 1 from "SYS"."SYSSYNCUSERS" as "u"
      where "s"."site_name" = "u"."site_name")
```

関連情報

SYSSYNC2 統合ビュー [1882 ページ]
SYSPUBLICATION システムビュー [1828 ページ]
SYSSYNCUSERS 統合ビュー [1885 ページ]

1.7.2.29 SYSSYNCUSERS 統合ビュー

同期ユーザに関連した同期の設定を表示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。

server_connect とオプションカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。

```
ALTER VIEW "SYS"."SYSSYNCUSERS"
  as select "SYSSYNC2"."sync_id",
    "SYSSYNC2"."site_name",
    "SYSSYNC2"."option",
    "SYSSYNC2"."server_connect",
    "SYSSYNC2"."server_conn_type"
    from "SYS"."SYSSYNC2" where
    "SYSSYNC2"."publication_id" is null
```

関連情報

SYSSYNC2 統合ビュー [1882 ページ]

1.7.2.30 SYSTABAUTH 統合ビュー

SYSTABAUTH ビューには、SYSTABLEPERM システムビューの情報が表示されますが、より読みやすい形式です。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSTABAUTH"( grantor,
  grantee,screator,stname,tcreator,ttname,
  selectauth,insertauth,deleteauth,
  updateauth,updatecols,alterauth,referenceauth,
  loadauth,truncateauth )
  as select u1.user_name,u2.user_name,u3.user_name,tab1.table_name,
    u4.user_name,tab2.table_name,tp.selectauth,tp.insertauth,
    tp.deleteauth,tp.updateauth,tp.updatecols,tp.alterauth,
    tp.referenceauth,tp.loadauth,tp.truncateauth
    from SYS.ISYSTABLEPERM as tp
      join SYS.ISYSUSER as u1 on u1.user_id = tp.grantor
      join SYS.ISYSUSER as u2 on u2.user_id = tp.grantee
      join SYS.ISYSTAB as tab1 on tab1.table_id = tp.stable_id
      join SYS.ISYSUSER as u3 on u3.user_id = tab1.creator
      join SYS.ISYSTAB as tab2 on tab2.table_id = tp.stable_id
      join SYS.ISYSUSER as u4 on u4.user_id = tab2.creator
```

関連情報

SYSTABLEPERM システムビュー [1847 ページ]
SYSUSER システムビュー [1855 ページ]
SYSTAB システムビュー [1842 ページ]

1.7.2.31 SYSTRIGGERS 統合ビュー

SYSTRIGGERS ビューの各ローは、データベース内のトリガ 1 つを示します。このテーブルには、参照トリガアクションを持つ外部キー定義に、自動的に作成されるトリガも含まれます (たとえば、ON DELETE CASCADE)。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSTRIGGERS"( owner,
  trigname,tname,event,trigtime,trigdefn )
  as select u.user_name,trig.trigger_name,tab.table_name,
    if trig.event = 'I' then 'INSERT'
    else if trig.event = 'U' then 'UPDATE'
      else if trig.event = 'C' then 'UPDATE'
        else if trig.event = 'D' then 'DELETE'
          else if trig.event = 'A' then 'INSERT,DELETE'
            else if trig.event = 'B' then 'INSERT,UPDATE'
              else if trig.event = 'E' then 'DELETE,UPDATE'
                else 'INSERT,DELETE,UPDATE'
                endif
              endif
            endif
```

```
          endif
        endif
      endif
    endif,if trig.trigger_time = 'B' or trig.trigger_time = 'P' then 'BEFORE'
    else if trig.trigger_time = 'A' or trig.trigger_time = 'S' then 'AFTER'
      else if trig.trigger_time = 'R' then 'RESOLVE'
        else 'INSTEAD OF'
        endif
      endif
    endif,trig.trigger_defn
    from SYS.ISYSTRIGGER as trig
      join SYS.ISYSTAB as tab on(tab.table_id = trig.table_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator where
    trig.foreign_table_id is null
```

関連情報

SYSTRIGGER システムビュー [1852 ページ]
SYSTAB システムビュー [1842 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.32 SYSUSEROPTIONS 統合ビュー

SYSUSEROPTIONS ビューには、各ユーザの有効なオプション設定が入っています。ユーザにオプション設定がない場合、このビューには PUBLIC のオプション設定が表示されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSEROPTIONS"( user_name,
  "option",setting )
  as select u.user_name,
    o."option",
    isnull((select s.setting
      from SYS.ISYSOPTION as s
      where s.user_id = u.user_id
      and s."option" = o."option"),
    o.setting)
    from SYS.SYSOPTIONS as o,SYS.ISYSUSER as u
    where o.user_name = 'PUBLIC'
```

関連情報

SYSOPTIONS 統合ビュー [1875 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.2.33 SYSVIEWS 統合ビュー

SYSVIEWS ビューの各ローは、ビュー定義など、個々のビューの内容を示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSVIEWS"( vcreator,
  viewname,viewtext )
  as select u.user_name,t.table_name,v.view_def
    from SYS.ISYSTAB as t
      join SYS.ISYSVIEW as v on(t.object_id = v.view_object_id)
      join SYS.ISYSUSER as u on(u.user_id = t.creator)
```

関連情報

SYSTAB システムビュー [1842 ページ]
SYSVIEW システムビュー [1860 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.3 互換ビュー

互換ビューは、バージョン 10 以前のソフトウェアとの互換性のために提供されるビューです。将来的なリリースで互換ビューを減らす可能性があるため、可能な場合、システムビューと統合ビューを使用してください。

このセクションの内容:

SYSCOLLATION 互換ビュー (廃止予定) [1890 ページ]
SYSCOLLATION 互換ビューにはそのデータベースの照合順情報が入っています。組み込み関数経由で取得でき、カタログには保存されません。このビューの定義を次に示します。

SYSCOLLATIONMAPPINGS 互換ビュー (廃止予定) [1890 ページ]
SYSCOLLATIONMAPPINGS 互換ビューにはデータベース照合マッピングの入ったローが 1 つあるだけです。組み込み関数経由で取得でき、カタログには保存されません。このビューの定義を次に示します。

SYSCOLUMN 互換ビュー (廃止予定) [1891 ページ]
SYSCOLUMN ビューは、SYSCOLUMN システムテーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。

SYSFILE 互換ビュー (廃止予定) [1891 ページ]
SYSFILE システムビューの各ローはデータベースの個々の DB 領域ファイルを記述します。各データベースは、1 つ以上の DB 領域で構成されます。各 DB 領域は、1 つのオペレーティングシステムファイルに対応します。

SYSFKCOL 互換ビュー (廃止予定) [1892 ページ]
SYSFKCOL の各ローは、外部テーブルの外部カラムと、プライマリテーブルのプライマリカラムの個々の関係を記述します。このビューは使用されなくなりました。代わりに SYSIDX と SYSIDXCOL のシステムビューを使用してください。

1.7.3.1 SYSCOLLATION 互換ビュー (廃止予定)

SYSCOLLATION 互換ビューにはそのデータベースの照合順情報が入っています。組み込み関数経由で取得でき、カタログには保存されません。このビューの定義を次に示します。

```
ALTER VIEW "SYS"."SYSCOLLATION"
  as select 1 as collation_id,
    DB_PROPERTY('Collation') as collation_label,
    DB_EXTENDED_PROPERTY('Collation','Description') as collation_name,
    cast(DB_EXTENDED_PROPERTY('Collation','LegacyData') as binary(1280)) as
collation_order
```

関連情報

DB_PROPERTY 関数 [システム] [324 ページ]
DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]

1.7.3.2 SYSCOLLATIONMAPPINGS 互換ビュー (廃止予定)

SYSCOLLATIONMAPPINGS 互換ビューにはデータベース照合マッピングの入ったローが 1 つあるだけです。組み込み関数経由で取得でき、カタログには保存されません。このビューの定義を次に示します。

```
ALTER VIEW "SYS"."SYSCOLLATIONMAPPINGS"
  as select DB_PROPERTY('Collation') as collation_label,
    DB_EXTENDED_PROPERTY('Collation','Description') as collation_name,
    DB_PROPERTY('Charset') as cs_label,
    DB_EXTENDED_PROPERTY('Collation','ASESensitiveSortOrder') as so_case_label,
    DB_EXTENDED_PROPERTY('Collation','ASEInsensitiveSortOrder') as
so_caseless_label,
    DB_EXTENDED_PROPERTY('Charset','java') as jdk_label
```

関連情報

DB_PROPERTY 関数 [システム] [324 ページ]
DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]

1.7.3.3 SYSCOLUMN 互換ビュー (廃止予定)

SYSCOLUMN ビューは、SYSCOLUMN システムテーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。

ただし、以前の SYSCOLUMN テーブルは ISYSTABCOL システムテーブルとそれに対応する SYSTABCOL システムビューによって置き換えられました。代わりに SYSTABCOL システムビューを使用してください。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCOLUMN"
  as select b.table_id,
    b.column_id,
    if c.sequence is null then 'N' else 'Y' endif as pkey,
    b.domain_id,
    b.nulls,
    b.width,
    b.scale,
    b.object_id,
    b.max_identity,
    b.column_name,
    r.remarks,
    b."default",
    b.user_type,
    b.column_type
    from SYS.SYSTABCOL as b
      left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)
      left outer join SYS.ISYSIDXCOL as c on(b.table_id = c.table_id and
b.column_id = c.column_id and c.index_id = 0)
```

関連情報

SYSTABCOL システムビュー [1845 ページ]
SYSREMARK システムビュー [1829 ページ]
SYSIDXCOL システムビュー [1811 ページ]

1.7.3.4 SYSFILE 互換ビュー (廃止予定)

SYSFILE システムビューの各ローはデータベースの個々の DB 領域ファイルを記述します。各データベースは、1 つ以上の DB 領域で構成されます。各 DB 領域は、1 つのオペレーティングシステムファイルに対応します。

メインデータベースファイル、テンポラリファイル、トランザクションログファイル、トランザクションログミラーファイル用に、自動的に DB 領域が作成されます。トランザクションログ、トランザクションログミラーの DB 領域に関する情報は、SYSFILE システムビューには表示されません。

```
ALTER VIEW "SYS"."SYSFILE"
  as select b.dbfile_id as file_id,
    if b.dbspace_id = 0 and b.dbfile_id = 0 then
      db_property('File')
    else
```

```
      if b.dbspace_id = 15 and b.dbfile_id = 15 then
        db_property('TempFileName')
      else
        b.file_name
      endif
    endif as file_name,
    a.dbspace_name,
    a.store_type,
    b.lob_map,
    b.dbspace_id
    from SYS.ISYSDBSPACE as a
      join SYS.ISYSDBFILE as b on(a.dbspace_id = b.dbspace_id)
```

1.7.3.5 SYSFKCOL 互換ビュー (廃止予定)

SYSFKCOL の各ローは、外部テーブルの外部カラムと、プライマリテーブルのプライマリカラムの個々の関係を記述します。このビューは使用されなくなりました。代わりに SYSIDX と SYSIDXCOL のシステムビューを使用してください。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSFKCOL"
  as select a.table_id as foreign_table_id,
    a.index_id as foreign_key_id,
    a.column_id as foreign_column_id,
    a.primary_column_id
    from SYS.ISYSIDXCOL as a
      ,SYS.ISYSIDX as b
    where a.table_id = b.table_id
    and a.index_id = b.index_id
    and b.index_category = 2
```

関連情報

SYSIDX システムビュー [1810 ページ]
SYSIDXCOL システムビュー [1811 ページ]

1.7.3.6 SYSFOREIGNKEY 互換ビュー (廃止予定)

SYSFOREIGNKEY ビューは、SYSFOREIGNKEY システムテーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前の SYSFOREIGNKEY システムテーブルは ISYSFKEY システムテーブルとそれに対応する SYSFKEY システムビューによって置き換えられたため、そちらを使用するようにしてください。

外部キーは、外部テーブルとプライマリテーブルの 2 つのテーブル間の関係です。外部キーはすべて、SYSFOREIGNKEY の 1 つのローと SYSFKCOL の 1 つまたは複数のローで定義されます。SYSFOREIGNKEY には、外部キーに関する一般的な情報が入っています。SYSFKCOL は外部キーのカラムを識別して、外部キーの各カラムとプライマリテーブルの中のプライマリキーのカラムを関連付けます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSFOREIGNKEY"
  as select b.foreign_table_id,
    b.foreign_index_id as foreign_key_id,
    a.object_id,
    b.primary_table_id,
    p.root,
    b.check_on_commit,
    b.nulls,
    a.index_name as role,
    r.remarks,
    b.primary_index_id,
    a.not_enforced as fk_not_enforced,
    10 as hash_limit
    from(SYS.ISYSIDX as a left outer join SYS.ISYSPHYSIDX as p on(a.table_id =
p.table_id and a.phys_index_id = p.phys_index_id))
      left outer join SYS.ISYSREMARK as r on(a.object_id = r.object_id)
      ,SYS.ISYSFKEY as b
    where a.table_id = b.foreign_table_id
    and a.index_id = b.foreign_index_id
```

関連情報

SYSIDX システムビュー [1810 ページ]
SYSPHYSIDX システムビュー [1823 ページ]
SYSREMARK システムビュー [1829 ページ]
SYSFKEY システムビュー [1806 ページ]

1.7.3.7 SYSGROUP 互換ビュー

SYSGROUP システムビューには各グループの各メンバに対して 1 つのローがあります。このビューは、グループとメンバーの多対多の関係を示します。グループには複数のメンバーがあり、ユーザは複数のグループのメンバーになることができます。

カラム名	データ型	説明
group_id	UNSIGNED INT	グループのユーザ番号。
group_member	UNSIGNED INT	メンバーのユーザ番号。

1.7.3.8 SYSGROUPS 互換ビュー

SYSGROUPS ビューには各グループの各メンバに対して 1 つのローがあります。このビューは、グループとメンバーの多対多の関係を示します。グループには複数のメンバーがあり、ユーザは複数のグループのメンバーになることができます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSGROUPS"( group_name,
  member_name )
  as select g.user_name,u.user_name
    from SYS.ISYSROLEGRANT,SYS.ISYSUSER as g,SYS.ISYSUSER as u
    where ISYSROLEGRANT.role_id = g.user_id and ISYSROLEGRANT.grantee = u.user_id
and(
    u.user_name in( 'SYS_SPATIAL_ADMIN_ROLE' )
    or u.user_id <= 2147483648) and(
    g.user_type = (0x02|0x04|0x08)
    or g.user_name in( 'SYS','PUBLIC','dbo','diagnostics',
    'rs_systabgroup','SA_DEBUG','SYS_SPATIAL_ADMIN_ROLE' ) )
```

関連情報

SYSUSER システムビュー [1855 ページ]
SYSGROUP 互換ビュー [1893 ページ]

1.7.3.9 SYSINDEX 互換ビュー (廃止予定)

SYSINDEX ビューは、SYSINDEX システムテーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前の SYSINDEX システムテーブルは ISYSIDX システムテーブルとそれに対応する SYSIDX システムビューによって置き換えられたため、そちらを使用するようにしてください。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSINDEX"
  as select b.table_id,
    b.index_id,
    b.object_id,
    p.root,
    b.dbspace_id,
    case b."unique"
    when 1 then 'Y'
    when 2 then 'U'
    when 3 then 'M'
    when 4 then 'N'
    when 5 then 'Y'
    else 'I'
    end as "unique",
    t.creator,
    b.index_name,
    r.remarks,
    10 as hash_limit,
```

```
    b.dbspace_id as file_id
    from(SYS.ISYSIDX as b left outer join SYS.ISYSPHYSIDX as p on(b.table_id =
p.table_id and b.phys_index_id = p.phys_index_id))
      left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)
      ,SYS.ISYSTAB as t
    where t.table_id = b.table_id
    and b.index_category = 3
```

関連情報

SYSIDX システムビュー [1810 ページ]
SYSPHYSIDX システムビュー [1823 ページ]
SYSTABLE 互換ビュー (廃止予定) [1896 ページ]
SYSREMARK システムビュー [1829 ページ]

1.7.3.10 SYSINFO 互換ビュー (廃止予定)

SYSINFO ビューには、データベースが作成されたときに定義されたデータベース特性が表示されます。これに含まれるローの数は常に 1 つのみです。このビューは組み込み関数経由で取得でき、カタログには保存されません。次に SYSINFO ビューの定義を示します。

```
ALTER VIEW "SYS"."SYSINFO"( page_size,
  encryption,
  blank_padding,
  case_sensitivity,
  default_collation,
  database_version )
  as select db_property('PageSize'),
    if db_property('Encryption') <> 'None' then 'Y' else 'N' endif,
    if db_property('BlankPadding') = 'On' then 'Y' else 'N' endif,
    if db_property('CaseSensitive') = 'On' then 'Y' else 'N' endif,
    db_property('Collation'),
    NULL
```

関連情報

DB_PROPERTY 関数 [システム] [324 ページ]
DB_EXTENDED_PROPERTY 関数 [システム] [317 ページ]

1.7.3.11 SYSIXCOL 互換ビュー (廃止予定)

SYSIXCOL の各ローは、インデックスのカラムを示し、SYSIXCOL システムテーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。

SYSIXCOL システムテーブルの代わりに ISYSIDXCOL システムテーブルとそれに対応する SYSIDXCOL システムビューが使用されます。SYSIDXCOL システムビューの使用をお奨めします。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSIXCOL"
  as select a.table_id,
    a.index_id,
    a.sequence,
    a.column_id,
    a."order"
    from SYS.ISYSIDXCOL as a
      ,SYS.ISYSIDX as b
    where a.table_id = b.table_id
    and a.index_id = b.index_id
    and b.index_category = 3
```

関連情報

SYSIDX システムビュー [1810 ページ]
SYSIDXCOL システムビュー [1811 ページ]

1.7.3.12 SYSTABLE 互換ビュー (廃止予定)

SYSTABLE ビューは、SYSTABLE システムテーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前の SYSTABLE テーブルは、ISYSTAB システムテーブルおよびこのテーブルに対応する SYSTAB システムビューで置換されたため、SYSTAB の使用をお奨めします。

SYSTABLE ビューの各ローは、データベース内のテーブル 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSTABLE"
  as select b.table_id,
    b.file_id,
    b.count,
    0 as first_page,
    b.commit_action as last_page,
    COALESCE(ph.root,0) as primary_root,
    b.creator,
    0 as first_ext_page,
    0 as last_ext_page,
    b.table_page_count,
    b.ext_page_count,
```

```
    b.object_id,
    b.table_name,
    b.table_type_str as table_type,
    v.view_def,
    r.remarks,
    b.replicate,
    p.existing_obj,
    p.remote_location,
    'T' as remote_objtype,
    p.srvid,
    case b.server_type
    when 1 then 'SA'
    when 2 then 'IQ'
    when 3 then 'OMNI'
    else 'INVALID'
    end as server_type,
    10 as primary_hash_limit,
    0 as page_map_start,
    s.source,
    b."encrypted"
    from SYS.SYSTAB as b
      left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)
      left outer join SYS.ISYSSOURCE as s on(b.object_id = s.object_id)
      left outer join SYS.ISYSVIEW as v on(b.object_id = v.view_object_id)
      left outer join SYS.ISYSPROXYTAB as p on(b.object_id = p.table_object_id)
      left outer join(SYS.ISYSIDX as i left outer join SYS.ISYSPHYSIDX as ph
on(i.table_id = ph.table_id
        and i.phys_index_id = ph.phys_index_id)) on(b.table_id = i.table_id and
i.index_category = 1
        and i.index_id = 0)
```

関連情報

SYSTAB システムビュー [1842 ページ]
SYSREMARK システムビュー [1829 ページ]
SYSSOURCE システムビュー [1836 ページ]
SYSVIEW システムビュー [1860 ページ]
SYSPROXYTAB システムビュー [1827 ページ]
SYSIDX システムビュー [1810 ページ]
SYSPHYSIDX システムビュー [1823 ページ]

1.7.3.13 SYSUSERAUTH 互換ビュー (廃止予定)

SYSUSERAUTH ビューの各ローは、ユーザを記述しますが、ユーザ ID とパスワードハッシュは公開されません。代わりに、ユーザはユーザ名によって識別されます。

このビューにアクセスするには、SELECT ANY TABLE システム権限が必要です。

SYSUSERAUTH ビューは、旧バージョンのソフトウェアとの互換性を保つために用意されています。代わりに SYSROLEGRANTS 統合ビューを使用してください。

パスワードカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。

このビューのタイトルには auth (権限) という単語が含まれていますが、セキュリティモデルはロールと権限に基づきます。したがって、ビュー内のデータは、ビューの定義に記載されているテーブルとビューからのロール情報を使用してコンパイルされます。

```
ALTER VIEW "SYS"."SYSUSERAUTH"( "name",
  "password","resourceauth","dbaauth","scheduleauth","user_group" )
  as select
"SYSUSERPERM"."user_name","SYSUSERPERM"."password","SYSUSERPERM"."resourceauth","SYS
USERPERM"."dbaauth","SYSUSERPERM"."scheduleauth","SYSUSERPERM"."user_group"
    from "SYS"."SYSUSERPERM"
```

関連情報

SYSROLEGRANTS 統合ビュー [1880 ページ]
SYSROLEGRANT システムビュー [1831 ページ]
SYSROLEGRANTEXT システムビュー [1832 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.3.14 SYSUSERAUTHORITY 互換ビュー (廃止予定)

SYSUSERAUTHORITY ビューは、古いバージョンのソフトウェアとの互換性を保つために用意されています。代わりに SYSROLEGRANTS 統合ビューを使用してください。

SYSUSERAUTHORITY システムビューの各ローは、個々のユーザ ID に付与された権限を示します。

このビューのタイトルには authority (権限) という単語が含まれていますが、セキュリティモデルはロールと権限に基づきます。したがって、ビュー内のデータは、ビューの定義に記載されているテーブルとビューからのロール情報を使用してコンパイルされます。

```
ALTER VIEW "SYS"."SYSUSERAUTHORITY" as
  select ISYSROLEGRANT.grantee as user_id,
    sp_auth_sys_role_info.auth
    from SYS.ISYSROLEGRANT
      natural join dbo.sp_auth_sys_role_info()
    where ISYSROLEGRANT.grant_type <> (0x02|0x04) and
    not ISYSROLEGRANT.grantee = any(select sp_auth_sys_role_info.role_id from
dbo.sp_auth_sys_role_info()) union
  select ISYSUSER.user_id,
    cast('GROUP' as varchar(20)) as auth
    from SYS.ISYSUSER
    where ISYSUSER.user_name
in( 'SYS','PUBLIC','diagnostics','SYS_SPATIAL_ADMIN_ROLE','rs_systabgroup','SA_DEBUG
','dbo' ) union
  select ISYSUSER.user_id,
    cast('GROUP' as varchar(20)) as auth
    from SYS.ISYSUSER
    where ISYSUSER.user_type = (0x02|0x04|0x08) union
  select cast(opt.setting as unsigned integer) as user_id,
    cast('PUBLISH' as varchar(20)) as auth
    from SYS.ISYSOPTION as opt
    where opt."option" like '%db_publisher%' and opt.setting not like '%-1%'
```

関連情報

SYSROLEGRANTS 統合ビュー [1880 ページ]
SYSROLEGRANT システムビュー [1831 ページ]
SYSROLEGRANTEXT システムビュー [1832 ページ]

1.7.3.15 SYSUSERLIST 互換ビュー (廃止予定)

SYSUSERAUTH ビューは、旧バージョンのソフトウェアとの互換性を保つために用意されています。

SYSUSERLIST ビューの各ローは、ユーザを記述しますが、ユーザ ID とパスワードは公開されません。各ユーザは、ユーザ名で識別されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSERLIST"( name,
  resourceauth,dbaauth,scheduleauth,user_group )
  as select
SYSUSERPERM.user_name,SYSUSERPERM.resourceauth,SYSUSERPERM.dbaauth,SYSUSERPERM.sched
uleauth,SYSUSERPERM.user_group
    from SYS.SYSUSERPERM
```

関連情報

SYSUSERPERM 互換ビュー (廃止予定) [1899 ページ]

1.7.3.16 SYSUSERPERM 互換ビュー (廃止予定)

SYSUSERPERM ビューの各ローは個々のユーザ ID を記述します。

前のバージョンで使用できる権限とパーミッションが表示されるだけなので、このビューは使用されなくなりました。SYSROLEGRANTS 統合ビューを使用するようにアプリケーションを変更してください。

このビューにアクセスするには、SELECT ANY TABLE システム権限が必要です。

パスワードカラムには、データベースに値がある場合は 3 つのアスタリスク (***) が表示され、値がない場合は NULL が表示されます。実際のパスワード情報を確認するには、SYSUSERPASSWORD システムビューを確認してください。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。

```
ALTER VIEW "SYS"."SYSUSERPERM"
  as select "b"."user_id",
    "b"."object_id",
    "b"."user_name",
    if "b"."password" is null then null else '***' endif as "password",
```

```
    if "AA"."resourceauth" is not null and "AA"."resourceauth" > 0 then
      'Y' else 'N' endif as "resourceauth",
    if "AA"."dbaauth" is not null and "AA"."dbaauth" > 0 then
      'Y' else 'N' endif as "dbaauth",
    'N' as "scheduleauth",
    if exists(select * from "SYS"."ISYSOPTION" as "opt"
      where "opt"."option" like '%db_publisher%' and "opt"."setting" not like '%-1'
      and "b"."user_id" = cast("opt"."setting" as integer)) then
      'Y' else 'N' endif as "publishauth",
    if "AA"."remotedbaauth" is not null and "AA"."remotedbaauth" > 0 then
      'Y' else 'N' endif as "remotedbaauth",
    if "b"."user_type" = (0x02|0x04|0x08) or "b"."user_name"
in( 'SYS','PUBLIC','diagnostics','SYS_SPATIAL_ADMIN_ROLE','rs_systabgroup','SA_DEBUG
','dbo' ) then
      'Y' else 'N' endif as "user_group",
    "r"."remarks"
    from "SYS"."ISYSUSER" as "b"
      left outer join "SYS"."ISYSREMARK" as "r" on("b"."object_id" =
"r"."object_id")
      left outer join(select "sum"(if "sp_auth_sys_role_info"."auth" = 'RESOURCE'
then 1 else 0 endif) as "resourceauth",
        "sum"(if "sp_auth_sys_role_info"."auth" = 'DBA' then 1 else 0 endif) as
"dbaauth",
        "sum"(if "sp_auth_sys_role_info"."auth" = 'REMOTE DBA' then 1 else 0 endif)
as "remotedbaauth",
        "ISYSROLEGRANT"."grantee"
        from "SYS"."ISYSROLEGRANT" natural join "dbo"."sp_auth_sys_role_info"()
        where "ISYSROLEGRANT"."grant_type" <> (0x02|0x04)
        and "sp_auth_sys_role_info"."auth" in( 'DBA','RESOURCE','REMOTE DBA' )
        group by "ISYSROLEGRANT"."grantee") as "AA"
      on("AA"."grantee" = "b"."user_id")
```

関連情報

SYSROLEGRANTS 統合ビュー [1880 ページ]
SYSROLEGRANT システムビュー [1831 ページ]
SYSROLEGRANTEXT システムビュー [1832 ページ]
SYSUSER システムビュー [1855 ページ]
SYSUSERPASSWORD システムビュー [1857 ページ]

1.7.3.17 SYSUSERPERMS 互換ビュー (廃止予定)

SYSUSERPERMS ビューの各ローは個々のユーザ ID を記述します。ただし、パスワード情報は含まれません。すべてのユーザがこのビューを表示できます。

前のバージョンで使用できる権限とパーミッションが表示されるだけなので、このビューは使用されなくなりました。SYSROLEGRANTS 統合ビューを使用するようにアプリケーションを変更してください。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSERPERMS"
```

```
  as select
SYSUSERPERM.user_id,SYSUSERPERM.user_name,SYSUSERPERM.resourceauth,SYSUSERPERM.dbaau
th,

SYSUSERPERM.scheduleauth,SYSUSERPERM.user_group,SYSUSERPERM.publishauth,SYSUSERPERM.
remotedbaauth,SYSUSERPERM.remarks
    from SYS.SYSUSERPERM
```

関連情報

SYSROLEGRANTS 統合ビュー [1880 ページ]
SYSROLEGRANT システムビュー [1831 ページ]
SYSROLEGRANTEXT システムビュー [1832 ページ]
SYSUSER システムビュー [1855 ページ]

1.7.4 Transact-SQL 互換のビュー

Adaptive Server Enterprise と SQL Anywhere のシステムカタログは異なります。

Adaptive Server Enterprise システムテーブルとビューは、ユーザ dbo が所有しています。一部は master データベースにあり、一部は sybsecurity データベースにあり、一部は個々のデータベースにあります。SQL Anywhere システムのテーブルとビューは、SYS という特殊なユーザが所有し、各データベースに別々に存在しています。

互換アプリケーションを使用しやすいように、SQL Anywhere は特殊なユーザ dbo が所有する次のビューセットを用意しています。これは Adaptive Server Enterprise のビューに対応しています。構造上の違いのために、特定の Adaptive Server Enterprise テーブルまたはビューの内容が SQL Anywhere のコンテキストで無意味になる場合、そのビューはカラム名とデータ型だけの空のビューになります。

ビュー名	説明
syscolumns	テーブルまたはビューのカラムごとに、またプロシージャのパラメータごとに 1 ロー。
syscomments	ビュー、ルール、デフォルト、トリガ、プロシージャごとに 1 つ以上のローを追加。SQL 定義文を表示。
sysindexes	クラスタドインデックスまたはノンクラスタドインデックスごとに 1 ロー、インデックスのないテーブルごとに 1 ロー、および text または image データを持つテーブルごとに 1 ロー追加。
sysobjects	テーブル、ビュー、プロシージャ、ルール、トリガ、デフォルト、ログ、またはテンポラリオブジェクト (tempdb 内でのみ) ごとに 1 ロー。
systypes	システム提供またはユーザ定義のデータ型ごとに 1 ロー追加。
sysusers	データベースについての許可を持つユーザごとに 1 ロー追加。
syslogins	有効なユーザアカウントごとに 1 ロー。

1.8 このマニュアルの印刷、再生、および再配布

次の条件に従うかぎり、このマニュアルの全部または一部を使用、印刷、再生、配布することができます。

1. ここに示したものとそれ以外のすべての版権と商標の表示をすべてのコピーに含めること。
2. マニュアルに変更を加えないこと。
3. SAP 以外の人間がマニュアルの著者または情報源であるかのように示す一切の行為をしないこと。

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼動システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証は行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (http://help.sap.com/disclaimer を参照)。

go.sap.com/registration/contact.html