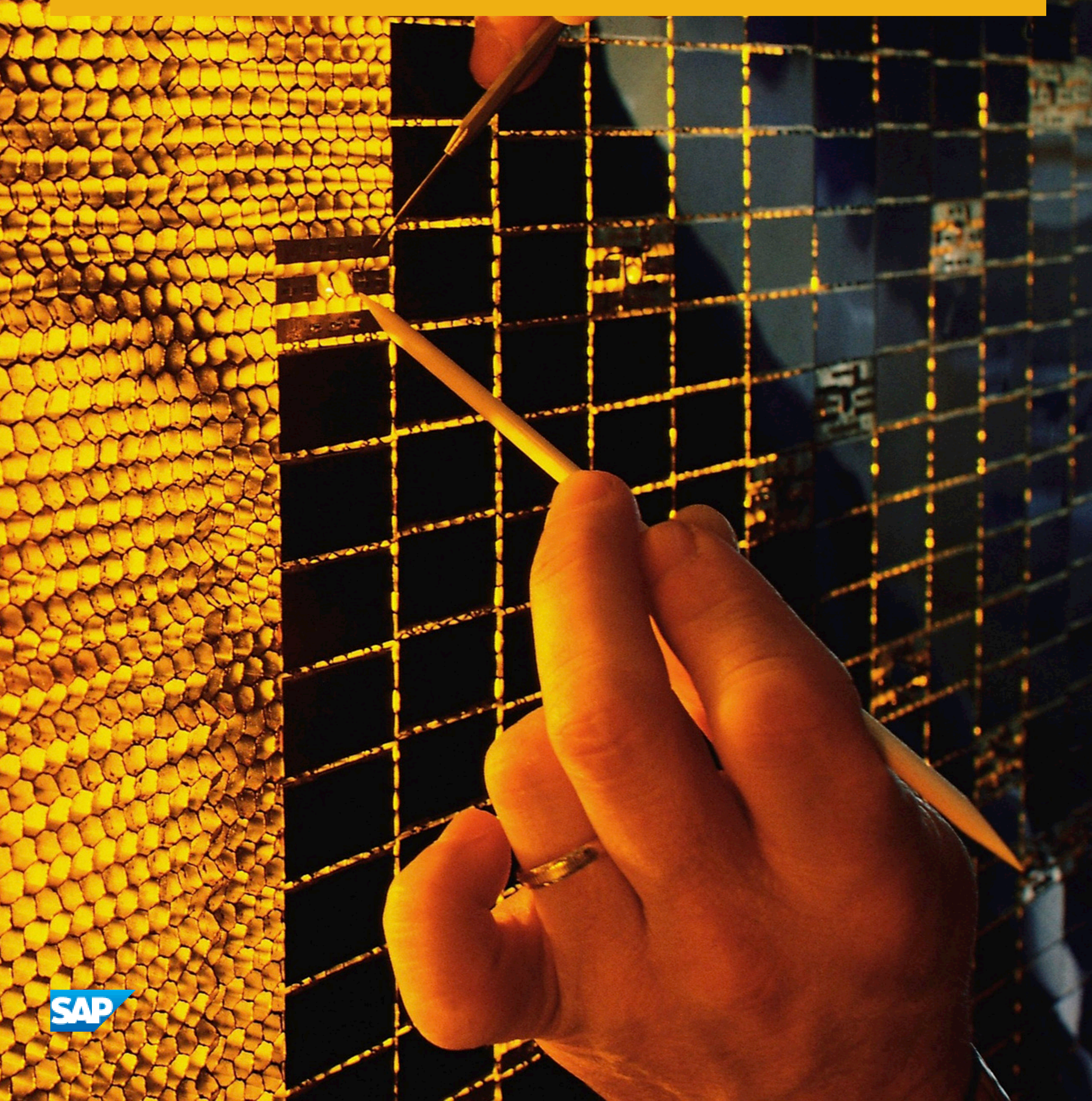
SQL Anywhere サーバ
文書バージョン: 17 – 2016-05-11

SQL Anywhere - SQL の使用法



目次

1 SQL Anywhere サーバ - SQL の使用法. 6

1.1 テーブル、ビュー、インデックス. 7

データベースオブジェクトの名前とプレフィクス. 8

システムオブジェクトのリストの表示 (SQL Central). 9

システムオブジェクトのリストの表示 (SQL の場合). 10

テーブル. 11

テンポラリテーブル. 18

計算カラム. 21

プライマリキー. 25

外部キー. 29

インデックス. 35

ビュー. 49

マテリアライズドビュー. 64

1.2 ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 87

プロシージャ、トリガ、およびユーザ定義関数の利点. 89

プロシージャ. 89

ユーザ定義関数. 101

トリガ. 107

バッチ. 119

プロシージャ、トリガ、ユーザ定義関数の構造. 122

制御文. 125

結果セット. 128

プロシージャ、トリガ、ユーザ定義関数、バッチのカーソル. 137

エラーと警告の処理. 140

プロシージャ、トリガ、ユーザ定義関数、バッチで使用される EXECUTE IMMEDIATE. 152

プロシージャ、トリガ、ユーザ定義関数でのトランザクションとセーブポイント 154

プロシージャ、トリガ、ユーザ定義関数、バッチを作成するときのヒント. 155

プロシージャ、トリガ、イベント、バッチで使用できる文. 156

プロシージャ、ファンクション、トリガ、イベント、またはビューの内容を隠す. 157

1.3 クエリとデータ修正. 159

クエリ. 160

全文検索. 248

チュートリアル: テーブルデータの行列変換. 352

クエリ結果の要約、グループ化、ソート. 355

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ジョイン: 複数テーブルからのデータ検索. 376
共通テーブル式. 420
OLAP のサポート. 435
サブクエリの使用. 474
データ操作文. 498
1.4 SQL のダイアレクトと互換性. 515
SQL Flagger を使用した SQL 準拠のテスト. 516
他の SQL ソフトウェアとは異なる機能. 518
Watcom SQL. 522
Transact-SQL との互換性. 523
SQL Anywhere と Adaptive Server Enterprise の比較. 525
Transact-SQL と互換性のあるデータベース. 530
Transact-SQL 互換性のある SQL 文. 538
Transact-SQL のプロシージャ言語. 544
ストアドプロシージャの自動変換. 546
Transact-SQL プロシージャから返される結果セット. 547
Transact-SQL プロシージャの中の変数. 548
Transact-SQL プロシージャでのエラー処理. 549
1.5 データベースにおける XML. 551
リレーショナルデータベースにおける XML 文書の格納. 552
XML としてエクスポートされるリレーショナルデータ. 553
XML 文書をリレーショナルデータとしてインポートする方法. 554
XML としてのクエリ結果. 561
結果を表示するための Interactive SQL の使用. 579
クエリ結果を XML として取得するための SQL/XML の使用. 580
1.6 データベース内の JSON. 588
クエリ結果を JSON として取り出すための FOR JSON 句の使用. 589
FOR JSON RAW. 589
FOR JSON AUTO. 590
FOR JSON EXPLICIT. 591
1.7 データのインポートとエクスポート. 594
バルクオペレーションのパフォーマンス面. 595
バルクオペレーションのデータリカバリの問題. 596
データインポート. 596
データエクスポート. 614
クライアントコンピュータ上のデータへのアクセス. 631
データベースの再構築. 633
データベース抽出. 650
SQL Anywhere へのデータベース移行. 650

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL スクリプトファイル. 656
Adaptive Server Enterprise の互換性. 661
1.8 リモートデータアクセス. 661
リモートサーバとリモートテーブルのマッピング. 663
ディレクトリアクセスサーバの代用としてのストアドプロシージャ. 672
ディレクトリアクセスサーバ. 672
外部ログイン. 681
プロキシテーブル. 684
ネイティブ文とリモートサーバ. 693
リモートプロシージャコール (RPC). 693
トランザクションの管理とリモートデータ. 698
クエリで実行される内部オペレーション. 699
その他の内部オペレーション. 700
トラブルシューティング: リモートデータには使用できない機能. 702
トラブルシューティング: 大文字と小文字の区別およびリモートデータアクセス. 703
トラブルシューティング: リモートデータアクセスの接続テスト. 703
トラブルシューティング: クエリ上でブロックされるクエリ. 703
トラブルシューティング:ODBC を使用したリモートデータアクセスの接続. 704
リモートデータアクセスのサーバクラス. 704
1.9 データ整合性. 727
データが有効でなくなる状況. 728
整合性制約. 728
データの整合性を維持するためのツール. 729
整合性制約を実装するための SQL 文. 730
カラムデフォルト. 730
テーブルとカラム制約. 738
ドメインを使用したデータの整合性の向上方法. 743
エンティティ整合性と参照整合性. 746
システムテーブルの整合性ルール. 756
1.10 トランザクションと独立性レベル. 757
トランザクション. 758
同時実行性. 761
トランザクション内のセーブポイント. 762
独立性レベルと一貫性. 763
トランザクションのブロックとデッドロック. 779
ロックの仕組み. 782
独立性レベル選択のガイドライン. 802
独立性レベルに関するチュートリアル. 807
ユニークな値を生成するためのシーケンスの使用. 830

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.11 SQL Anywhere のデバッガ 834
デバッガの稼働条件 835
チュートリアル: デバッガの使用開始 835
ブレークポイント 841
デバッガによる変数の変更 844
接続とブレークポイント 847
1.12 このマニュアルの印刷、再生、および再配布 847

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1 SQL Anywhere サーバ - SQL の使用法

このマニュアルでは、データベースへのオブジェクトの追加方法、データのインポート、エクスポートおよび変更方法、データの検索方法、ストアドプロシージャとトリガの構築方法について説明します。

このセクションの内容:

テーブル、ビュー、インデックス [7 ページ]
テーブル、ビュー、およびインデックスはデータベース内にデータを保持しています。

ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 [87 ページ]
プロシージャとトリガは、手続き型 SQL 文をデータベースに格納します。

クエリとデータ修正 [159 ページ]
データベースのデータの問い合わせや修正を行うために、多くの機能が用意されています。

SQL のダイアレクトと互換性 [515 ページ]
準拠に関する情報については、ソフトウェアの各機能のリファレンスマニュアルを参照してください。

データベースにおける XML [551 ページ]
Extensible Markup Language (XML) は、構造化データをテキスト形式で表します。XML は、大規模な電子出版の課題を満たすために設計されました。

データベース内の JSON [588 ページ]
JavaScript Object Notation (JSON) は、言語に依存しないテキストベースのデータ交換フォーマットで、JavaScript データの直列化のために開発されました。

データのインポートとエクスポート [594 ページ]
バルクオペレーションという用語は、データのインポートやエクスポートのプロセスを説明するために使用されます。

リモートデータアクセス [661 ページ]
リモートデータアクセスによって、他のデータソースのデータや、データベースサーバを実行しているコンピュータにあるファイルにアクセスすることができます。

データ整合性 [727 ページ]
データに整合性があるということは、データが有効、つまり適切であり正確で、データベースの関係構造が保たれていることを意味します。

トランザクションと独立性レベル [757 ページ]
トランザクションと独立性レベルは、一貫性を通じてデータ整合性の確保に役立ちます。

SQL Anywhere のデバッガ [834 ページ]
SQL Anywhere のデバッガは、作成した SQL のストアドプロシージャ、トリガ、イベントハンドラ、ユーザ定義関数をデバッグするために使用できます。

このマニュアルの印刷、再生、および再配布 [847 ページ]
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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1 テーブル、ビュー、インデックス

テーブル、ビュー、およびインデックスはデータベース内にデータを保持しています。

データベーステーブル、ビュー、インデックスを作成、変更、削除する SQL 文は、データ定義言語 (DDL) と呼ばれます。データベースオブジェクトの定義は、データベーススキーマを形成します。スキーマは、データベースの論理的なフレームワークです。

このセクションの内容:

データベースオブジェクトの名前とプレフィクス [8 ページ]
すべてのデータベースオブジェクトの名前は、プレフィクスを含めて、識別子です。

システムオブジェクトのリストの表示 (SQL Central) [9 ページ]
SQL Central を使用して、システムテーブル、システムビュー、格納されたプロシージャ、ドメインなどシステムオブジェクトに関する情報を表示します。

システムオブジェクトのリストの表示 (SQL の場合) [10 ページ]
SYSOBJECT システムビューのクエリを行って、システムテーブル、システムビュー、格納されたプロシージャ、ドメインなどシステムオブジェクトに関する情報を表示します。

テーブル [11 ページ]
データベースを初めて作成した場合、そのデータベースにあるテーブルはシステムテーブルだけです。システムテーブルにはデータベースのスキーマが保管されます。

テンポラリテーブル [18 ページ]
テンポラリテーブルはテンポラリファイルに格納されます。

計算カラム [21 ページ]
計算カラムとは、同一ロー内にある従属カラムと呼ばれる他のカラムの値を参照する式のことです。

プライマリキー [25 ページ]
リレーショナルデータベース内の各テーブルには、プライマリキーが必要です。プライマリキーとは 1 つのカラム、またはカラムのセットで、各ローをユニークに識別します。

外部キー [29 ページ]
外部キーは、カラムまたはカラムのセットで構成されており、キー値が一致するプライマリテーブル内のローの参照を表します。

インデックス [35 ページ]
インデックスは、テーブルの 1 カラムまたは複数のカラムについてローに順序を付けます。

ビュー [49 ページ]
ビューとは、ビュー定義の結果セットによって定義される計算テーブルです。ビュー定義は SQL クエリとして表現されます。

マテリアライズドビュー [64 ページ]
マテリアライズドビューとは、ベーステーブルとよく似ていて、参照先のベーステーブルから結果セットが事前に計算されてディスクに格納されるビューです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 [87 ページ]
データ整合性 [727 ページ]

1.1.1 データベースオブジェクトの名前とプレフィクス

すべてのデータベースオブジェクトの名前は、プレフィクスを含めて、識別子です。

このマニュアルの例では、サンプルデータベースのデータベースオブジェクトは、通常その識別子のみを使用して参照されます。次に例を示します。

```
SELECT * FROM Employees;
```

テーブル、プロシージャ、ビューにはすべて所有者が存在します。GROUPO ユーザは、サンプルデータベースのサンプルテーブルを所有しています。場合によっては、オブジェクト名に所有者のユーザ ID をプレフィクスとして付ける必要があります。次に例を示します。

```
SELECT * FROM GROUPO.Employees;
```

Employees テーブルの参照は修飾されています。単にオブジェクト名を示すだけでよい場合もあります。

データベースオブジェクトを参照するときプレフィクスが必要ないのは、次の場合です。

- 自分がデータベースオブジェクトの所有者である場合。
- 自分が付与したロールの誰かがデータベースオブジェクトを所有している場合。

例

Acme 社のコーポレートデータベースの次の例を検討します。ユーザ ID Asmin は、データベースに対するすべての管理権限とともに作成されます。他の 2 つのユーザ ID である Joe と Sally は、販売部で働く従業員用に作成されます。

```
CREATE USER Admin IDENTIFIED BY secret;
GRANT ROLE SYS_AUTH_SSO_ROLE TO Admin;
GRANT ROLE SYS_AUTH_SA_ROLE TO Admin;
CREATE USER Sally IDENTIFIED BY xxxxx;
CREATE USER Joe IDENTIFIED BY xxxxx;
```

Admin ユーザは、データベースにテーブルを作成して、Acme ロールに所有者を割り当てます。

```
CREATE ROLE Acme;
CREATE TABLE Acme.Customers ( ... );
CREATE TABLE Acme.Products ( ... );
CREATE TABLE Acme.Orders ( ... );
CREATE TABLE Acme.Invoices ( ... );
CREATE TABLE Acme.Employees ( ... );
CREATE TABLE Acme.Salaries ( ... );
```

会社の全員がすべての情報にアクセスできるようにはしません。販売部で働く Joe と Sally には、Customers、Products、Orders テーブルが必要ですが、その他のテーブルは不要です。これを実行するには、SalesForce ロールを

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

作成して、制限されたテーブルのセットへのアクセスに必要な権限をこのロールに割り当て、この 2 人の従業員にこのロールを割り当てます。

```
CREATE ROLE SalesForce;
GRANT ALL ON Acme.Customers TO SalesForce;
GRANT ALL ON Acme.Orders TO SalesForce;
GRANT SELECT ON Acme.Products TO SalesForce;
GRANT ROLE SalesForce TO Sally;
GRANT ROLE SalesForce TO Joe;
```

Joe と Sally はこれらのテーブルを使用するのに必要な権限を持っていますが、テーブルの所有者が Acme であるため、テーブルを参照する際には依然として修飾が必要です。

```
SELECT * FROM Acme.Customers;
```

この状況を変更するには、Acme ロールを Sales ロールに付与します。

```
GRANT ROLE Acme TO SalesForce;
```

Sales ロールが付与されている Joe と Sally に Acme ロールが間接的に付与され、識別子なしでテーブルを参照できるようになりました。SELECT 文は、次のように簡略化できます。

```
SELECT * FROM Customers;
```

i 注記

Acme ユーザ定義ロールによって、オブジェクトレベルの権限が与えられるわけではありません。単に、ロールが所有しているオブジェクトを所有者の修飾なしで参照できるようになるだけです。Joe と Sally は Acme ロールを割り当てられたからといって、それ以外の権限も持つわけではありません。Acme ロールには、特別な権限は明示的に付与されていません。Admin ユーザは、テーブルを作成して適切な権限を持っているため、Salaries のようなテーブルを表示する暗黙的な権限を持っています。したがって、Joe と Sally が次のいずれかの文を実行するとエラーになります。

```
SELECT * FROM Acme.Salaries;
SELECT * FROM Salaries;
```

どちらの場合も、Joe と Sally は Salaries テーブルを参照するのに必要な権限を持っていません。

1.1.2 システムオブジェクトのリストの表示 (SQL Central)

SQL Central を使用して、システムテーブル、システムビュー、格納されたプロシージャ、ドメインなどシステムオブジェクトに関する情報を表示します。

コンテキスト

この作業を行うのは、データベース内のシステムオブジェクトのリストとその定義を見るとき、または定義を使用して似たような他のオブジェクトを作成するときです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. SQL Central で、*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. データベースを選択し、 ファイル > 所有者フィルタの設定 をクリックします。
3. *SYS* および *dbo* を選択します。
4. *OK* をクリックします。

結果

SQL Central にシステムオブジェクトのリストが表示されます。

1.1.3 システムオブジェクトのリストの表示 (SQL の場合)

SYSOBJECT システムビューのクエリを行って、システムテーブル、システムビュー、格納されたプロシージャ、ドメインなどシステムオブジェクトに関する情報を表示します。

コンテキスト

この作業を行うのは、データベース内のシステムオブジェクトのリストとその定義を見るとき、または定義を使用して似たような他のオブジェクトを作成するときです。

手順

1. Interactive SQL では、データベースに接続します。
2. SELECT 文を実行し、SYSOBJECT システムビューに対してオブジェクトのリストを問い合わせます。

結果

Interactive SQL のシステムオブジェクトのリストが表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

次の SELECT 文は、SYSOBJECT システムビューに対してクエリを行い、SYS と dbo が所有するすべてのテーブルとビューのリストを返します。SYSTAB システムビューに対してジョインを行ってオブジェクト名を返し、SYSUSER システムビューに対してジョインを行って所有者名を返します。

```
SELECT b.table_name "Object Name",
      c.user_name "Owner",
      b.object_id "ID",
      a.object_type "Type",
      a.status "Status"
  FROM ( SYSOBJECT a JOIN SYSTAB b
         ON a.object_id = b.object_id )
  JOIN SYSUSER c
WHERE c.user_name = 'SYS'
   OR c.user_name = 'dbo'
ORDER BY c.user_name, b.table_name;
```

1.1.4 テーブル

データベースを初めて作成した場合、そのデータベースにあるテーブルはシステムテーブルだけです。システムテーブルにはデータベースのスキーマが保管されます。

必要に応じてデータベーススキーマを簡単に再作成するには、SQL スクリプトファイルを作成してデータベースのテーブルを定義します。SQL スクリプトファイルには、CREATETABLE 文と ALTERTABLE 文を含めてください。

このセクションの内容:

テーブルの作成 [12 ページ]
SQL Central を使用して、データベースにテーブルを作成します。

テーブル変更 [13 ページ]
カラムの追加、さまざまなカラム属性の変更、またはカラムの削除により、テーブルの構造またはカラム定義を変更します。

テーブルでのデータの表示 (SQL Central の場合) [17 ページ]
SQL Central を使用して、テーブルでデータを閲覧します。

テーブルでのデータの表示 (SQL の場合) [18 ページ]
Interactive SQL を使用して、テーブルのデータを表示します。

関連情報

データベースオブジェクトの名前とプレフィクス [8 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.4.1　テーブルの作成

SQL Central を使用して、データベースにテーブルを作成します。

前提条件

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。

コンテキスト

CREATE TABLE...LIKE 構文を使用し、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 文に基づいてテーブルを作成したりすることもできます。

手順

1. SQL Central で、*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. 左ウィンドウ枠で テーブルを右クリックし、新規 > テーブル をクリックします。
3. テーブル作成ウィザードの指示に従います。
4. 右ウィンドウ枠で *[カラム]* タブをクリックして、テーブルに新しいカラムを作成します。
5. ファイル > 保存 をクリックします。

結果

新しいテーブルが、データベースに作成されます。

次のステップ

データをテーブルに入力またはロードします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

INSERT を使用したデータの追加 [501 ページ]
INSERT 文を使用したデータのインポート [604 ページ]

1.1.4.2 テーブル変更

カラムの追加、さまざまなカラム属性の変更、またはカラムの削除により、テーブルの構造またはカラム定義を変更します。

テーブルの変更とビューの依存性

テーブルを変更する前に、sa_dependent_views システムプロシージャを使用して、そのテーブルに依存するビューがあるかどうかを判断します。

従属ビューでテーブルのスキーマを変更するときは、ビューの種類によって、追加の手順がある場合があります。

従属した通常のビュー

テーブルのスキーマを変更すると、データベース内でテーブルの定義が更新されます。従属した通常のビューが存在する場合、データベースサーバはテーブル変更を実行後にそれらを自動的に再コンパイルします。テーブルのスキーマを変更した後で、データベースサーバが従属した通常のビューを再コンパイルできない場合、変更によってビュー定義が無効になったことが原因と考えられます。この場合は、ビュー定義を訂正する必要があります。

従属したマテリアライズドビュー

従属したマテリアライズドビューが存在する場合は、テーブルの変更前にそれらのビューを無効にし、テーブルの変更後にもう一度有効にする必要があります。テーブルのスキーマを変更した後で、従属した非マテリアライズドビューをもう一度有効にできない場合は、変更によってマテリアライズドビュー定義が無効になったことが原因と考えられます。この場合は、マテリアライズドビューを削除してから、有効な定義を使用してもう一度作成する必要があります。または、基となるテーブルに適切な変更を加えてから、マテリアライズドビューをもう一度有効にしてみてください。

テーブルの所有者の変更

ALTER TABLE 文または SQL Central を使用して、テーブルの所有者を変更します。テーブルの所有者を変更するときに、テーブル内の既存の外部キーとそれらが参照しているテーブルを保存するかどうかを指定します。すべての外部キーを削除するとテーブルが分離されますが、必要に応じてセキュリティが強化されます。明示的に付与された既存の権限を保存するかどうかも指定できます。セキュリティ上の目的で、明示的に付与された権限のうち、テーブルへのアクセスを許可しているものすべてを削除します。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。

このセクションの内容:

テーブルの変更 [14 ページ]
SQL Central を使用して、データベースのテーブルを変更します。たとえば、カラムの追加または削除、テーブルの所有者の変更を行うことができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブルの削除 [15 ページ]
SQL Central を使用して、必要なくなったときなどに、データベースからテーブルを削除できます。

関連情報

ビューの依存性 [50 ページ]
通常のビューの変更 [58 ページ]
マテリアライズドビューの作成 [72 ページ]

1.1.4.2.1 テーブルの変更

SQL Central を使用して、データベースのテーブルを変更します。たとえば、カラムの追加または削除、テーブルの所有者の変更を行うことができます。

前提条件

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 権限に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限
- ALTER ANY TABLE システム権限
- ALTER ANY OBJECT システム権限
- ALTER ANY OBJECT OWNER 権限 (テーブルの所有者を変更する場合) と ALTER ANY OBJECT システム権限の 1 つ、ALTER ANY TABLE システム権限またはテーブルに対する ALTER 権限

従属マテリアライズドビューがある場合はテーブルの変更は失敗します。あらかじめ従属マテリアライズドビューを無効にしておく必要があります。sa_dependent_views システムプロシージャを使用して、依存しているマテリアライズドビューがあるかどうかを決定します。

手順

1. SQL Central で、*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. 次のオプションのうちの 1 つを選択してください。

オプション	アクション
カラムの変更	1. 変更するテーブルをダブルクリックします。 2. 右ウィンドウ枠でカラムタブをクリックして、テーブルのカラムを変更します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプション	アクション
	3. ファイル » 保存 をクリックします。
テーブルの所有者の変更	テーブルを右クリックして、プロパティ » 今すぐ所有者を変更 をクリックして、テーブルの所有者を変更します。

結果

データベースで、テーブルの定義が更新されます。

次のステップ

テーブルを変更するため、マテリアライズドビューを無効にする場合、各マテリアライズドビューを再び有効にし、初期化してください。

関連情報

データ整合性 [727 ページ]
ビューの依存性 [50 ページ]
依存性とスキーマ変更 [51 ページ]
マテリアライズドビューの有効化または無効化 [75 ページ]

1.1.4.2.2 テーブルの削除

SQL Central を使用して、必要なくなったときなどに、データベースからテーブルを削除できます。

前提条件

所有者であるか、または DROP ANY TABLE と DROP ANY OBJECT のいずれかのシステム権限を持っている必要があります。

パブリケーションでアーティクルとして使用されているテーブルは削除できません。SQL Central でこれを実行しようとすると、エラーが発生します。また、従属ビューのあるテーブルを削除する場合は、追加の手順がある場合があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

従属マテリアライズドビューがある場合はテーブルの削除は失敗します。あらかじめ従属マテリアライズドビューを無効にしておく必要があります。sa_dependent_views システムプロシージャを使用して、依存しているマテリアライズドビューがあるかどうかを決定します。

手順

1. SQL Central で、*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. テーブルをダブルクリックします。
3. テーブルを右クリックして、*[削除]* をクリックします。
4. はいをクリックします。

結果

テーブルを削除すると、その定義がデータベースから削除されます。従属した通常のビューが存在する場合、データベースサーバはテーブル変更を実行後にそれらを再コンパイルしてもう一度有効にしようとします。失敗した場合は、テーブルの削除によってビューの定義が無効になったことが原因と考えられます。この場合は、ビュー定義を訂正する必要があります。

従属したマテリアライズドビューが存在する場合、その定義は有効でなくなっているため、以降の再表示は失敗します。この場合は、マテリアライズドビューを削除してから、有効な定義を使用してもう一度作成する必要があります。

テーブルのインデックスはすべて、削除されます。

テーブルを削除すると、COMMIT 文が実行されます。したがって、最後に COMMIT または ROLLBACK を実行した後の変更はすべて確定されます。

次のステップ

従属する通常ビューまたはマテリアライズドビューを削除します。またはその定義を変更して、削除されたテーブルの参照を削除します。

関連情報

ビューの依存性 [50 ページ]
依存性とスキーマ変更 [51 ページ]
マテリアライズドビューの有効化または無効化 [75 ページ]
通常のビューの変更 [58 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.4.3 テーブルでのデータの表示 (SQL Central の場合)

SQL Central を使用して、テーブルでデータを閲覧します。

前提条件

そのテーブルに対する 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。

手順

1. SQL Central で、*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. テーブルをダブルクリックします。
3. 右ウィンドウ枠で、*[データ]* タブをクリックします。

結果

テーブルのデータが、*[データ]* タブに表示されます。

次のステップ

データは *[データ]* タブで編集できます。

関連情報

通常のビュー内のデータのブラウズ [63 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.4.4 テーブルでのデータの表示 (SQL の場合)

Interactive SQL を使用して、テーブルのデータを表示します。

前提条件

そのテーブルに対する 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。

手順

table-name が、表示するデータを含んだテーブルである場合、次のような文を実行します。

```
SELECT * FROM table-name;
```

結果

テーブルのデータが、結果ウィンドウ枠に表示されます。

次のステップ

結果ウィンドウ枠のデータを編集できます。

1.1.5 テンポラリテーブル

テンポラリテーブルはテンポラリファイルに格納されます。

他の DB 領域のページと同様に、テンポラリファイルのページはキャッシュできます。

テンポラリテーブルに対する操作はトランザクションログに書き込まれません。テンポラリテーブルには、ローカルテンポラリテーブルとグローバルテンポラリテーブルの 2 種類があります。

ローカルテンポラリテーブル

ローカルテンポラリテーブルは、接続の間だけ、または複合文内で定義されている場合はその複合文が使われている間だけしか存在しません。

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

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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。

ローカルテンポラリテーブルにインデックスを作成するときに、auto_commit_on_create_local_temp_index オプションを OFF に設定すると、テーブルにインデックスを作成する前にコミットはされません。

グローバルテンポラリテーブル

グローバルテンポラリテーブルは、DROP TABLE 文を使用して明示的に削除しないかぎり、データベース内に残ります。同じアプリケーションまたは異なるアプリケーションからの複数の接続は、グローバルテンポラリテーブルを同時に使用できます。グローバルテンポラリテーブルの特性は次のとおりです。

- テーブルの定義はカタログに記録され、テーブルが明示的に削除されるまで保持されます。
- テーブルでの挿入、更新、削除は、トランザクションログに記録されません。
- テーブルのカラム統計は、データベースサーバによってメモリ内に保持されます。

グローバルテンポラリテーブルには、非共有と共有の 2 種類があります。通常、グローバルテンポラリテーブルは非共有です。つまり、各接続はテーブル内で各自のローしか認識しません。接続が終了すると、その接続のローはテーブルから削除されます。

グローバルテンポラリテーブルが共有されると、テーブルのすべてのデータがすべての接続で共有されます。共有されたグローバルテンポラリテーブルを作成するには、テーブルの作成時に SHARE BY ALL 句を指定します。共有されたグローバルテンポラリテーブルには、グローバルテンポラリテーブルの一般的な特性だけでなく、次の特性が適用されます。

- 明示的に削除されるまで、またはデータベースが停止するまで、テーブルのコンテンツは持続します。
- データベースの起動時、テーブルは空です。
- テーブルでのローのロック処理動作は、ベーステーブルの場合と同じです。

非トランザクション指向のテンポラリテーブル

テンポラリテーブルを非トランザクション指向として宣言するには、CREATE TABLE 文の NOT TRANSACTIONAL 句を使用します。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や、テーブルに多くのローが含まれる場合は、NOT TRANSACTIONAL が有用です。非トランザクション指向テンポラリテーブルへの変更は、COMMIT または ROLLBACK の影響を受けません。

このセクションの内容:

グローバルテンポラリテーブルの作成 [20 ページ]
SQL Central を使用したグローバルテンポラリテーブルの作成

プロシージャ内でのテンポラリテーブルの参照 [21 ページ]
プロシージャ間でテンポラリテーブルを共有すると、テーブル定義が矛盾している場合に問題が発生する場合があります。

関連情報

トランザクションと独立性レベル [757 ページ]
ロックの仕組み [782 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.5.1 グローバルテンポラリテーブルの作成

SQL Central を使用したグローバルテンポラリテーブルの作成

前提条件

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。

コンテキスト

ローのロックを心配せずにデータを処理し、トランザクションと再実行ログで必要ないアクティビティを減らす場合、この作業を実行してグローバルテンポラリテーブルを作成します。

DECLARE LOCAL TEMPORARY TABLE...LIKE 構文を使用し、別のテーブル定義に直接基づくテンポラリ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 文に基づいてテーブルを作成したりすることもできます。

手順

1. SQL Central で、*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. **テーブル**を右クリックし、 **新規** > **グローバルテンポラリテーブル** をクリックします。
3. **グローバルテンポラリテーブル作成ウィザード**の指示に従います。
4. 右ウィンドウ枠で *[カラム]* タブをクリックして、テーブルを設定します。
5. **ファイル** > **保存** をクリックします。

結果

グローバルテンポラリテーブルが作成されます。グローバルテンポラリテーブル定義は、指定された時点で削除されるまでデータベースに格納され、他の接続に使用できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.5.2 プロシージャ内でのテンポラリテーブルの参照

プロシージャ間でテンポラリテーブルを共有すると、テーブル定義が矛盾している場合に問題が発生する場合があります。

たとえば、procA と procB という 2 つのプロシージャがあり、その両方がテンポラリテーブル temp_table を定義して、sharedProc という名前の別のプロシージャを呼び出すとします。procA と procB のどちらもまだ呼び出されていないため、テンポラリテーブルはまだ存在しません。

ここで、procA の temp_table の定義が procB の定義と少し異なるとします。両方とも同じカラム名と型を使用していますが、カラムの順序が異なります。

procA を呼び出すと、予期した結果が返されます。一方で、procB を呼び出すと、異なる結果が返されます。

これは、procA が呼び出されたときに temp_table が作成され、その後で sharedProc が呼び出されたためです。sharedProc が呼び出されると、内部の SELECT 文が解析および検証され、その後、別の SELECT 文が実行されたときに使用できるように、解析された文の表現がキャッシュされます。キャッシュされたバージョンは、procA のテーブル定義のカラムの順序を反映しています。

procB を呼び出すと temp_table が再作成されますが、カラムの順序が異なります。procB が sharedProc を呼び出すと、データベースサーバは SELECT 文のキャッシュされた表現を使用します。そのため、結果が異なります。

次のいずれかを実行すると、このような状況の発生を防ぐことができます。

- このような方法で使用されるテンポラリテーブルは、一致するように定義します
- 代わりにグローバルテンポラリテーブルを使用します

1.1.6 計算カラム

計算カラムとは、同一ロー内にある従属カラムと呼ばれる他のカラムの値を参照する式のことです。

計算カラムは、1 つまたは複数の従属カラムの値を含む複雑な式をインデックス化するような場合に特に便利です。データベースサーバは、計算カラムの COMPUTE 式と一致する式が認識できる場合は、常に計算カラムを使用します。これには SELECT リストや述部が含まれます。ただし、クエリ式に CURRENT TIMESTAMP などの特別値が含まれる場合は、この一致は起こりません。

TIMESTAMP WITH TIME ZONE カラムは、計算カラムとして使用しないでください。time_zone_adjustment オプションの値は、ロケーションと日付に基づいて、接続ごとに異なる値となる場合があるため、これらの値が計算されると、不正な結果や予期しない動作が発生することがあります。

クエリの最適化中、SQL Anywhere オプティマイザは、複雑な式を含む述部を、単純に計算カラムの定義を参照する述部へ自動的に変換しようとします。たとえば、クエリに製品出荷に関する一覧情報で構成されたテーブルを要求したとします。

```
CREATE TABLE Shipments(
   ShipmentID INTEGER NOT NULL PRIMARY KEY,
   ShipmentDate TIMESTAMP,
   ProductCode CHAR(20) NOT NULL,
   Quantity INTEGER NOT NULL,
   TotalPrice DECIMAL(10,2) NOT NULL
);
```

特に、クエリは平均コストが 2 ～ 4 ドルである製品出荷を返します。クエリは、次のように記述できます。

```
SELECT *
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
   FROM Shipments
   WHERE ( TotalPrice / Quantity ) BETWEEN 2.00 AND 4.00;
```

しかし、上記のクエリで、WHERE 句の述部は単一ベースのカラムを参照しないため、検索引数が使用できません。

Shipments テーブルのサイズが比較的大きい場合、インデックス検索の方が逐次スキャンより適している場合があります。インデックス検索を向上させるには、次のように Shipments テーブルに AverageCost という名前の計算カラムを作成してから、そのカラムにインデックスを作成します。

```
ALTER TABLE Shipments
   ADD AverageCost DECIMAL(21,13)
   COMPUTE( TotalPrice / Quantity );
 CREATE INDEX IDX_average_cost
   ON Shipments( AverageCost ASC );
```

計算カラムのタイプを選択することは重要です。クエリ内の式のデータ型が計算カラムのデータ型と正確に一致した場合、SQL Anywhere のオプティマイザは複雑な式のみを計算カラムに置き換えます。式のタイプを判別するために、SQL 文内の式のタイプを返す EXPRTYPE 組み込み関数を使用できます。

```
SELECT EXPRTYPE(
   'SELECT ( TotalPrice/Quantity ) AS X FROM Shipments', 1 )
   FROM SYS.DUMMY;
```

Shipments テーブルに対して、上記のクエリは decimal(21,13) を返します。最適化中に、SQL Anywhere オプティマイザは上記のクエリを次のように書き換えます。

```
SELECT *
   FROM Shipments
   WHERE AverageCost
   BETWEEN 2.00 AND 4.00;
```

この場合 WHERE 句内の述部は検索引数可能なものとなり、オプティマイザは、新しい IDX_average_cost インデックスを使用して、クエリのアクセスプラン用のインデックススキャンを選択できます。

このセクションの内容:

計算カラムの変更 [23 ページ]
　計算カラムで使用された式を変更または削除します。

計算カラムへの挿入と更新 [24 ページ]
　計算カラムへの挿入と更新に関して、考慮が必要なことがいくつかあります。

計算カラムの再計算 [25 ページ]
　計算カラムの値は、ローが挿入され更新されると自動的にデータベースサーバによって維持されます。

関連情報

クエリの述部 [162 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.6.1　計算カラムの変更

計算カラムで使用された式を変更または削除します。

前提条件

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 権限に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限
- ALTER ANY TABLE システム権限
- ALTER ANY OBJECT システム権限

手順

1. データベースに接続します。
2. 次のような ALTER TABLE 文を実行して、計算カラムに使用する式を変更します。

```
ALTER TABLE table-name
ALTER column-name
SET COMPUTE ( new-expression );
```

3. カラムを通常のカラム(未計算)に変換するには、次のような ALTER TABLE 文を実行します。

```
ALTER TABLE
table-name
ALTER column-name
DROP COMPUTE;
```

結果

カラムの計算を変更する場合、この文を実行するとカラムは再計算されます。

計算済みのカラムを通常のカラム(未計算)に変更する場合、文を実行してもカラム内の既存の値は変更されないので、自動的に更新されません。

例

テーブル alter_compute_test を作成してデータを移植し、次の文を実行してテーブルのクエリを選択します。

```
CREATE TABLE alter_compute_test (
   c1 INT,
   c2 INT
) ;
INSERT INTO alter_compute_test (c1) VALUES(100);
SELECT * FROM alter_compute_test ;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

カラム c2 は NULL 値を返します。カラム c2 を変更して計算カラムとして、カラムにデータを移植して、alter_compute_test テーブルで別の SELECT 文を実行します。

```
ALTER TABLE alter_compute_test
   ALTER c2
   SET COMPUTE ( DAYS ( '2001-01-01' , CURRENT DATE ) )
INSERT INTO alter_compute_test (c1) VALUES(200) ;
SELECT * FROM alter_compute_test ;
```

これで、カラム c2 には 2001-01-01 以降の日数が含まれました。次に、計算カラムとしないようにカラム c2 を変更します。

```
ALTER TABLE alter_compute_test
ALTER c2
DROP COMPUTE ;
```

関連情報

計算カラムの再計算 [25 ページ]

1.1.6.2 計算カラムへの挿入と更新

計算カラムへの挿入と更新に関して、考慮が必要なことがいくつかあります。

直接挿入と直接更新

INSERT または UPDATE 文では計算カラムの値を指定できますが、値は無視されます。サーバは COMPUTE の指定に基づいて計算カラムの値を計算し、その計算カラム値を INSERT または UPDATE 文の指定された値の場所に使用します。

カラムの依存性

たとえば、NULL 値を NULL 値以外の値に変更するなど、計算カラムの定義で参照されるカラムの値を設定するときにトリガを使用しないことを強くお奨めします。これは、計算カラムの値が意図した計算内容を反映していない可能性があるためです。

カラム名のリスト

計算カラムのあるテーブルに対する INSERT 文では、常にカラム名を明示的に指定してください。

トリガ

計算カラムにトリガを定義すると、そのカラムに影響するすべての INSERT 文または UPDATE 文はトリガを起動します。

LOAD TABLE 文では、計算カラムのオプション計算が可能です。ロード操作中に計算を行わないようにすると、複雑なアンロード / 再ロード手順を速くすることができます。これは、COMPUTE 式が CURRENT TIMESTAMP などの非確定値を参照する場合でも、計算カラムの値が一定である必要がある場合に便利です。

値を変更すると計算カラムの値がカラム定義と一致しなくなる場合があるため、トリガの従属カラムの値は変更しないでください。

計算カラム x が NOT NULL と宣言されたカラム y に依存する場合、y に NULL を設定しようとすると、トリガが起動される前にエラーが発生し、拒否されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.6.3 計算カラムの再計算

計算カラムの値は、ローが挿入され更新されると自動的にデータベースサーバによって維持されます。

ほとんどのアプリケーションでは、計算カラム値を直接更新したり挿入したりする必要はありません。

計算カラムは、次の状況で再計算されます。

- いずれかのカラムが削除、追加、または名前が変更された場合
- テーブルは、任意のカラムのデータ型を修正する ALTER TABLE 文または COMPUTE 句によって変更されます。
- ローが挿入された場合
- ローが更新された場合

計算カラムは、次の状況では再計算されません。

- テーブルの名前が変更された場合
- 計算カラムが問い合わせされている。
- 計算カラムは (サブクエリまたはユーザ定義関数を使用した) 他のローの値に依存しており、しかもこれらのローは変更されている。

1.1.7 プライマリキー

リレーショナルデータベース内の各テーブルには、プライマリキーが必要です。プライマリキーとは 1 つのカラム、またはカラムのセットで、各ローをユニークに識別します。

1 つのテーブル内で、2 つのローが同じプライマリキーの値を持つことはできません。また、プライマリキーのカラムを NULL 値にすることはできません。

ベーステーブルとグローバルテンポラリテーブルのみがプライマリキーを持つことができます。宣言されたテンポラリテーブルでは、NOT NULL カラムのセットにユニークインデックスを作成し、プライマリキーのセマンティックを模倣できます。

プライマリキーや、一意性制約があるカラムには FLOAT や DOUBLE などの概数値データ型を使用しないでください。概数値データ型は、算術演算後の丸め誤差がでます。

CLUSTERED 句を使用して、プライマリキー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。

注記

プライマリキーカラムの順序は、CREATE TABLE (または ALTER TABLE) 文のプライマリキー宣言で指定したカラム順序によって決まります。個別のカラムのソート順 (昇順または降順) を指定することもできます。これらのソート順指定は、プライマリキーインデックスを作成するときにデータベースサーバによって使用されます。

プライマリキー内のカラムの順序は、参照整合性制約のカラムの順序を指定しません。外部キー宣言で、異なるカラム順序、異なるソート順序を指定できます。

例

SQL Anywhere サンプルデータベースでは、Employees テーブルに従業員の個人情報が格納されています。このテーブルにはプライマリキーカラム EmployeeID があり、各従業員に割り当てられたユニークな ID 番号が入っています。ID 番号が入っている単一のカラムにプライマリキーを割り当てるのが一般的な方法です。名前やその他の識別子は、必ずしもユニークではないため、ID 番号の方が適しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere サンプルデータベースの SalesOrderItems テーブルのプライマリキーは、より複雑です。このテーブルには、会社からの注文に含まれる個々の項目に関する情報が入っており、次のカラムで構成されています。

ID

その注文項目が含まれている注文を識別する注文番号

LineID

注文内での注文項目を識別する行番号

ProductID

受注した製品を識別する製品 ID

Quantity

受注した項目の数量

ShipDate

受注品が出荷された日付

特定の注文項目はその注文項目が含まれている注文番号と、注文内での行番号によって識別されます。この 2 つの番号は、ID カラムと LineID カラムに格納されています。注文項目が同一の ID 値を共有する場合 (1 回の注文に複数の注文項目がある場合) と、注文項目が同一の LineID 番号を共有する場合 (各注文における最初の注文項目はすべて LineID が 1) があります。どの注文項目も、これら 2 つの値を共有することはないので、プライマリキーは 2 つのカラムで構成されています。

このセクションの内容:

プライマリキーの管理 (SQL Central の場合) [26 ページ]
SQL Central を使用してプライマリキーを管理して、テーブルのクエリパフォーマンスを向上させます。

プライマリキーの管理 (SQL の場合) [27 ページ]
SQL を使用してプライマリキーを管理して、テーブルのクエリパフォーマンスを向上させます。

関連情報

クラスタドインデックス [39 ページ]

プライマリキーによるエンティティ整合性の確保 [748 ページ]

1.1.7.1 プライマリキーの管理 (SQL Central の場合)

SQL Central を使用してプライマリキーを管理して、テーブルのクエリパフォーマンスを向上させます。

前提条件

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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ALTER ANY OBJECT システム権限
- ALTER ANY INDEX と ALTER ANY TABLE のシステム権限
- そのテーブルに対する ALTER 権限と REFERENCES に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テーブルをダブルクリックします。
3. テーブルを右クリックし、次のいずれかのオプションを選択します。

オプション	アクション
プライマリキーの作成または変更	プライマリキーの設定をクリックして、プライマリキー設定ウィザードの指示に従います。
プライマリキーを削除します。	テーブルの カラムウィンドウ枠で、*PKey* カラムからチェックマークをクリアして、保存をクリックします。

結果

プライマリキーが追加、変更、または削除されます。

関連情報

プライマリキーによるエンティティ整合性の確保 [748 ページ]
プライマリキーの管理 (SQL の場合) [27 ページ]

1.1.7.2 プライマリキーの管理 (SQL の場合)

SQL を使用してプライマリキーを管理して、テーブルのクエリパフォーマンスを向上させます。

前提条件

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

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ALTER ANY OBJECT システム権限

プライマリキーのカラムに NULL 値を含むことはできません。

手順

データベースに接続します。

オプション	アクション
プライマリキーを作成します。	ALTER TABLE `table-name` ADD PRIMARY KEY `(column-name)` 文を実行します。
プライマリキーを削除します。	ALTER TABLE `table-name` DROP PRIMARY KEY 文を実行します。
プライマリキーを変更します。	テーブルの新しいプライマリキーを作成する前に、既存のプライマリキーを削除します。

結果

プライマリキーが追加、削除、または変更されます。

例

次の文は、Skills という名前のテーブルを作成し、SkillID カラムをプライマリキーとして割り当てます。

```
CREATE TABLE Skills (
   SkillID INTEGER NOT NULL,
   SkillName CHAR( 20 ) NOT NULL,
   SkillType CHAR( 20 ) NOT NULL,
   PRIMARY KEY( SkillID )
);
```

プライマリキー値は、テーブル内のローごとにユニークである必要があります。この例では、特定の SkillID を持つローを複数設定できません。テーブルの各ローは、そのプライマリキーによってユニークに識別されます。

プライマリキーに SkillID カラムと SkillName カラムを組み合わせて使用するようにプライマリキーを変更する場合は、作成したプライマリキーを削除してから、新しいプライマリキーを追加します。

```
ALTER TABLE Skills DELETE PRIMARY KEY;
ALTER TABLE Skills ADD PRIMARY KEY ( SkillID, SkillName );
```

関連情報

プライマリキーによるエンティティ整合性の確保 [748 ページ]
プライマリキーの管理 (SQL Central の場合) [26 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.8 外部キー

外部キーは、カラムまたはカラムのセットで構成されており、キー値が一致するプライマリテーブル内のローの参照を表します。

外部キーはベーステーブルのみで使用できます。テンポラリテーブル、グローバルテンポラリテーブル、ビュー、またはマテリアライズドビューでは使用できません。外部キーは参照整合性制約と呼ばれることもあります。外部キーを含むベーステーブルは参照元テーブルと呼ばれ、プライマリキーを含むテーブルは参照先テーブルと呼ばれます。

外部キーが NULL 入力可能な場合、参照先テーブル内に一致するプライマリキー値がない外部ローが存在する場合があるため、関係はオプションです。これは、プライマリキーと UNIQUE 制約のカラムはどちらも NULL にできないためです。外部キーカラムが NOT NULL と宣言された場合、関係は必須であり、参照元テーブルの各ローには、参照先テーブルにプライマリキーとして存在する外部キー値が含まれている必要があります。

外部キーと孤立したロー

参照整合性を維持するには、不一致の NULL でない外部キー値がデータベースに含まれないようにします。参照整合性に違反する外部ローは、参照先テーブルのプライマリキー値と一致しないため、オーファンと呼ばれます。オーファンは次の操作で作成されます。

- 参照先テーブルのプライマリキー値と一致しない NULL でない値が外部キーカラムにある参照元テーブルで、ローを挿入または更新します。
- プライマリテーブルでのローの更新または削除すると、一致するプライマリキー値を含まない参照元テーブルに少なくとも 1 つのローができます。

データベースサーバでは、孤立したローを作成しないようにして、参照整合性違反を防ぎます。

複合外部キー

複合キーと呼ばれる、複数カラムのプライマリキーと外部キーもサポートされています。複合外部キーでも NULL 値は不一致を示しますが、オーファンの識別方法は MATCH 句での参照整合性制約の定義方法によって異なります。

外部キーインデックスとソートの程度の順序

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。外部キーカラムの順序はプライマリキーカラムの順序を反映している必要はなく、プライマリキーインデックスのソートの順序が外部キーインデックスのソートの順序と一致している必要もありません。外部キーインデックスでのインデックス付けされた各カラムのソート (昇順または降順) はカスタマイズでき、外部キーインデックスのソートの順序が、アプリケーションの特定の SQL クエリに必要な、そのクエリの文の ORDER BY 句で指定されたソートの順序に一致するようにできます。各カラムのソートの順序は、外部キー制約を設定するときに指定できます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

例 1: SQL Anywhere サンプルデータベースには、従業員情報を格納しているテーブルと、部署情報を格納しているテーブルがあります。Departments テーブルには、次のカラムがあります。

DepartmentID

部署の ID 番号。これがテーブルのプライマリキーになります。

DepartmentName

部署の名前。

DepartmentHeadID

部長の従業員 ID。

特定の従業員の所属部署名を探せるように、その従業員の部署名を Employees テーブルに入力しておく必要はありません。その代わり Employees テーブルには、Departments テーブルの DepartmentID 値の 1 つと一致する値の入った DepartmentID カラムがあります。

Employees テーブルの DepartmentID カラムは、Departments テーブルに対する外部キーです。外部キーは、対応するプライマリキーを持つテーブル内の特定のローを参照します。

そのため、Employees テーブル (関係付けの外部キーを持つ) を外部テーブルまたは参照元テーブルと呼びます。Departments テーブル (参照先のプライマリキーを持つ) は、プライマリテーブルまたは参照先テーブルと呼びます。

例 2: 次の文を実行して、複合プライマリキーを作成します。

```
CREATE TABLE pt(
     pk1 INT NOT NULL,
     pk2 INT NOT NULL,
     str VARCHAR(10),
     PRIMARY KEY ( pk1, pk2 ));
```

次の文は、プライマリキーとカラムの順序が異なる外部キーと、外部キーカラムに対して異なるソートの程度を作成します。これは、外部キーインデックスを作成する場合に使用されます。

```
CREATE TABLE ft1(
     fpk INT PRIMARY KEY,
     ref1 INT,
     ref2 INT );
```

```
ALTER TABLE ft1 ADD FOREIGN KEY ( ref2 ASC, ref1 DESC)
     REFERENCES pt ( pk2, pk1 ) MATCH SIMPLE;
```

次の文を実行して、プライマリキーとカラムの順序が同じだが、外部キーインデックスに対してソートの程度が異なる外部キーを作成します。また、この例では、MATCH FULL 句を使用して、両方のカラムが NULL だった場合に孤立したローになることを指定します。UNIQUE 句は、NULL ではないカラムの pt テーブルと ft2 テーブルの間に 1 対 1 の関係を適用します。

```
CREATE TABLE ft2(
     fpk INT PRIMARY KEY,
     ref1 INT,
     ref2 INT );
```

```
ALTER TABLE ft2 ADD FOREIGN KEY ( ref1, ref2 DESC )
     REFERENCES pt ( pk1, pk2 ) MATCH UNIQUE FULL;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

外部キーの作成 (SQL Central の場合) [31 ページ]
テーブル間に外部キー関係を作成します。

外部キーの作成 (SQL の場合) [32 ページ]
Interactive SQL では、CREATE TABLE 文と ALTER TABLE 文を使用して、外部キーを作成および修正します。

関連情報

参照整合性 [748 ページ]

1.1.8.1 外部キーの作成 (SQL Central の場合)

テーブル間に外部キー関係を作成します。

前提条件

そのテーブルに対する 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。

さらに、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 権限に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限
- ALTER ANY TABLE システム権限
- ALTER ANY OBJECT システム権限

コンテキスト

外部キーの関係は制約として機能します。データベースサーバは、子テーブルに挿入した新しいローに対して、外部キーカラムに挿入している値がプライマリテーブルのプライマリキーの値と一致するかどうかを確認します。外部テーブルの作成時に外部キーを作成する必要はありません。外部キーは自動的に作成されます。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テーブルをダブルクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

3. 作成するテーブルまたは外部キーを選択します。
4. 右ウィンドウ枠で、*[制約]* タブをクリックします。
5. 外部キーを作成します。
    a. ファイル > 新規 > 外部キー をクリックします。
    b. 外部キー作成ウィザードの指示に従います。

結果

SQL Central では、テーブルの外部キーは、テーブルが選択されている場合、右ウィンドウ枠内の制約タブに表示されます。テーブルが更新され、その外部キー定義がテーブルに含まれます。

次のステップ

ウィザードを使用して外部キーを作成するときに、その外部キーのプロパティを設定できます。外部キーを作成した後でプロパティを表示するには、制約タブで外部キーを選択し、ファイル > プロパティ をクリックします。

参照元制約タブでテーブルを選択してから ファイル > プロパティ をクリックすると、参照元外部キーのプロパティを表示できます。

所定のテーブルを参照するテーブルのリストを表示するには、*[テーブル]* でテーブルを選択し、右ウィンドウ枠で *[参照元制約]* タブをクリックします。

1.1.8.2 外部キーの作成 (SQL の場合)

Interactive SQL では、CREATE TABLE 文と ALTER TABLE 文を使用して、外部キーを作成および修正します。

前提条件

テーブルの所有者に基づいて外部キーを作成するのに必要な権限は次のとおりです。

参照されるテーブル (プライマリキー) と参照するテーブル (外部キー) を所有

権限は必要ありません。

参照されるテーブルではなく、参照するテーブルを所有

そのテーブルに対する REFERENCES 権限を持っているか、または CREATE ANY INDEX と CREATE ANY OBJECT のどちらか一方のシステム権限を持っていることが必要です。

参照するテーブルではなく、参照されるテーブルを所有

- ALTER ANY OBJECT または ALTER ANY TABLE のシステム権限が必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- あるいは、そのテーブルに対する ALTER 権限に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限が必要です。
- さらに、そのテーブルに対する SELECT 権限または SELECT ANY TABLE システム権限も必要です。

どちらのテーブルも所有していない

- そのテーブルに対する REFERENCES 権限を持っているか、または CREATE ANY INDEX と CREATE ANY OBJECT のどちらか一方のシステム権限を持っていることが必要です。
- ALTER ANY OBJECT または ALTER ANY TABLE のシステム権限が必要です。
- あるいは、そのテーブルに対する ALTER 権限に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限が必要です。
- さらに、そのテーブルに対する SELECT 権限または SELECT ANY TABLE システム権限も必要です。

そのテーブルに対する 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。

さらに、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 権限に加え、COMMENT ANY OBJECT、CREATE ANY OBJECT、または CREATE ANY TABLE のシステム権限
- ALTER ANY TABLE システム権限
- ALTER ANY OBJECT システム権限

コンテキスト

これらの文によって、カラムの制約や検査など、多くのテーブル属性を設定できます。

外部テーブルの作成時に外部キーを作成する必要はありません。外部キーは自動的に作成されます。

手順

1. データベースに接続します。
2. 次のような ALTER TABLE 文を実行します。

```
ALTER TABLE table-name ADD FOREIGN KEY foreign-key-name
( column-name ASC ) REFERENCES table-name ( column-name )
```

結果

テーブルが更新され、その外部キー定義がテーブルに含まれます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

次の例では、スキルの一覧を格納する Skills というテーブルを作成し、次に Skills テーブルに対して外部キーの関係を持つ EmployeeSkills というテーブルを作成します。EmployeeSkills.SkillID には、Skills テーブルのプライマリキーカラム (Id) と外部キーの関係があります。

```
CREATE TABLE Skills (
   Id INTEGER PRIMARY KEY,
   SkillName CHAR(40),
   Description CHAR(100)
);
CREATE TABLE EmployeeSkills (
   EmployeeID INTEGER NOT NULL,
   SkillID INTEGER NOT NULL,
   SkillLevel INTEGER NOT NULL,
   PRIMARY KEY( EmployeeID ),
   FOREIGN KEY (SkillID) REFERENCES Skills ( Id )
);
```

テーブルを作成した後で、ALTER TABLE 文を使用して外部キーを追加することもできます。次の例では、前の例で作成したテーブルに似たテーブルを作成します。ただし、テーブルの作成後に外部キーを追加します。

```
CREATE TABLE Skills2 (
   ID INTEGER PRIMARY KEY,
   SkillName CHAR(40),
   Description CHAR(100)
);
CREATE TABLE EmployeeSkills2 (
   EmployeeID INTEGER NOT NULL,
   SkillID INTEGER NOT NULL,
   SkillLevel INTEGER NOT NULL,
   PRIMARY KEY( EmployeeID ),
);
ALTER TABLE EmployeeSkills2
   ADD FOREIGN KEY SkillFK ( SkillID )
   REFERENCES Skills2 ( ID );
```

外部キーを作成するとき、そのプロパティを指定できます。たとえば、次の文は例 2 と同じ外部キーを作成しますが、この外部キーは、データの更新または削除に対する制限がある NOT NULL として定義されています。

```
ALTER TABLE Skills2
ADD NOT NULL FOREIGN KEY SkillFK ( SkillID )
REFERENCES Skills2 ( ID )
ON UPDATE RESTRICT
ON DELETE RESTRICT;
```

外部キーカラム名とプライマリキーのカラム名とは、1 対 1 で対応する 2 つのリスト位置に従ってペアになります。外部キーの定義時にプライマリテーブルのカラム名が指定されていない場合、プライマリキーカラムが使用されます。たとえば、次のようにして 2 つのテーブルを作成するとします。

```
CREATE TABLE Table1( a INT, b INT, c INT, PRIMARY KEY ( a, b ) );
CREATE TABLE Table2( x INT, y INT, z INT, PRIMARY KEY ( x, y ) );
```

次に、外部キー fk1 を作成し、2 つのテーブル間でのカラムのペア方法を厳密に指定します。

```
ALTER TABLE Table2 ADD FOREIGN KEY fk1( x,y ) REFERENCES Table1( a, b );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の文を使用して、外部テーブルカラムのみを指定することで、2 つ目の外部キー fk2 を作成します。データベースサーバは、これらの 2 つのカラムを、プライマリテーブル上のプライマリキーにある最初の 2 つのカラムと自動的にペアにします。

```
ALTER TABLE Table2 ADD FOREIGN KEY fk2( x, y ) REFERENCES Table1;
```

次の文を使用して、プライマリテーブルと外部テーブルのいずれに対してもカラムを指定せずに外部キーを作成します。

```
ALTER TABLE Table2 ADD FOREIGN KEY fk3 REFERENCES Table1;
```

参照元カラムを指定していないため、データベースサーバは、プライマリテーブル (Table1) のカラムと同じ名前を持つカラムを外部テーブル (Table2) で検索します。同じ名前のカラムが存在する場合、データベースサーバはデータ型が一致することを確認し、それらのカラムを使用して外部キーを作成します。カラムが存在しない場合は、Table2 に作成されます。この例では、Table2 には a および b というカラムが存在しないため、Table1.a および Table1.b と同じデータタイプでこれらのカラムが作成されます。これらの自動的に作成されたカラムは外部テーブルのプライマリキーの一部にはなりません。

関連情報

外部キーの作成 (SQL Central の場合) [31 ページ]

1.1.9 インデックス

インデックスは、テーブルの 1 カラムまたは複数のカラムについてローに順序を付けます。

電話帳のように、インデックスは最初に姓でソートし、次に同じ姓の人を名前でソートします。この順序は、特定の姓を持つ人の電話番号をすばやく検索できますが、特定の住所から電話番号を検索する場合には意味がありません。同様に、データベースインデックスは、特定のカラムを 1 つまたは複数検索する場合にのみ役立ちます。

インデックスの有用性は、テーブルのサイズが大きくなるにつれて増大します。住所から電話番号を検索する速度は電話帳の厚さに比例しますが、姓で検索する場合は電話帳のサイズにあまり関係ないのと同じです。K. Kaminski を検索する場合、厚い電話帳と薄い電話帳ではほとんど時間は変わりません。

オプティマイザでは、自動的にインデックスを使用して、データベースの文のパフォーマンスを改善できる場合は改善します。ローが削除、更新、挿入された場合は、自動的にインデックスの更新が行われます。クエリの作成時にインデックスヒントを使用してインデックスを明示的に参照できますが、その必要はありません。

インデックスの作成にはいくつか欠点があります。特に、カラム内のデータが変更された場合はインデックスをテーブル自体とともに管理する必要があるため、挿入、更新、削除のパフォーマンスがインデックスによって影響される場合があります。このため、不要なインデックスは削除してください。インデックスコンサルタントを使用して、不要なインデックスを識別します。

作成するインデックスの決定

データベースに適切なインデックスセットを選択することは、パフォーマンスを最適化する上で重要です。適切なセットを識別することは、労力を要する作業でもあります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

インデックスの作成が必要かどうかは、単純な計算式では判断できません。インデックス検索の利点と、そのインデックスの管理に伴うオーバヘッドのトレードオフを考慮してください。次の要素を考慮して、インデックスの作成が必要かどうかを判断できます。

キーとユニークなカラム

データベースサーバは、プライマリキー、外部キー、ユニークなカラムのインデックスを自動的に作成します。これらのカラムについては、インデックスを追加して作成しないでください。ただし、複合キーの場合は、追加インデックスで強化できることがあります。

検索頻度

特定のカラムが頻繁に検索される場合は、そのカラムのインデックスを作成するとパフォーマンスを向上できます。検索頻度の低いカラムにインデックスを作成しても意味がありません。

テーブルのサイズ

多数のローを持つ比較的大きなテーブルのインデックスを作成すると、比較的小さなテーブルのインデックスの場合よりも多くの利点が得られます。たとえば、ローが 20 しかないテーブルの場合、逐次スキャンにはインデックスルックアップと同程度の時間しかかからないため、インデックスを作成しても利点は得られません。

更新回数

インデックスは、テーブルに対してローが挿入または削除されたり、インデックスカラムが更新されたりするたびに、更新されます。カラムにインデックスがあると、挿入、更新、削除のパフォーマンスが低下します。更新頻度の高いデータベースのインデックスは、読み込み専用データベースの場合より少なくなるようにしてください。

領域の注意事項

インデックスはデータベース内の領域を占有します。データベースサイズが重要な場合は、インデックスの作成を控えてください。

データ分散

インデックスルックアップから返される値が多すぎると、逐次スキャンよりも高コストになります。データベースサーバは、この条件を認識するとインデックスを使用しません。たとえば、SQL Anywhere サンプルデータベース内の Employees.Sex のように、値が 2 つしかないカラムについては、インデックスを使用しません。このため、固有値が少数しかないカラムのインデックスは作成しないでください。

インデックスを作成するときは、カラムを指定する順序は、インデックスでカラムが出現する順序になります。インデックス定義でカラム名を重複参照することはできません。

注記

インデックスコンサルタントは、適切なインデックス選択を支援するためのツールです。インデックスコンサルタントは、単一のクエリまたは一連の操作を分析して、データベースに追加するインデックスを推奨します。また、使用されていないインデックスを通知します。

テンポラリテーブルのインデックス

インデックスは、ローカルとグローバルの両方のテンポラリテーブル上で作成できます。テンポラリテーブルが大きく、ソートされた順序またはジョインで数回アクセスされることが予想される場合は、インデックスを作成します。そのような予想がない状況では、クエリを処理するパフォーマンスの改善よりも、インデックスを作成し削除するコストの方が上回ってしまいます。

このセクションの内容:

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

複合インデックス [37 ページ]
複数のカラムに対するインデックスは、複合インデックスと呼ばれます。

クラスタドインデックス [39 ページ]
インデックスのクラスタ化を宣言することで、大規模なインデックススキャンのパフォーマンスを改善することができます。

インデックスの作成 [40 ページ]
ベーステーブルのインデックス、テンポラリテーブル、マテリアライズドビューを作成します。

インデックスの検証 [42 ページ]
インデックスで参照されているすべてのローが、実際にテーブルに存在するかどうか検証します。

インデックスの再構築 [42 ページ]
テーブルまたはマテリアライズドビューでの広範囲の挿入と削除によりフラグメント化されたインデックスを再構築します。

インデックスの削除 [44 ページ]
インデックスが必要なくなった場合、または、プライマリキーまたは外部キーの一部であるカラムの定義を変更する必要がある場合、インデックスを削除します。

高度: カタログ内のインデックス情報 [45 ページ]
カタログにはデータベースのインデックスに関する情報を提供する複数のシステムテーブルがあります。

高度: 論理インデックスと物理インデックス [46 ページ]
ソフトウェアでは論理インデックスと物理インデックスがサポートされています。

高度: インデックスの選択性とインデックスファンアウト [47 ページ]
インデックスの選択性とは、追加データを読み込まないで必要なインデックスエントリを検索するインデックスの機能です。

高度: データベースサーバでのインデックスの別の使用方法 [48 ページ]
データベースサーバでは、パフォーマンスを向上させる目的でインデックスを使用します。

1.1.9.1 複合インデックス

複数のカラムに対するインデックスは、複合インデックスと呼ばれます。

たとえば、次の文では 2 カラムの複合インデックスが作成されます。

```
CREATE INDEX name
ON Employees ( Surname, GivenName );
```

複合インデックスは、最初のカラムだけでは高い選択性が得られない場合に役立ちます。たとえば、Surname と GivenName に対する複合インデックスは、従業員の姓が同じ場合に便利です。各従業員はユニークな ID を持っており、カラム Surname は追加の選択性を提供しないため、EmployeeID と Surname の複合インデックスは役に立ちません。

インデックスにカラムを追加すると検索対象を限定できますが、2 カラムのインデックスを使用することと 2 つの別個のインデックスを使用することは異なります。複合インデックスは、電話帳でまず姓が並べられ、次に同じ姓の中で名前順に並べられるのとよく似た構造を持っています。電話帳は姓を知っていれば役に立ちますし、姓と名前の両方を知っていればなお役に立ちます。しかし、名前だけを知っていても役には立ちません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

カラムの順序

複合インデックスを作成する場合は、カラムの順序を慎重に検討してください。複合インデックスは、インデックスのすべてのカラムまたは最初のカラムだけを検索する場合に役立ちます。2 番目以降のカラムだけを検索する場合には役立ちません。

1 つのカラムだけを何度も検索する場合は、そのカラムを複合インデックスの最初のカラムにしてください。2 カラムインデックスの両方のカラムを個別に検索する場合は、第 2 のカラムだけで構成される 2 番目のインデックスを作成することを検討します。

たとえば、2 つのカラムに複合インデックスを作成するとします。1 つのカラムには従業員の名前が格納され、もう 1 つには従業員の姓が格納されます。名前、姓の順に格納するインデックスを作成できます。または、姓、名前の順にインデックスを付けることもできます。この 2 つのインデックスは両方のカラムの情報を編成するものですが、その機能は異なります。

```
CREATE INDEX IX_GivenName_Surname
   ON Employees ( GivenName, Surname );
CREATE INDEX IX_Surname_GivenName
   ON Employees ( Surname, GivenName );
```

次に、名前 John を検索するとします。使用できる唯一のインデックスは、インデックスの最初のカラムに名前を格納しているインデックスです。姓、名前の順に編成されているインデックスは使用できません。これは、John という名前の人がインデックスのどこに現れるかわからないためです。

名前だけ、または姓だけで人を検索する場合、これらのインデックスの両方を作成することを検討してください。

または、それぞれが 1 つのカラムだけを含むインデックスを 2 つ作成する方法もあります。ただし、データベースサーバは、1 つのクエリを処理するとき、1 つのテーブルにアクセスするために 1 つのインデックスしか使用しません。両方の名前がわかっていても、データベースサーバは正しい姓を持つローを検索するため、追加のローを読み込む必要があります。

前述の例のように、CREATE INDEX 文を使用してインデックスを作成した場合、カラムは文で指定した順序で表示されます。

複合インデックスと ORDER BY

デフォルトでは、インデックスのカラムは昇順でソートされますが、オプションで、CREATE INDEX 文で DESC を指定すると降順でソートできます。

ORDER BY 句にインデックスに含まれるカラムだけが指定されている限り、データベースサーバは、そのインデックスを使用して ORDER BY クエリを最適化するように選択できます。また、インデックスのカラムは、ORDER BY 句と完全に同じ、または正反対の順序になります。1 カラムインデックスの場合、順序付けは常に最適化できますが、複合インデックスには多少の考慮が必要です。次の表に、2 カラムインデックスで可能な操作を示します。

インデックスカラム	最適化可能な ORDER BY クエリ	最適化できない ORDER BY クエリ
ASC、ASC	ASC、ASC または DESC、DESC	ASC、DESC または DESC、ASC
ASC、DESC	ASC、DESC または DESC、ASC	ASC、ASC または DESC、DESC
DESC、ASC	DESC、ASC または ASC、DESC	ASC、ASC または DESC、DESC
DESC、DESC	DESC、DESC または ASC、ASC	ASC、DESC または DESC、ASC

3 つ以上のカラムを持つインデックスには、上記と同じ規則が適用されます。たとえば、次のインデックスがあるとします。

```
CREATE INDEX idx_example
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ON table1 ( col1 ASC, col2 DESC, col3 ASC );
```

この場合は、次に示すクエリを最適化できます。

```
SELECT col1, col2, col3 FROM table1
ORDER BY col1 ASC, col2 DESC, col3 ASC;
```

```
SELECT col1, col2, col3 FROM example
ORDER BY col1 DESC, col2 ASC, col3 DESC;
```

ORDER BY 句に ASC と DESC の他のパターンを持つクエリの最適化には、このインデックスは使用されません。たとえば、次の文は最適化されません。

```
SELECT col1, col2, col3 FROM table1
ORDER BY col1 ASC, col2 ASC, col3 ASC;
```

1.1.9.2 クラスタドインデックス

インデックスのクラスタ化を宣言することで、大規模なインデックススキャンのパフォーマンスを改善することができます。

クラスタドインデックスを使用すると、連続したインデックスエントリにおける 2 つのローがデータベース内の同じページに現れる確率が高くなります。この戦略では、テーブルページをバッファプールに読み込む回数が減り、パフォーマンスがさらに向上します。

クラスタ化プロパティを持つインデックスが存在すると、データベースサーバはテーブルのローをクラスタドインデックスに出現する場合とほぼ同じ順序で格納しようとします。ただし、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、クラスタ化は概算であり、完全なクラスタは保証されません。このため、データベースサーバはテーブルを順次スキャンできず、クラスタドインデックスキーのシーケンスですべてのローが取得されるわけではありません。テーブルのローがソートされた順序で返されるようにするには、インデックスを使用してローにアクセスするアクセスプランか、物理ソートを実行するアクセスプランが必要です。

オプティマイザはクラスタ化プロパティを持つインデックスを利用します。これは、一致または隣接するインデックスキー値を持つテーブルローについて、オプティマイザが物理的な隣接性の予測を考慮に入れてインデックス取得コストの予測を修正することによって行われます。

多くのローが挿入または更新されていくため、テーブルのクラスタ化の程度は時間とともに低下することがあります。データベースサーバは、ISYSPHYSIDX システムテーブルのクラスタドインデックスごとにクラスタ化の程度を自動的に追跡します。テーブルのローで非クラスタ化が大幅に進行したことをデータベースサーバが検出すると、オプティマイザは予測したインデックス取得コストを調整します。

テーブルのいずれかのインデックスをクラスタ化することを決定する際は、予測されるクエリの負荷を考慮してください。通常は実験が必要になります。一般的に、指定されたクエリに次のような状態が起こる場合には、データベースサーバはクラスタドインデックスを使用してパフォーマンスを向上させることができます。

- クエリの応答に必要なテーブルページの多くが、メモリ内にまだ存在していません。テーブルページがすでにメモリ内に存在する場合、サーバはこれらのページを読み込む必要がないため、クラスタリングは影響しません。
- 非自明な数のローが返されると予想されるインデックス検索を実行し、クエリが応答できます。たとえば、通常、クラスタリングは単純なプライマリキーの検索には影響しません。
- インデックス専用取得の実行とは対照的に、データベースサーバは実際にテーブルページを読み込む必要があります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

クラスタドインデックスの宣言

インデックスのクラスタ化プロパティは、SQL 文を使用していつでも追加または削除できます。あらゆるプライマリキーインデックス、外部キーインデックス、一意性制約インデックス、セカンダリインデックスは、CLUSTERED プロパティを使用して宣言できます。ただし、宣言できるクラスタドインデックスはテーブルあたり多くても 1 つです。これは、次のいずれかの文で行います。

- CREATE TABLE 文
- ALTER DATABASE 文
- CREATE INDEX 文
- DECLARE LOCAL TEMPORARY TABLE 文

複数の文を組み合わせて使用すると、クラスタ化の効果の維持やリストアができます。

- UNLOAD TABLE 文を使用すると、クラスタドインデックスキーの順序でテーブルをアンロードできます。
- LOAD TABLE 文は、クラスタドインデックスキーの順序でローをテーブルに挿入します。
- INSERT 文は、新しいローを挿入するときに、クラスタドインデックスキーの順序が隣接するローと同じテーブルページに挿入しようとします。
- REORGANIZE TABLE 文はクラスタドインデックスに従ってローを再編成することによって、テーブルのクラスタ化をリストアします。クラスタ化を指定していないテーブルで REORGANIZE TABLE 文を使用すると、プライマリキーを使用してテーブルが並べ替えられます。

インデックス作成ウィザードを使用して、SQL Central でクラスタドインデックスを作成することもできます。メッセージが表示されたら、クラスタドインデックスを作成するをクリックします。

関連情報

インデックスの作成 [40 ページ]

1.1.9.3 インデックスの作成

ベーステーブルのインデックス、テンポラリテーブル、マテリアライズドビューを作成します。

前提条件

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。

- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限
- そのテーブルに対する REFERENCES 権限に加え、COMMENT ANY OBJECT システム権限、ALTER ANY INDEX システム権限、または ALTER ANY OBJECT システム権限

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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

通常のインデックスは作成できません。無効になっているマテリアライズドビューのインデックスは作成できません。

コンテキスト

計算カラムを使用して、組み込み関数のインデックスを作成することもできます。

インデックスを作成するときは、カラムを指定する順序は、インデックスでカラムが出現する順序になります。インデックス定義でカラム名を重複参照することはできません。データベースに適切なインデックスの選択を手助けをするインデックスコンサルタントも使用できます。

auto_commit_on_create_local_temp_index オプションを ON に設定した場合、ローカルテンポラリテーブルでインデックスを作成すると、オートコミットが実行されます。このオプションのデフォルト設定は Off です。

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)。

手順

1. SQL Central で、*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. 左ウィンドウ枠でインデックスを右クリックし、 新規 > インデックス をクリックします。
3. インデックス作成ウィザードの指示に従います。

結果

新しいインデックスがテーブルの *[インデックス]* タブと *[インデックス]* に表示されます。新しいインデックスは、クエリで使用可能です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.9.4 インデックスの検証

インデックスで参照されているすべてのローが、実際にテーブルに存在するかどうか検証します。

前提条件

インデックスの所有者であるか、VALIDATE ANY OBJECT システム権限を持っている必要があります。

接続によるデータベースの変更がない場合のみ、検証を実行します。

コンテキスト

外部キーインデックスの場合は、対応するローがプライマリテーブルにあることも確認します。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、インデックスをダブルクリックします。
3. インデックスを右クリックして、*[検証]* をクリックします。
4. *OK* をクリックします。

結果

インデックスで参照されているすべてのローが、実際にテーブルに存在するかどうか検証するため、チェックが実行されます。外部キーインデックスの場合は、対応するローがプライマリテーブルにあることをチェックします。氏

1.1.9.5 インデックスの再構築

テーブルまたはマテリアライズドビューでの広範囲の挿入と削除によりフラグメント化されたインデックスを再構築します。

前提条件

テーブルのインデックスを再構築するには、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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 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再構築するには、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

コンテキスト

インデックスを再構築するときは、物理インデックスを再構築します。物理インデックスを使用するすべての論理インデックスは、再構築操作により恩恵を受けます。論理インデックスで再構築を実行する必要はありません。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、インデックスをダブルクリックします。
3. インデックスを右クリックして、*[再構築]* をクリックします。
4. *OK* をクリックします。

結果

断片化は削除され、インデックスが再構築されます。

関連情報

高度: 論理インデックスと物理インデックス [46 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.9.6 インデックスの削除

インデックスが必要なくなった場合、または、プライマリキーまたは外部キーの一部であるカラムの定義を変更する必要がある場合、インデックスを削除します。

前提条件

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

外部キー、プライマリキー、または一意性制約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 ALTER 権限
- ALTER ANY TABLE 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

手順

1. SQL Central で、*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. 左ウィンドウ枠で、インデックスをダブルクリックします。
3. インデックスを右クリックして、*[削除]* をクリックします。
4. はいをクリックします。

結果

インデックスはデータベースから削除されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

プライマリキーまたは外部キーの一部であるカラムの定義を削除または変更するため、インデックスを削除しなければならない場合合、新しいインデックスを追加する必要があります。

関連情報

インデックスの作成 [40 ページ]

1.1.9.7 高度: カタログ内のインデックス情報

カタログにはデータベースのインデックスに関する情報を提供する複数のシステムテーブルがあります。

ISYSIDX システムテーブルは、プライマリキーインデックスや外部キーインデックスを含む、データベース内にあるすべてのインデックスのリストを提供します。インデックスの補足情報は、ISYSPHYSIDX、ISYSIDXCOL、ISYSFKEY の各システムテーブルにあります。SQL Central または Interactive SQL を使用すると、これらのテーブルのシステムビューをブラウズして、それに含まれるデータを確認できます。

次に、インデックス情報がシステムテーブルに格納される仕組みの概要について説明します。

ISYSIDX システムテーブル

インデックスを追跡するための中央テーブルです。ISYSIDX システムテーブルの各ローは、データベース内の論理インデックス (PKEY、FKEY、一意性制約、セカンダリインデックス) を定義します。

ISYSPHYSIDX システムテーブル

ISYSPHYSIDX システムテーブルの各ローは、データベース内の物理インデックスを定義します。

ISYSIDXCOL システムテーブル

SYSIDX システムビューの各ローがデータベース内のインデックス 1 つを示すように、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。

ISYSFKEY システムテーブル

データベース内の各外部キーは、ISYSFKEY システムテーブル内のロー 1 つと、ISYSIDX システムテーブル内のロー 1 つによって定義されます。

関連情報

高度: 論理インデックスと物理インデックス [46 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.9.8 高度: 論理インデックスと物理インデックス

ソフトウェアでは論理インデックスと物理インデックスがサポートされています。

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。プライマリキー、セカンダリキー、外部キー、一意性制約を作成するときに、データベースサーバは、制約の論理インデックスを作成することで参照整合性を確保します。次に、データベースサーバは制約を満たすインデックスがすでに存在するかどうかを確認します。条件を満たす物理インデックスがすでに存在する場合、データベースサーバはその物理インデックスへの論理インデックスを指します。そのような物理インデックスが存在しない場合、データベースサーバは新しい物理インデックスを作成してから、その物理インデックスへの論理インデックスを指します。

物理インデックスが論理インデックスの要件を満たすには、カラムとカラムの順序、および各カラムのデータの順序 (昇順や降順) が同一である必要があります。

データベース内のすべての論理インデックスと物理インデックスの情報は、それぞれ ISYSIDX システムテーブルと ISYSPHYSIDX システムテーブルに記録されます。論理インデックスを作成すると、そのインデックス定義を保持するためにエントリが ISYSIDX システムテーブルに作成されます。論理インデックスを満たすために使用される物理インデックスへの参照は、ISYSIDX.phys_id カラムに記録されます。物理インデックスは ISYSPHYSIDX システムテーブルに定義されます。

複数の論理インデックスは単一の物理インデックスを指すことができるため、論理インデックスを使用するということは、データベースサーバは重複した物理インデックスを作成して管理する必要がないということを意味します。

論理インデックスを削除すると、その定義が ISYSIDX システムテーブルから削除されます。特定の物理インデックスを参照するだけの論理インデックスの場合は、その物理インデックスと、ISYSPHYIDX システムテーブル内で対応するエントリも削除されます。

リモートテーブルに対して物理インデックスは作成されません。テンポラリテーブルの場合、物理インデックスは作成されますが、ISYSPHYSIDX に記録されず、使用後に廃棄されます。また、テンポラリテーブルの物理インデックスは共有されません。

このセクションの内容:

物理インデックスを共有する論理インデックスの特定 [46 ページ]
複数の論理インデックスで、物理インデックスを共有することができます。

1.1.9.8.1 物理インデックスを共有する論理インデックスの特定

複数の論理インデックスで、物理インデックスを共有することができます。

インデックスの削除を行うと、物理インデックスを使用する論理インデックスを削除することになります。物理インデックスを使用している論理インデックスが 1 つのみの場合、物理インデックスも削除されます。別の論理インデックスが同じ物理インデックスを共有している場合、その物理インデックスは削除されません。インデックスを削除することでディスク領域が解放されることを期待していたり、物理的に再作成するためにインデックスを削除しようとしたりする場合は特に注意する必要があります。

テーブルのインデックスが他のインデックスと物理インデックスを共有しているか判断するには、SQL Central でテーブルを選択し、インデックスタブをクリックします。インデックスの物理 ID 値が、リスト内の別のインデックスにも表示されているかどうかに注意してください。物理 ID の値が一致するということは、それらのインデックスが同じ物理インデックスを共有しているということです。物理インデックスを再作成する場合は、ALTER INDEX...REBUILD 文を使用できます。また、すべてのインデックスを削除してから再作成する方法もあります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

物理インデックスが共有されているテーブルの特定

次のようなクエリを実行することで、物理インデックスが共有されているすべてのテーブルのリストをいつでも取得できます。

```
SELECT tab.table_name, idx.table_id, phys.phys_index_id, COUNT(*)
 FROM SYSIDX idx JOIN SYSTAB tab ON (idx.table_id = tab.table_id)
 JOIN SYSPHYSIDX phys ON ( idx.phys_index_id  = phys.phys_index_id
   AND idx.table_id = phys.table_id )
 GROUP BY tab.table_name, idx.table_id, phys.phys_index_id
   HAVING COUNT(*) > 1
ORDER BY tab.table_name;
```

このクエリの結果セットの例を次に示します。

table_name	table_id	phys_index_id	COUNT()
ISYSCHECK	57	0	2
ISYSCOLSTAT	50	0	2
ISYSFKEY	6	0	2
ISYSSOURCE	58	0	2
MAINLIST	94	0	3
MAINLIST	94	1	2

各テーブルのローの数は、テーブルの共有物理インデックスの数を示します。この例では、すべてのテーブルに共有物理インデックスが 1 つありますが、架空のテーブル MAINLIST には 2 つあります。phys_index_id 値は、共有されている物理インデックスを表し、COUNT カラムの値は、物理インデックスを共有している論理インデックスの数を表します。

当該テーブルで物理インデックスを共有しているインデックスを確認するには、SQL Central を使用する方法もあります。このことを行うには、左ウィンドウ枠でテーブルを選択し、右ウィンドウ枠でインデックスタブをクリックし、次に、物理 ID カラムの値が同じであるローが複数あるかどうかを確認します。物理 ID の値が同じインデックスは、同じ物理インデックスを共有しています。

関連情報

インデックスの再構築 [42 ページ]

1.1.9.9 高度: インデックスの選択性とインデックスファンアウト

インデックスの選択性とは、追加データを読み込まないで必要なインデックスエントリを検索するインデックスの機能です。

選択性が低い場合は、インデックスが参照するテーブルページから追加情報を取り出します。このような取出しは完全比較と呼ばれ、インデックスのパフォーマンスを低下させます。

FullCompare プロパティは、発生した完全比較の数を追跡します。また、この統計は Windows パフォーマンスモニタを使用してモニタできます。

さらに、完全比較の数は、統計情報付きのグラフィカルなプランの形式で提供されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

インデックスは、ツリーのようにいくつかのレベルで編成されています。インデックスの最初のページはルートページと呼ばれ、その次の下位レベルの 1 つ以上のページへと分岐して、さらにそれが分岐して、インデックスの最下位レベルに達します。最下位レベルのインデックスページは、リーフページと呼ばれます。n レベルを持つインデックスの場合、特定のローを探すにはインデックスページを n 回読み込む必要があり、実際のローを含むデータページを 1 回読み込む必要があります。通常、使用頻度の高いインデックスページはキャッシュに格納される傾向があるため、ディスクからの読み込みは n 回よりも少なくて済みます。

インデックスファンアウトは、1 ページに格納されるインデックスエントリの数です。ファンアウトが大きなインデックスは、ファンアウトが小さなインデックスよりレベル数が少なくなります。したがって、通常はインデックスファンアウトが大きいほど、インデックスのパフォーマンスが向上します。データベースに適切なページサイズを選択すると、インデックスファンアウトを向上できます。

インデックスレベル数を確認するには、sa_index_levels システムプロシージャを使用します。

関連情報

実行プランのコンポーネント [230 ページ]

1.1.9.10 高度: データベースサーバでのインデックスの別の使用方法

データベースサーバでは、パフォーマンスを向上させる目的でインデックスを使用します。

インデックスを使用すると、データベースサーバはカラムの一意性を強化し、ロックするローとページの数を減らし、述部の選択性を適切に推定できます。

カラム一意性の強化

インデックスがないと、データベースサーバは、値が挿入されるたびにテーブル全体をスキャンし、その値がユニークであることを確認する必要があります。このため、データベースサーバは一意性制約付きで各カラムのインデックスを自動的に作成します。

ロックの減少

インデックスによって、挿入、更新、削除中にロックされるローとページの数が減少します。これは、インデックスがテーブルに適用する順序付けによるものです。

選択性の推定

インデックスは順序付けされているため、オプティマイザはインデックスの上位レベルをスキャンすることで、特定のクエリを満たす値のパーセンテージを推定できます。このアクションは、部分インデックススキャンと呼ばれます。

関連情報

ロックの仕組み [782 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.10 ビュー

ビューとは、ビュー定義の結果セットによって定義される計算テーブルです。ビュー定義は SQL クエリとして表現されます。

ビューを使用して、制御できるフォーマットでデータベースのユーザに正確な情報を表示できます。次の 2 種類のビューがサポートされています。通常のビューとマテリアライズドビューの 2 つです。

データベース内の各ビューの定義は、SYSVIEW システムビューで利用できます。

このセクションの内容:

通常のビュー、マテリアライズドビュー、およびテーブルの機能 [49 ページ]
通常のビューとマテリアライズドビューには、特にテーブルと比較して異なる機能があります。

ビューを使用する利点 [50 ページ]
ビューを使用すると、複数の方法でデータベース中のデータへのアクセスを調整できます。

ビューの依存性 [50 ページ]
ビュー定義は、カラム、テーブル、他のビューなど、その他のオブジェクトを参照し、これらの参照によってビューは参照先のオブジェクトに依存しています。

通常のビュー [53 ページ]
ビューを作成するクエリには名前が与えられ、データベースのシステムテーブルに定義が格納されます。

1.1.10.1 通常のビュー、マテリアライズドビュー、およびテーブルの機能

通常のビューとマテリアライズドビューには、特にテーブルと比較して異なる機能があります。

許可内容	通常のビュー	マテリアライズドビュー	テーブル
アクセス権限	○	○	○
SELECT	○	○	○
UPDATE	一部	×	○
INSERT	一部	×	○
DELETE	一部	×	○
従属ビュー	○	○	○
インデックス	×	○	○
整合性制約	×	×	○
キー	×	×	○

ビューの表記規則

通常のビューとは、ユーザがビューを参照するたびに再計算され、結果セットがディスクに格納されないビューのことです。これは最も一般的に使用される種類のビューです。マニュアルのほとんどの部分は、通常のビューを指しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

マテリアライズドビューとは、結果セットが事前計算され、ベーステーブルの内容と類似する、ディスク上で実体化されたビューを指します。

マニュアル内でのビュー (そのもの) の意味は、文脈に基づきます。通常のビューとマテリアライズドビューの共通の事項に関するセクションで使用される場合は、通常のビューとマテリアライズドビューの両方を指します。用語がマテリアライズドビューのマニュアルで使用されている場合はマテリアライズドビューを指し、通常のビューのマニュアルの場合は通常のビューを指します。

1.1.10.2 ビューを使用する利点

ビューを使用すると、複数の方法でデータベース中のデータへのアクセスを調整できます。

効率的なリソース使用

通常のビューでは、データを格納するために追加の記憶領域は必要ありません。呼び出しごとに再計算されます。マテリアライズドビューはディスク領域を必要としますが、呼び出しごとに再計算する必要はありません。データベースのサイズが大きく、同じテーブルに対して頻繁に繰り返し発生するジョイン要求をデータベースサーバが処理するような環境では、マテリアライズドビューを使用すると、応答時間が向上します。

セキュリティの向上

関連データへのアクセスだけが許可されます。

利便性の向上

ベーステーブルよりも見やすい形でユーザとアプリケーション開発者にデータを提供します。

一貫性の向上

よく使われるクエリの定義をデータベース内で集中管理します。

1.1.10.3 ビューの依存性

ビュー定義は、カラム、テーブル、他のビューなど、その他のオブジェクトを参照し、これらの参照によってビューは参照先のオブジェクトに依存しています。

あるビューの参照先オブジェクトのセットには、そのビューが直接的または間接的に参照するすべてのオブジェクトが含まれます。たとえば、ビューはあるテーブルを参照する別のビューを参照することにより、間接的にそのテーブルを参照できます。

次のテーブルとビューのセットを考えてみます。

```
CREATE TABLE t1 ( c1 INT, c2 INT );
CREATE TABLE t2( c3 INT, c4 INT );
CREATE VIEW v1 AS SELECT * FROM t1;
CREATE VIEW v2 AS SELECT c3 FROM t2;
CREATE VIEW v3 AS SELECT c1, c3 FROM v1, v2;
```

次のビューの依存性は、上記の定義から特定できます。

- ビュー v1 は t1 の各カラムと t1 自体に依存しています。
- ビュー v2 は、t2.c3 と t2 自体に依存しています。
- ビュー v3 はカラム t1.c1 と t2.c3、テーブル t1 と t2、ビュー v1 と v2 に依存しています。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベースサーバは指定されたビューが参照するカラム、テーブル、ビューを追跡します。データベースサーバではこの依存性情報を使用して、参照先オブジェクトのスキーマ変更が使用できない状態で参照元ビューに残らないようにします。

このセクションの内容:

依存性とスキーマ変更 [51 ページ]
テーブルやビューに定義したスキーマを変更しようとする場合、従属ビューに対して変更による影響があるかどうかをデータベースサーバが検討する必要があります。

依存性情報の取得 (SQL の場合) [52 ページ]
データベース内の任意のテーブルまたはビューに依存しているオブジェクトのリストを取得します。

1.1.10.3.1 依存性とスキーマ変更

テーブルやビューに定義したスキーマを変更しようとする場合、従属ビューに対して変更による影響があるかどうかをデータベースサーバが検討する必要があります。

スキーマ変更操作の例を次に示します。

- テーブル、ビュー、マテリアライズドビュー、またはカラムの削除
- テーブル、ビュー、マテリアライズドビュー、またはカラムの名前変更
- カラムの追加、削除、または変更
- カラムのデータ型、サイズ、または NULL 入力属性の変更
- ビュー、またはテーブルのビューの依存性の無効化

スキーマ変更操作中に発生するイベント

1. データベースサーバは、変更するテーブルやビューに直接的または間接的に依存するビューのリストを生成します。DISABLED ステータスのビューは無視されます。
   いずれかの従属ビューがマテリアライズドビューである場合、要求は失敗し、エラーが返され、残りのイベントは発生しません。従属したマテリアライズドビューを明示的に無効にしてから、スキーマ変更操作を続行してください。
2. データベースサーバは、変更するオブジェクトとすべての従属した通常のビューに対して、排他スキーマロックを取得します。
3. データベースサーバはすべての従属した通常のビューのステータスを INVALID に設定します。
4. データベースサーバはスキーマ変更操作を実行します。操作が失敗すると、ロックは解放され、従属した通常のビューのステータスは VALID にリセットされます。さらにエラーが返され、残りの手順は発生しません。
5. データベースサーバは従属した通常のビューを再コンパイルし、成功した場合は各ビューのステータスを VALID に設定します。いずれかの通常のビューでコンパイルが失敗した場合も、そのビューのステータスは INVALID のままです。INVALID である通常のビューに対する以後の要求により、データベースサーバはビューを再コンパイルしようとします。それらの試行に失敗した場合は、INVALID ビューまたはそのビューが依存するオブジェクトを変更する必要があります。

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

通常のビュー: 依存性とスキーマ変更

- 通常のビューは、マテリアライズドビューを含め、テーブルやビューを参照できます。
- テーブルまたはビューのスキーマを変更すると、データベースはすべての参照元の通常ビューを自動的に再コンパイルしようとします。
- ビューまたはテーブルを無効にするか削除すると、すべての従属した通常のビューが自動的に無効になります。
- ALTER TABLE 文の DISABLE VIEW DEPENDENCIES 句を使用して、従属した通常のビューを無効にできます。

マテリアライズドビュー: 依存性とスキーマ変更

- マテリアライズドビューは、ベーステーブルだけを参照します。
- 有効なマテリアライズドビューが参照している場合は、ベーステーブルに対するスキーマ変更は許可されません。テーブルに外部キーを追加することはできます (ALTER TABLE、ADD FOREIGN KEY など)。
- テーブルを削除する前に、すべての従属マテリアライズドビューを無効にするか削除する必要があります。
- ALTER TABLE 文の DISABLE VIEW DEPENDENCIES 句は、マテリアライズドビューには影響しません。マテリアライズドビューを無効にするには、ALTER MATERIALIZED VIEW...DISABLE 文を使用する必要があります。
- マテリアライズドビューを無効にした場合は、ALTER MATERIALIZED VIEW...ENABLE 文を実行するなどして、再度明示的に有効にする必要があります。

関連情報

マテリアライズドビューの有効化または無効化 [75 ページ]

1.1.10.3.2 依存性情報の取得 (SQL の場合)

データベース内の任意のテーブルまたはビューに依存しているオブジェクトのリストを取得します。

前提条件

この作業の実行に権限は必要なく、PUBLIC でカタログにアクセスできることが想定されています。

コンテキスト

SYSDEPENDENCY システムビューは依存性情報を格納します。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。直接依存性とは、あるオブジェクトがその定義内で別のオブジェクトを直接参

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

照していることです。データベースサーバは直接依存性情報を使用して、間接依存性を判断します。たとえば、ビュー A はビュー B を参照し、ビュー B はテーブル C を参照しているとします。この場合、ビュー A はビュー B に直接依存していて、テーブル C に間接依存しています。

テーブルまたはビューを変更し、影響を受ける可能性のある他のオブジェクトを把握する必要がある場合、このタスクは便利です。

手順

1. データベースに接続します。
2. sa_dependent_views システムプロシージャを呼び出す文を実行します。

結果

従属ビューの ID リストが返されます。

例

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;
```

1.1.10.4 通常のビュー

ビューを作成するクエリには名前が与えられ、データベースのシステムテーブルに定義が格納されます。

通常のビューの作成時に、データベースサーバはビュー定義をデータベースに格納します。ただし、ビューのデータは格納されません。ビュー定義は、そのビュー定義が参照される場合で、かつそのビューの使用中のみ実行されます。ビューの作成時は、データベースに重複したデータを格納する必要がありません。

各部署の従業員数をリストする必要が頻繁にあるとします。このリストは次の文で取得できます。

```
SELECT DepartmentID, COUNT(*)
FROM Employees
GROUP BY DepartmentID;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

通常のビューの **SELECT** 文に対する制限

通常のビューとして使用できる SELECT 文には制限があります。特に、SELECT クエリ中では ORDER BY 句を使用できません。リレーショナルテーブルでは、ローやカラムの並び順には意味がありませんが、ORDER BY 句を使用すると、ビューのローの順序が規定されるからです。GROUP BY 句、サブクエリ、ジョインは、ビューの定義で使用できます。

ビューを作成するには、必要とする正確な結果が必要なフォーマットで得られるまで SELECT クエリを編集します。SELECT 文が作成できたら、先頭に次の句を追加するとビューが完成します。

```
CREATE VIEW view-name AS query;
```

通常のビューを更新する文

ビューを定義するクエリ指定が更新可能な場合、UPDATE 文、INSERT 文、DELETE 文を使用してビューを更新できます。ビューを定義するクエリ指定に次のいずれかが含まれている場合、そのビューは派生の関係で更新不可能となります。

- UNION、EXCEPT、または INTERSECT
- DISTINCT 句
- GROUP BY 句
- WINDOW 句
- FIRST、TOP、または LIMIT 句
- 集合関数
- FROM 句に複数のテーブル (ansi_update_constraints オプションが Strict または Cursors に設定されている場合)。
- ORDER BY 句 (ansi_update_constraints オプションが Strict または Cursors に設定されている場合)。
- すべての SELECT リスト項目がベーステーブルのカラムではありません。

WITH CHECK OPTION 句

ビューを作成する場合、WITH CHECK OPTION 句は、ビューを介したベーステーブルに対する挿入時または更新時に変更されるデータの制御に役立ちます。次の例で説明します。

WITH CHECK OPTION 句を使用して次の文を実行し、SalesEmployees ビューを作成します。

```
CREATE VIEW SalesEmployees AS
   SELECT EmployeeID, GivenName, Surname, DepartmentID
   FROM Employees
   WHERE DepartmentID = 200
   WITH CHECK OPTION;
```

選択すると、このビューの内容が次のように表示されます。

```
SELECT * FROM SalesEmployees;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

EmployeeID	GivenName	Surname	DepartmentID
129	Philip	Chin	200
195	Marc	Dill	200
299	Rollin	Overbey	200
467	James	Klobucher	200
...	...	...	...

次に、Philip Chin の DepartmentID を 400 に更新しようとします。

```
UPDATE SalesEmployees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

WITH CHECK OPTION が指定されたため、データベースサーバは、この更新によってビュー定義 (この場合は WHERE 句の式) で違反が発生するかどうかを評価します。DepartmentID は 200 でなければならないため、この文は失敗し、データベースサーバが "ベーステーブル 'Employees' の挿入/更新に対して WITH CHECK OPTION が違反しています" というエラーを返します。

ビュー定義で WITH CHECK OPTION を指定しなかった場合は更新操作が実行され、Employees テーブルが新しい値で修正されてしまい、これ以降 Philip Chin がビューから消えてしまいます。

SalesEmployees ビューを参照するビュー (たとえば View2) が作成された場合は、View2 自体に WITH CHECK OPTION 句がなくても、SalesEmployees ビューの WITH CHECK OPTION の基準に合わなければ、View2 に対する更新や挿入は拒否されます。

このセクションの内容:

通常のビューのステータス [56 ページ]
通常のビューには、ステータスが関連付けられています。

通常のビューの作成 [57 ページ]
1 つ以上のソースからデータを結合するビューを作成します。

通常のビューの変更 [58 ページ]
データベースの定義を編集することで、通常のビューを変更します。

通常のビューの削除 [59 ページ]
ビューが必要なくなったら、削除します。

通常のビューの無効化または有効化 (SQL Central の場合) [60 ページ]
通常のビューを有効または無効にすることで、データベースサーバがそのビューを使用できるかどうかを制御します。

通常のビューの無効化または有効化 (SQL の場合) [61 ページ]
通常のビューを有効または無効にすることで、データベースサーバがそのビューを使用できるかどうかを制御します。

通常のビュー内のデータのブラウズ [63 ページ]
通常のビュー内のデータをブラウズします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

クエリ結果の要約、グループ化、ソート [355 ページ]
マテリアライズドビュー [64 ページ]

1.1.10.4.1 通常のビューのステータス

通常のビューには、ステータスが関連付けられています。

ステータスは、データベースサーバが使用するビューの利用可能性を反映しています。

すべてのビューのステータスを表示するには、SQL Central の左ウィンドウ枠でビューをクリックし、右ウィンドウ枠でステータスカラムの値を検査します。また、単一のビューのステータスを表示するには、SQL Central でビューを右クリックし、プロパティをクリックしてステータスの値を検査します。

通常のビューのステータスの種類について、次に説明します。

VALID

ビューは有効で、その定義と一貫性があることが保証されています。データベースサーバは、追加の作業なくこのビューを利用できます。有効にされたビューのステータスは VALID です。

SYSOBJECT システムビューで、値 1 はステータス VALID を表します。

INVALID

INVALID ステータスは、参照先オブジェクトのスキーマ変更後に発生し、変更したためにビューを有効にしようとして失敗したことを表します。たとえば、ビュー v1 がテーブル t 内のカラム c1 を参照するとします。t を変更して c1 を削除する場合、カラムを削除する ALTER 操作の一環としてデータベースサーバがビューを再コンパイルすると、v1 のステータスは INVALID に設定されます。このとき、v1 は c1 が t に追加された場合だけ再コンパイルできます。再コンパイルしない場合、v1 は c1 を参照しないように変更されます。ビューが参照するテーブルやビューを削除した場合も、そのビューは INVALID になります。

INVALID ビューは DISABLED ビューと異なり、クエリなどで INVALID ビューが参照されるたびに、データベースサーバはそのビューを再コンパイルしようとします。コンパイルに成功すると、クエリが処理されます。ビューを明示的に有効にしないかぎり、ステータスは INVALID のままです。失敗した場合は、エラーが返されます。

データベースサーバは、INVALID ビューを内部的に有効にすると、パフォーマンス警告を発行します。

SYSOBJECT システムビューで、値 2 はステータス INVALID を表します。

DISABLED

無効にされたビューは、データベースサーバがクエリに応答するために使用できません。無効にされたビューを使用しようとするクエリは、エラーを返します。

通常のビューは、次の場合にこのステータスになります。

- ビューを明示的に無効にした場合。ALTER VIEW...DISABLE 文を実行した場合など。
- そのビューが依存するビュー (マテリアライズドビューまたは非マテリアライズドビュー) を無効にした場合。
- テーブルのビューの依存性を無効にした場合。ALTER TABLE...DISABLE VIEW DEPENDENCIES 文を実行した場合など。

SYSOBJECT システムビューで、値 4 はステータス DISABLED を表します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

通常のビューの無効化または有効化 (SQL の場合) [61 ページ]

1.1.10.4.2 通常のビューの作成

1 つ以上のソースからデータを結合するビューを作成します。

前提条件

ビューによりパフォーマンスが向上し、ユーザが問い合わせできるデータを管理できます。

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。

手順

1. SQL Central で、*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. 左ウィンドウ枠で ビュー を右クリックし、 新規 > ビュー をクリックします。
3. ビュー作成ウィザードの指示に従います。
4. 右ウィンドウ枠で、*[SQL]* タブをクリックし、定義のコードを編集します。変更を保存するには、 ファイル > 保存 をクリックします。

結果

作成したビューの定義がデータベースに追加されます。クエリがビューを参照するたびに、定義を使用してビューにデータを取り込み、結果を返します。

次のステップ

ビューに問い合わせて結果を検証し、正しいデータが返されたか確認します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.10.4.3 通常のビューの変更

データベースの定義を編集することで、通常のビューを変更します。

前提条件

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

コンテキスト

ビューに追加のテーブルのデータを含める場合、ビュー定義を更新して、テーブルデータをビュー定義の既存のデータソースにジョインさせます。

ビュー定義が古い場合 (基本となるデータのスキーマ変更によりコンパイルしない)、カラムを追加または削除したり、設定に関連した変更を実行したりする必要があります。

既存のビューの名前を変更することはできません。代わりに、新しい名前を付けて新しくビューを作成し、以前の定義をそこにコピーしてから、元のビューを削除します。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. ビューを選択します。
4. 右ウィンドウ枠で、*SQL* タブをクリックし、定義のコードを編集します。

   ヒント

   複数のビューを編集する場合は、各ビューを右ウィンドウ枠の *SQL* タブで編集するより、各ビューに対して別のウィンドウを開く方が便利な場合があります。別のウィンドウを開くには、ビューを選択し、ファイル > 新しいウィンドウで編集 をクリックします。

5. ファイル > 保存 をクリックします。

結果

データベース内のビューの定義が更新されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

ビューに問い合わせて結果を検証し、正しいデータが返されたか確認します。

通常のビューを変更する場合、ビューに他のビューの依存性が存在するときは、変更後に追加の操作が必要なことがあります。たとえば、ビューの変更後、データベースサーバはそのビューを自動的に再コンパイルして、データベースサーバが使用できるように有効にします。従属した通常のビューが存在する場合、データベースサーバはそれらも無効にしてからもう一度有効にします。有効にできない場合、ステータスは INVALID になるため、通常のビューの定義と従属した通常のビューの定義が一貫性を保つようにする必要があります。通常のビューに従属ビューが存在するかどうかを判断するには、sa_dependent_views システムプロシージャを使用します。

関連情報

ビューの依存性 [50 ページ]
通常のビューの削除 [59 ページ]

1.1.10.4.4 通常のビューの削除

ビューが必要なくなったら、削除します。

前提条件

所有者であるか、または DROP ANY VIEW と DROP ANY OBJECT のいずれかのシステム権限を持っている必要があります。

ビューを削除する前に、ビューを参照するすべての INSTEAD OF トリガを削除してください。

コンテキスト

ビューの名前を変更する場合も、ビューを削除 (および再作成) してください。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. ビューを右クリックして、*[削除]* をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

4. はいをクリックします。

結果

通常のビューの定義が、データベースから削除されます。

次のステップ

従属ビューを持つ通常のビューを削除すると、削除操作の一環としてその従属ビューは INVALID になります。従属ビューは、変更されるか、元の削除済みビューが再作成されるまで、使用できません。

通常のビューに従属ビューが存在するかどうかを判断するには、sa_dependent_views システムプロシージャを使用します。

関連情報

ビューの依存性 [50 ページ]
INSTEAD OF トリガ [118 ページ]
通常のビューの変更 [58 ページ]
トリガの削除 [114 ページ]

1.1.10.4.5 通常のビューの無効化または有効化 (SQL Central の場合)

通常のビューを有効または無効にすることで、データベースサーバがそのビューを使用できるかどうかを制御します。

前提条件

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

通常のビューを有効にするには、基本となるテーブルに対する SELECT 権限または SELECT ANY TABLE システム権限も必要です。

通常のビューを有効にする前に、そのビューが参照する無効なビューを再度有効にする必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. 通常のビューを無効にするには、ビューを右クリックし、*[無効にする]* をクリックします。
4. 通常のビューを有効にするには、ビューを右クリックし、*[再コンパイルして有効にする]* をクリックします。

結果

通常のビューを無効にした場合、データベースサーバはデータベース内のビューの定義を保持しますが、クエリを満たすときにそのビューを使用できません。

無効なビューをクエリが明示的に参照すると、そのクエリは失敗し、エラーが返されます。

次のステップ

このため、ビューを再度有効にしたら、無効にした時点でそのビューに依存していた他のすべてのビューを、もう一度有効にする必要があります。ビューを無効にする前に sa_dependent_views システムプロシージャを使用することで、従属ビューのリストを特定できます。

通常のビューを有効にすると、データベースサーバはデータベース内に格納されたビューの定義を使用してそのビューを再コンパイルします。コンパイルが成功すると、ビューのステータスが VALID に変更されます。再コンパイルに失敗した場合、1 つ以上の参照先オブジェクトでスキーマが変更された可能性があります。その場合は、ビュー定義と参照先のオブジェクトのどちらかを変更し、相互に整合させてから、ビューを有効にします。

ビューが無効になったら、データベースサーバがそのビューを使用できるように、明示的にもう一度有効にする必要があります。

1.1.10.4.6 通常のビューの無効化または有効化 (SQL の場合)

通常のビューを有効または無効にすることで、データベースサーバがそのビューを使用できるかどうかを制御します。

前提条件

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

通常のビューを有効にするには、基本となるテーブルに対する SELECT 権限または SELECT ANY TABLE システム権限も必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

通常のビューを有効にする前に、そのビューが参照する無効なビューを再度有効にする必要があります。

コンテキスト

ビューを無効にすると、そのビューを直接的または間接的に参照する他のビューも自動的に無効になります。このため、ビューを再度有効にしたら、無効にした時点でそのビューに依存していた他のすべてのビューを、もう一度有効にする必要があります。ビューを無効にする前に sa_dependent_views システムプロシージャを使用することで、従属ビューのリストを特定できます。

手順

1. データベースに接続します。
2. 通常のビューを無効にするには、ALTER VIEW...DISABLE 文を実行します。
3. 通常のビューを有効にするには、ALTER VIEW...ENABLE 文を実行します。

結果

通常のビューを無効にした場合、データベースサーバはデータベース内のビューの定義を保持しますが、クエリを満たすときにそのビューを使用できません。

無効なビューをクエリが明示的に参照すると、そのクエリは失敗し、エラーが返されます。

例

次の例では、GROUPO が所有する通常のビュー ViewSalesOrders が無効になります。

```
ALTER VIEW GROUPO.ViewSalesOrders DISABLE;
```

次の例では、GROUPO が所有する通常のビュー ViewSalesOrders がもう一度有効になります。

```
ALTER VIEW GROUPO.ViewSalesOrders ENABLE;
```

次のステップ

このため、ビューを再度有効にしたら、無効にするまでそのビューに依存していた他のすべてのビューを、もう一度有効にする必要があります。ビューを無効にする前に sa_dependent_views システムプロシージャを使用することで、従属ビューのリストを特定できます。

通常のビューを有効にすると、データベースサーバはデータベース内に格納されたビューの定義を使用してそのビューを再コンパイルします。コンパイルが成功すると、ビューのステータスが VALID に変更されます。再コンパイルに失敗した場合、1 つ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

以上の参照先オブジェクトでスキーマが変更された可能性があります。その場合は、ビュー定義と参照先のオブジェクトのどちらかを変更し、相互に整合させてから、ビューを有効にします。

ビューが無効になったら、データベースサーバがそのビューを使用できるように、明示的にもう一度有効にする必要があります。

1.1.10.4.7 通常のビュー内のデータのブラウズ

通常のビュー内のデータをブラウズします。

前提条件

通常のビューは定義済みで、有効になったビューである必要があります。

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。

- そのビューに対する SELECT 権限
- SELECT ANY TABLE システム権限

コンテキスト

通常のビューは、ビューの定義としてデータベースに格納されます。ビューのデータが最新になるように、問い合わせのとき、ビューにデータが取り込まれます。

この作業は、表示する通常のビューを要求する SQL Central で開始し、通常のビューのデータが表示される Interactive SQL で完了します。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをクリックします。
3. ビューを選択して、 ファイル *Interactive SQL によるデータ表示* をクリックします。

結果

InteractiveSQL が起動して、結果 ウィンドウ枠の 結果 タブにビューの内容が表示されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

クエリ [160 ページ]
通常のビューの無効化または有効化 (SQL Central の場合) [60 ページ]

1.1.11 マテリアライズドビュー

マテリアライズドビューとは、ベーステーブルとよく似ていて、参照先のベーステーブルから結果セットが事前に計算されてディスクに格納されるビューです。

概念としては、マテリアライズドビューはビューでもあり (カタログに格納されたクエリ指定がある)、テーブルでもあります (永続的な実体化したローがある)。したがって、テーブルで実行する多くの操作は、マテリアライズドビューでも実行できます。たとえば、マテリアライズドビューに対して、インデックスを構築できます。

マテリアライズドビューを作成すると、データベースサーバが定義を検証して正しく準拠していることを確認します。すべてのカラムとテーブルの参照はデータベースサーバによって完全に修飾されるため、ビューにアクセスするすべてのユーザが同一の定義を確認できます。マテリアライズドビューの作成に成功したら、データを設定します。データの設定は、ビューの初期化とも呼ばれます。

マテリアライズドビューは、SQL Central のビューフォルダにリストされます。

このセクションの内容:

マテリアライズドビューを使用したパフォーマンスの向上 [65 ページ]
マテリアライズドビューを使用すると、ジョインなどのコストの高い操作を事前に計算し、その結果をディスクに保管されるビューの形式で格納することによって、パフォーマンスを大幅に向上させることができます。

マテリアライズドビューの作成 [72 ページ]
マテリアライズドビューを作成して、クエリからのデータを格納します。

マテリアライズドビューの初期化 [73 ページ]
マテリアライズドビューを初期化してデータを取り込み、データベースサーバから使用できるようにします。

マテリアライズドビューの手動での再表示 [74 ページ]
自動的に再表示するよう設定されていないマテリアライズドビューを、手動で再表示します。

マテリアライズドビューの有効化または無効化 [75 ページ]
マテリアライズドビューを有効または無効にすることで、問い合わせに使用できるかどうか管理します。

マテリアライズドビュー定義の難読化 [76 ページ]
マテリアライズドビューの定義をユーザから隠します。データベースに格納されているビュー定義を難読化するものです。

マテリアライズドビューの削除 [78 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。

マテリアライズドビューの暗号化または復号化 [79 ページ]
データのセキュリティ強化のために、マテリアライズドビューを暗号化します。

オプティマイザによるマテリアライズドビューの使用の有効化または無効化 [80 ページ]
クエリを満たすため、オプティマイザによるマテリアライズドビューの使用を有効または無効にします。

カタログ内のマテリアライズドビュー情報の表示 [81 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

すべてのマテリアライズドビューとそのステータスのリストを表示し、また、各マテリアライズドビューを作成したときに実行されていたデータベースオプションも確認します。

[マテリアライズドビューの再表示タイプの変更 [82 ページ]](#)
マテリアライズドビューの再表示タイプを手動ビューから即時ビュー、またその逆に変更します。

高度: マテリアライズドビューのステータスとプロパティ [83 ページ]
マテリアライズドビューの可用性と状態は、そのステータスとプロパティから判断できます。

高度: マテリアライズドビューに対するデータの古さの制御の設定 [87 ページ]
マテリアライズドビュー内のデータは、そのビューが参照するテーブルのデータが変更されることによって古くなります。

1.1.11.1 マテリアライズドビューを使用したパフォーマンスの向上

マテリアライズドビューを使用すると、ジョインなどのコストの高い操作を事前に計算し、その結果をディスクに保管されるビューの形式で格納することによって、パフォーマンスを大幅に向上させることができます。

オプティマイザでは、クエリを満たす最も効率的な方法を決定するときに、クエリ内でマテリアライズドビューが参照されていなくても、マテリアライズドビューを検討します。

アプリケーションの設計では、負荷の高い集約操作やジョイン操作を含むクエリのように、コストの高いクエリやクエリでコストの高い部分を頻繁に実行する場合は、マテリアライズドビューを定義することを検討してください。マテリアライズドビューは、次のような環境でのパフォーマンスを向上することを目的に設計されています。

- データベースのサイズが大きい。
- 頻繁なクエリにより、大量のデータに対する集約操作やジョイン操作が繰り返し実行される。
- 基本となるデータへの変更は頻度が比較的少ない。
- 最新のデータにアクセスすることは重大な要件ではない。

マテリアライズドビューを使用する前に、次の要件、設定、制限を検討してください。

ディスク領域の要件

マテリアライズドビューにはベーステーブルからのデータの複製が含まれるため、作成するマテリアライズドビューのサイズ分の領域をデータベースのディスク上に追加で割り付ける必要があることがあります。得られる利点がマテリアライズドビューの使用コストと釣り合うように、追加領域の要件は慎重に検討する必要があります。

保守コストとデータの最新性の要件

マテリアライズドビューのデータは、基本となるテーブルのデータが変更されたときに再表示する必要があります。次のような競合要因を考慮の上、マテリアライズドビューを再表示しなければならない頻度を判断する必要があります。

基本となるデータの変更頻度

データに対して頻繁な変更や大規模な変更が行われると、手動ビューが古くなります。データの最新性が重要な場合は、即時ビューの使用を検討します。

再表示のコスト

各マテリアライズドビューの基本となるクエリの複雑さや関係するデータの量に応じて、再表示に必要な計算のコストが非常に高くなることがあります。そのためマテリアライズドビューが頻繁に再表示されると、データベースサーバが耐えられないほどの負荷がかかる可能性があります。さらに、マテリアライズドビューは再表示操作中は使用できません。

アプリケーションのデータ最新性の要件

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベースサーバが古いマテリアライズドビューを使用すると、アプリケーションに対して古いデータを提示することになります。古いデータは、基本となるテーブル内のデータの現在の状態を表していません。古さの程度は、マテリアライズドビューが再表示される頻度によって決まります。高いパフォーマンスを実現するために、許容できる古さの程度を判断するようにアプリケーションを設計する必要があります。

データの一貫性の要件

マテリアライズドビューを再表示するときは、マテリアライズドビューを再表示しなければならない一貫性を判断する必要があります。

最適化の使用

クエリの実行時にオプティマイザがマテリアライズドビューを検討することを検証してください。特定のクエリで使用されるマテリアライズドビューのリストは、Interactive SQL でクエリのグラフィカルなプランの高度な詳細ウィンドウで確認できます。

データ変更操作

マテリアライズドビューは読み込み専用であるため、データ変更操作 (INSERT、LOAD、DELETE、UPDATE など) を適用できません。

キー、制約、トリガ、アーティクル

マテリアライズドビューのインデックスは作成できますが、キー、制約、トリガ、またはアーティクルを作成することはできません。

このセクションの内容:

[マテリアライズドビューとビューの依存性 [66 ページ]]
マテリアライズドビューを有効または無効にすることで、データベースサーバがそのビューを使用できるかどうかを制御できます。

再表示タイプの手動または即時への設定 [67 ページ]
マテリアライズドビューには、手動と即時の 2 つの再表示タイプがあります。

マテリアライズドビューの制限 [69 ページ]
マテリアライズドビューを作成、初期化、再表示、および使用する際に多くの制限があります。

関連情報

高度: マテリアライズドビューに対するデータの古さの制御の設定 [87 ページ]
高度: クエリ実行プラン [210 ページ]
オプティマイザによるマテリアライズドビューの使用の有効化または無効化 [80 ページ]

1.1.11.1.1 マテリアライズドビューとビューの依存性

マテリアライズドビューを有効または無効にすることで、データベースサーバがそのビューを使用できるかどうかを制御できます。

無効になったマテリアライズドビューは、最適化時にオプティマイザによって検討されません。クエリが無効なマテリアライズドビューを明示的に参照している場合、そのクエリは失敗し、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、デー

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

タベースサーバはそのビューのデータを削除しますが、定義はデータベース内に保持します。マテリアライズドビューをもう一度有効にすると、初期化されていない状態になるため、データを設定するためにはそのビューを再表示する必要があります。

マテリアライズドビューに依存する通常のビューは、マテリアライズドビューが無効になると、データベースサーバによって自動的に無効になります。その結果、マテリアライズドビューをもう一度有効にする場合は、すべての従属ビューをもう一度有効にする必要があります。このため、マテリアライズドビューを無効にする前に、ビューの依存性のリストを決定します。この操作は、sa_dependent_views システムプロシージャを使用して行います。このプロシージャは ISYSDEPENDENCY システムテーブルを検査して、従属ビューが存在する場合はそのリストを返します。

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。

関連情報

依存性とスキーマ変更 [51 ページ]
マテリアライズドビューの有効化または無効化 [75 ページ]
依存性情報の取得 (SQL の場合) [52 ページ]

1.1.11.1.2 再表示タイプの手動または即時への設定

マテリアライズドビューには、手動と即時の 2 つの再表示タイプがあります。

手動ビュー

手動のマテリアライズドビューまたは手動ビューとは、再表示タイプが MANUAL REFRESH と定義されているマテリアライズドビューです。手動ビューは、たとえば REFRESH MATERIALIZED VIEW 文または sa_refresh_materialized_views システムプロシージャを使用して再表示を明示的に要求するまで再表示されないため、手動ビューのデータが古くなる場合があります。デフォルトでは、マテリアライズドビューを作成すると手動ビューになります。

基本となるいずれかのテーブルが変更されると、マテリアライズドビューのデータが変更による影響を受けなくても、手動ビューは古くなったと見なされます。a_materialized_view_info システムプロシージャが返す DataStatus の値を調べて、手動ビューが古くなったかどうかを判断することができます。S が返されると、手動ビューが古いことを表します。

即時ビュー

即時のマテリアライズドビューまたは即時ビューとは、再表示タイプが IMMEDIATE REFRESH と定義されているマテリアライズドビューです。即時ビューのデータは、基本となるテーブルへの変更がビューのデータに影響を与える場合に、自動的に再表示されます。基本となるテーブルへの変更がビューのデータに影響しない場合、ビューは再表示されません。

また、即時ビューが再表示されるときは、古いローのみを変更する必要があります。この点が手動ビューの再表示とは異なります。手動ビューの再表示では、再表示するためにすべてのデータが削除され、再作成されます。

手動ビューを即時ビューに、または即時ビューを手動ビューに変更できます。ただし、手動ビューから即時ビューへの変更処理では、少し多くの手順があります。

マテリアライズドビューの再表示タイプを変更すると、特に手動ビューから即時ビューに変更する場合は、ビューのステータスとプロパティに影響を与える場合があります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

古さと手動マテリアライズドビュー [68 ページ]
基本となるベーステーブルで変更が発生すると、手動で再表示されるマテリアライズドビューは古くなります。

関連情報

高度: マテリアライズドビューのステータスとプロパティ [83 ページ]
マテリアライズドビューの再表示タイプの変更 [82 ページ]

1.1.11.1.2.1 古さと手動マテリアライズドビュー

基本となるベーステーブルで変更が発生すると、手動で再表示されるマテリアライズドビューは古くなります。

マテリアライズドビューに対して設定された古さのしきい値をデータが超えた場合、オプティマイザはそのビューを、クエリを満たすための候補と見なしません。手動ビューを再表示するということは、データベースサーバがそのビューのクエリ定義を再実行し、ビューのデータをそのクエリの新しい結果セットで置き換えることを意味します。再表示を行うと、ビューのデータが基本となるデータと一致します。手動ビューのデータの古さについて許容可能な程度を検討し、再表示の方式を考案してください。その際には、再表示操作中はビューでクエリを処理できないため、再表示が完了するまでにかかる時間を考慮する必要があります。

また、イベントを使用してビューを再表示する方式を設定することもできます。たとえば、イベントを作成して一定の間隔で再表示することができます。

即時マテリアライズドビューは、トランケートされた後などの未初期化状態 (データがない) の場合を除き、再表示する必要はありません。

materialized_view_optimization データベースオプションを使用して、古さのしきい値を設定できます。オプティマイザは、クエリの処理時にこのしきい値を超えたマテリアライズドビューを使用しません。

注記

データベースサーバをアップグレードした後、またはアップグレード後のデータベースサーバで使用できるようにデータベースを再構築またはアップグレードした後は、マテリアライズドビューを再表示してください。

関連情報

高度: マテリアライズドビューに対するデータの古さの制御の設定 [87 ページ]
マテリアライズドビューの手動での再表示 [74 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.11.1.3 マテリアライズドビューの制限

マテリアライズドビューを作成、初期化、再表示、および使用する際に多くの制限があります。

作成に関する制限

- マテリアライズドビューを作成するときは、マテリアライズドビューの定義でカラム名を明示的に定義する必要があります。カラム定義の一部として `SELECT *` 構成要素を含めることはできません。
- TIMESTAMP WITH TIME ZONE として定義されているカラムを、マテリアライズドビューに含めないでください。time_zone_adjustment オプションの値は、ロケーションと日付に基づいて、接続ごとに異なる値となる場合があるため、不正な結果や予期しない動作が発生することがあります。
- マテリアライズドビューを作成するときは、マテリアライズドビューの定義に次の項目を含めることはできません。
  - 他のビュー (マテリアライズドビューまたは通常のビュー) に対する参照
  - リモートテーブルまたはテンポラリテーブルに対する参照
  - CURRENT USER などの変数。すべての式は決定的でなければなりません
  - ストアドプロシージャ、ユーザ定義関数、または外部関数の呼び出し
  - Transact-SQL 外部ジョイン
  - FOR XML 句

  グループ化されたプロジェクト選択ジョインのクエリブロックには select リストに COUNT(*) を含める必要があり、SUM と COUNT 集合関数のみを使用できます。
- 次のデータベースオプションでは、マテリアライズドビューを作成するときに特定の設定が必要です。そのように設定しない場合は、エラーが返されます。これらのデータベースオプションの値は、オプティマイザが使用するビューにも必要です。
  - ansinull=On
  - conversion_error=On
  - divide_by_zero_error=On
  - sort_collation=Internal
  - string_rtruncation=On
- 次のデータベースオプションは、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ます。マテリアライズドビューが最適化で使用されるには、接続の現在のオプションの値がビューに対して格納された値と一致する必要があります。
  - date_format
  - date_order
  - default_timestamp_increment
  - first_day_of_week
  - nearest_century
  - precision
  - scale
  - time_format
  - timestamp_format
  - timestamp_with_time_zone_format
  - default_timestamp_increment

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- uuid_has_hyphens
- ビューが再表示されるときは、上記の項目に示されたすべてのオプションの接続設定が無視されます。代わりに、データベースオプションの設定 (格納されたビューの設定と一致する必要がある) が使用されます。

マテリアライズドビュー定義内の ORDER BY 句は影響しない

マテリアライズドビューは、ローが特定の順序で格納されない点がベーステーブルに似ています。データベースサーバは、データの計算時に最も効率の良い方法でローを並べます。そのため、マテリアライズドビュー定義で ORDER BY 句を指定しても、ビューが実体化されるときのローの順序に影響はありません。また、ビューマッチングの実行時に、ビューの定義内の ORDER BY 句はオプティマイザによって無視されます。

マテリアライズドビューを手動から即時に変更する際の制限

手動ビューを即時ビューに変更するときには、次の制限が検査されます。ビューがいずれかの制限に違反している場合は、エラーが返されます。

> **注記**
>
> sa_materialized_view_can_be_immediate システムプロシージャを使用して、手動ビューを即時ビューにする条件がそろっているかどうかを調べることができます。

- ビューは未初期化状態である必要があります。
- ビューに外部ジョインがない場合は、ビューの NULL 入力不可のカラムにユニークインデックスが作成されている必要があります。ビューに外部ジョインがある場合は、ビューの NULL 入力不可のカラムにユニークインデックスを作成するか、NULL 入力可のカラムにユニークインデックスを WITH NULLS NOT DISTINCT と宣言して作成してください。
- ビューの定義がグループ化されたクエリである場合、ユニークインデックスのカラムは、集合関数ではない SELECT リスト項目に対応する必要があります。
- ビューの定義に次のものを含むことはできません。
  - GROUPING SETS 句
  - CUBE 句
  - ROLLUP 句
  - DISTINCT 句
  - ロー制限句
  - 非決定的な式
  - セルフジョインと再帰ジョイン
  - LATERAL、CROSS APPLY、または APPLY 句
- ビューの定義は、単一のプロジェクト選択ジョインまたはグループ化されたプロジェクト選択ジョインのブロックである必要があり、グループ化されたプロジェクト選択ジョインのクエリブロックには HAVING 句を含めることはできません。
- グループ化されたプロジェクト選択ジョインのクエリブロックには、SELECT リストに COUNT(*) を含める必要があり、SUM と COUNT 集合関数のみを使用できます。
- SELECT リストの集合関数は、複雑な式の中では参照できません。たとえば、`SUM( expression ) + 1` は SELECT リストで使用できません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- SELECT リストに SUM(expression) 集合関数が含まれており、expression が NULL 入力可能な式である場合は、SELECT リストに COUNT(expression) 集合関数を含めます。
- ビュー定義に外部ジョイン (LEFT OUTER JOIN、RIGHT OUTER JOIN、FULL OUTER JOIN) が含まれる場合は、ビュー定義が以下の追加条件を満たすようにしてください。
  1. テーブル T が OUTER JOIN の ON 条件で保護側として参照されている場合は、T にはプライマリキーがあり、かつ、そのプライマリキーのカラムがビューの SELECT リストに含まれている必要があります。たとえば、SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON T1.Y = R.Y として定義された即時マテリアライズドビュー V において、ON 句で保護テーブル T1 が参照され、かつ、このテーブルのプライマリキーのカラム T1.pk が即時マテリアライズドビュー V の SELECT リストに含まれています。
  2. 外部ジョインの各 NULL 入力側については、NULL 入力不可のカラムのいずれかが即時マテリアライズドビューの SELECT リストに含まれているベーステーブルが、少なくとも 1 つ存在する必要があります。たとえば、SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON T1.Y = R1.Y と定義された即時マテリアライズドビュー V の場合、左外部ジョインの NULL 入力側はテーブル式 (R1 KEY JOIN R2) です。カラム R1.X はテーブル R1 の NULL 入力不可のカラムであり、V の SELECT リストに含まれています。
  3. ビューが GROUP 化されたビューで以前の条件が保持されていない場合は、外部ジョインの各 NULL 入力側に、ベーステーブル T (NULL 入力不可のカラムの 1 つである T.C が、即時マテリアライズドビューの SELECT リストの集合関数 COUNT(T.C) で使用されている) を少なくとも 1 つ指定してください。たとえば、SELECT T1.pk, COUNT(R1.X) FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON T1.Y = R1.Y GROUP BY T1.pk と定義された即時マテリアライズドビュー V の場合、左外部ジョインの NULL 入力側はテーブル式 (R1 KEY JOIN R2) です。集合関数 COUNT(R1.X) は V の SELECT リストに含まれており、R1.X はテーブル R1 の NULL 入力不可のカラムです。
  4. 外部ジョインを持つビューの述部が、以下の条件を満たすようにしてください。
     - LEFT OUTER JOIN、RIGHT OUTER JOIN、FULL OUTER JOIN の ON 句述部によって、保護テーブル式と NULL 入力テーブル式の両方が参照される必要があります。たとえば、T LEFT OUTER JOIN R ON R.X = 1 では、述部 R.X=1 によって NULL 入力側 R のみが参照されているため、この条件が満たされません。
     - ネストした外部ジョインによって生成された NULL 入力されるローが、すべての述部で拒否される必要があります。つまり、述部で、ネストした外部ジョインによって NULL 入力されるテーブル式が参照されている場合は、その外部ジョインによって生成された、NULL を持つすべてのローが拒否されるようにします。
       たとえば、ビュー V1 SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON (T1.Y = R1.Y) WHERE R1.Z = 10 の述部の R1.Z=10 では、T2 LEFT OUTER JOIN (R1 KEY JOIN R2) によって NULL 入力される可能性のあるテーブル R1 が参照されるため、NULL 入力されるすべてのローが拒否されるようにしてください。カラム R1.Z が NULL の場合、述部が UNKNOWN と評価されるため、このことが必要となります。
       ただし、ビュー V2 SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON (T1.Y = R1.Y) WHERE R1.Z IS NULL はこのプロパティを持っていません。述部 R1.Z IS NULL は NULL 入力側 R1 を参照していますが、テーブル R1 が NULL 入力される (つまり、R1.Z カラムが NULL) 場合は、TRUE と評価されます。NULL 入力されるローを拒否する方法は、NULL を許容しないプロパティと比較して制限が少なくなります。たとえば、述部 R.X IS NOT DISTINCT FROM T.X and rowid(T) IS NOT NULL は、T.X が NULL の場合に TRUE と評価されるため、テーブル T の NULL が許容されます。ただし、述部はベーステーブル T で NULL 入力されるすべてのローを拒否します。

関連情報

高度: マテリアライズドビューのステータスとプロパティ [83 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

外部ジョイン [388 ページ]
インデックスの作成 [40 ページ]
マテリアライズドビューの作成 [72 ページ]

1.1.11.2 マテリアライズドビューの作成

マテリアライズドビューを作成して、クエリからのデータを格納します。

前提条件

自分が所有するマテリアライズドビューを作成する場合は、基礎となるすべてのテーブルに対する SELECT 権限とともに、CREATE MATERIALIZED VIEW システム権限を持っていることが必要です。

他者が所有するマテリアライズドビューを作成する場合は、基礎となるすべてのテーブルに対する SELECT 権限とともに、CREATE ANY MATERIALIZED VIEW または CREATE ANY OBJECT のシステム権限を持っていることが必要です。

コンテキスト

マテリアライズドビューを作成して、頻繁に実行されるために大量のデータで繰り返し集約やジョイン操作が発生するクエリの要求を満たします。マテリアライズドビューを使用すると、ディスクに保管されるビュー形式のコストの高い操作を事前に計算することで、パフォーマンスを向上させることができます。

手順

1. SQL Central で、*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. 左ウィンドウ枠でビューを右クリックし、新規 > マテリアライズドビュー をクリックします。
3. マテリアライズドビュー作成ウィザードの指示に従います。

結果

初期化されていないマテリアライズドビューが、データベースに作成されます。まだデータはありません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

マテリアライズドビューを初期化して、データを取り込んでから使用する必要があります。

関連情報

マテリアライズドビューの制限 [69 ページ]
マテリアライズドビューの削除 [78 ページ]
マテリアライズドビューの初期化 [73 ページ]

1.1.11.3 マテリアライズドビューの初期化

マテリアライズドビューを初期化してデータを取り込み、データベースサーバから使用できるようにします。

前提条件

そのマテリアライズドビューの所有者であるか、そのマテリアライズドビューに対する INSERT 権限を持っているか、または INSERT ANY TABLE 権限を持っていることが必要です。

マテリアライズドビューを作成、初期化、または再表示する前に、マテリアライズドビューの制限をすべて満たしていることを確認してください。

コンテキスト

マテリアライズドビューを初期化するには、マテリアライズドビューの再表示と同じ手順に従います。

データベースの sa_refresh_materialized_views システムプロシージャを使用することで、すべての初期化されていないマテリアライズドビューを 1 回で初期化できます。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、**ビュー**をダブルクリックします。
3. マテリアライズドビューを右クリックして、**データの再表示**をクリックします。
4. 独立性レベルを選択して、*[OK]*をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

マテリアライズドビューにデータが取り込まれ、データベースサーバから使用できるようになります。これで、マテリアライズドビューに問い合わせできます。

次のステップ

マテリアライズドビューに問い合わせして、希望のデータを戻すか確認します。

初期化(再表示)に失敗した場合、マテリアライズドビューは未初期化状態に戻ります。初期化に失敗した場合、マテリアライズドビューの定義を確認して、指定された基本テーブルとカラムが有効で、データベースで使用できるオブジェクトであることを確認します。

関連情報

マテリアライズドビューの制限 [69 ページ]
マテリアライズドビューの削除 [78 ページ]
マテリアライズドビューの有効化または無効化 [75 ページ]

1.1.11.4 マテリアライズドビューの手動での再表示

自動的に再表示するよう設定されていないマテリアライズドビューを、手動で再表示します。

前提条件

そのマテリアライズドビューの所有者であるか、そのマテリアライズドビューに対する INSERT 権限を持っているか、または INSERT ANY TABLE システム権限を持っていることが必要です。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. マテリアライズドビューを右クリックして、データの再表示をクリックします。
4. 独立性レベルを選択して、*[OK]*をクリックします。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

マテリアライズドビューのデータは再表示され、基本となるオブジェクトの最新データを表示します。

次のステップ

マテリアライズドビューに問い合わせして、希望のデータを戻すか確認します。

再表示に失敗した場合は、マテリアライズドビューは未初期化状態に戻ります。これが発生した場合、マテリアライズドビューの定義を確認して、指定された基本テーブルとカラムが有効で、データベースで使用できるオブジェクトであることを確認します。

関連情報

マテリアライズドビューの削除 [78 ページ]
マテリアライズドビューの再表示タイプの変更 [82 ページ]

1.1.11.5 マテリアライズドビューの有効化または無効化

マテリアライズドビューを有効または無効にすることで、問い合わせに使用できるかどうか管理します。

前提条件

そのマテリアライズドビューの所有者であるか、または次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。

- ALTER ANY MATERIALIZED VIEW
- ALTER ANY OBJECT

マテリアライズドビューを有効にするには、基本となるテーブルに対する SELECT 権限または SELECT ANY TABLE システム権限も必要です。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプション	アクション
マテリアライズドビューの有効化	1. ビューを右クリックして、*[再コンパイルして有効にする]* をクリックします。 2. (オプション) ビューを右クリックし、*[データの再表示]* をクリックしてビューにデータを取り込みます。有効後にビューに対して最初に実行されるクエリによってビューにデータが取り込まれるので、この手順はオプションになります。
マテリアライズドビューの無効化	ビューを右クリックして、*[無効にする]* をクリックします。

結果

マテリアライズドビューを有効にすると、データベースサーバから使用できるので、問い合わせできます。

マテリアライズドビューを無効にすると、データとインデックスは削除されます。ビューが即時ビューである場合、手動ビューに変わります。無効であるマテリアライズドビューへの問い合わせは失敗し、エラーが返されます。

次のステップ

ビューを再び有効にしたらビューのインデックスを再構築し、ビューを無効にしたときに即時ビューであった場合は、即時ビューに戻す必要があります。

関連情報

ビューの依存性 [50 ページ]
マテリアライズドビューの再表示タイプの変更 [82 ページ]

1.1.11.6 マテリアライズドビュー定義の難読化

マテリアライズドビューの定義をユーザから隠します。データベースに格納されているビュー定義を難読化するものです。

前提条件

そのマテリアライズドビューの所有者であるか、または次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。

- ALTER ANY MATERIALIZED VIEW
- ALTER ANY OBJECT

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

マテリアライズドビューを隠すと、この設定は元に戻せません。

マテリアライズドビューが隠されると、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。このビューはアンロードして、他のデータベースに再ロードできます。

マテリアライズドビューを非表示にすると元に戻せず、SQL を使用するだけで非表示にできます。

手順

1. データベースに接続します。
2. ALTER MATERIALIZED VIEW...SET HIDDEN 文を実行します。

結果

オートコミットが実行されます。

カタログをブラウズするとき、ビューは表示されません。ただし、ビューは直接参照でき、クエリ処理中に使用できることは変わりません。

例

次の文は、マテリアライズドビュー EmployeeConfid3 を作成し、再表示して、ビュー定義を難読化します。

警告

次の例を実行するとき、作成したマテリアライズドビューを削除してください。このようにしない場合は、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。

```
CREATE MATERIALIZED VIEW EmployeeConfid3 AS
   SELECT EmployeeID, Employees.DepartmentID, SocialSecurityNumber, Salary,
ManagerID,
      Departments.DepartmentName, Departments.DepartmentHeadID
   FROM Employees, Departments
   WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid3;
ALTER MATERIALIZED VIEW EmployeeConfid3 SET HIDDEN;
```

関連情報

マテリアライズドビューの削除 [78 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.11.7 マテリアライズドビューの削除

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。

前提条件

所有者であるか、または DROP ANY MATERIALIZED VIEW と DROP ANY OBJECT のいずれかのシステム権限を持っている必要があります。

マテリアライズドビューを削除する前に、すべての従属ビューを削除または無効にする必要があります。マテリアライズドビューに従属ビューが存在するかどうかを判断するには、sa_dependent_views システムプロシージャを使用します。

コンテキスト

マテリアライズドビューが必要なくなったとき、または基本となる参照オブジェクトにスキーマ変更を実施して、マテリアライズドビュー定義が有効でなくなったとき、この作業を実行します。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. マテリアライズドビューを右クリックして、*[削除]* をクリックします。
4. はいをクリックします。

結果

マテリアライズドビューはデータベースから削除されます。

次のステップ

通常のビューがマテリアライズドビューに依存している場合、これらを有効にすることはできません。ビュー定義を変更するか、ビューを削除する必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

ビューの依存性 [50 ページ]

1.1.11.8 マテリアライズドビューの暗号化または復号化

データのセキュリティ強化のために、マテリアライズドビューを暗号化します。

前提条件

所有者であるか、または CREATE ANY MATERIALIZED VIEW と DROP ANY MATERIALIZED VIEW の両方のシステム権限を持っているか、または CREATE ANY OBJECT と DROP ANY OBJECT の両方のシステム権限を持っていることが必要です。

マテリアライズドビューを暗号化するには、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。

コンテキスト

この作業を実行する例として、基本となるテーブルで暗号化されているデータがマテリアライズドビューに含まれている場合や、マテリアライズドビュー内でもデータを暗号化する場合が挙げられます。

データベースの作成時に指定した暗号化アルゴリズムとキーを使用して、マテリアライズドビューを暗号化します。テーブル暗号化が有効であるかどうかなど、暗号化設定がデータベースで有効であることを確認するには、次のように DB_PROPERTY 関数を使用して Encryption データベースプロパティの値を取得します。

```
SELECT DB_PROPERTY( 'Encryption' );
```

テーブルの暗号化と同様に、マテリアライズドビューを暗号化するとパフォーマンスに影響がある可能性があります。データベースサーバがビューから取得したデータを復号化する必要があるためです。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. マテリアライズドビューを右クリックして、プロパティをクリックします。
4. その他タブをクリックします。
5. 必要に応じて、*[マテリアライズド ビューのデータは暗号化済み]* チェックボックスをオンにする、またはクリアします。
6. *OK* をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

マテリアライズドビューのデータが暗号化されます。

1.1.11.9 オプティマイザによるマテリアライズドビューの使用の有効化または無効化

クエリを満たすため、オプティマイザによるマテリアライズドビューの使用を有効または無効にします。

前提条件

所有者であるか、または ALTER ANY MATERIALIZED VIEW と ALTER ANY OBJECT のいずれかのシステム権限を持っている必要があります。

コンテキスト

クエリがマテリアライズドビューを参照しない場合でも、クエリを満たすとパフォーマンスが向上する場合、オプティマイザは、クエリを満たすためビューを使用するよう決定できます。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ビューをダブルクリックします。
3. マテリアライズドビューを右クリックして、プロパティをクリックします。
4. 必要に応じて、*[一般]* タブをクリックして、*[最適化に使用]* をオンにする、またはクリアします。
5. *OK* をクリックします。

結果

オプティマイザがマテリアライズドビューを使用できる場合、クエリでビューが明示的に参照されないとしても、オプティマイザは、クエリを満たすための最良プランを計算するときにビューを考慮します。オプティマイザがマテリアライズドビューを使用できない場合、オプティマイザはビューを考慮しません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

クエリ実行プランを見て、オプティマイザがビューを使用するかどうか判断するため、ビューの基本となるオブジェクトに問い合わせます。ただし、ビューの可用性によって、オプティマイザがビューを使用しない可能性があります。オプティマイザの選択は、パフォーマンスに基づいています。

関連情報

マテリアライズドビューを使用したパフォーマンスの向上 [65 ページ]
高度: クエリ実行プラン [210 ページ]

1.1.11.10 カタログ内のマテリアライズドビュー情報の表示

すべてのマテリアライズドビューとそのステータスのリストを表示し、また、各マテリアライズドビューを作成したときに実行されていたデータベースオプションも確認します。

前提条件

マテリアライズドビューを非表示にすることはできません。

コンテキスト

依存性の情報は、SYSDEPENDENCY システムビューで探すこともできます。

手順

1. データベースに接続します。
2. すべてのマテリアライズドビューとそのステータスのリストを表示するには、次の文を実行します。

```
SELECT * FROM sa_materialized_view_info();
```

3. マテリアライズドビューが作成されたときに、各ビューに対して実行されていたデータベースオプションを確認するには、次の文を実行します。

```
SELECT b.object_id, b.table_name, a.option_id, c.option_name, a.option_value
FROM SYSMVOPTION a, SYSTAB b, SYSMVOPTIONNAME c
WHERE a.view_object_id=b.object_id
AND b.table_type=2;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

4. 所定のマテリアライズドビューに依存している通常のビューのリストを要求するには、次の文を実行します。

```
CALL sa_dependent_views( 'materialized-view-name' );
```

結果

要求されたマテリアライズドビュー情報が返されます。

関連情報

高度: マテリアライズドビューのステータスとプロパティ [83 ページ]

1.1.11.11 マテリアライズドビューの再表示タイプの変更

マテリアライズドビューの再表示タイプを手動ビューから即時ビュー、またその逆に変更します。

前提条件

所有者であるか、または CREATE ANY MATERIALIZED VIEW と DROP ANY MATERIALIZED VIEW の両方のシステム権限を持っているか、または CREATE ANY OBJECT と DROP ANY OBJECT の両方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。

手動から即時に変更するには、ビューが未初期化状態である (データを含まない) 必要があります。ビューが作成された直後でまだ再表示されていない場合は、未初期化状態になっています。マテリアライズドビューにデータがある場合、即時に変更する前に TRUNCATE 文を実行して未初期化状態に戻す必要があります。また、マテリアライズドビューにはユニークインデックスも必要で、即時ビューに必要な制限に従う必要があります。

再表示タイプの変更以外の手順を追加しないで、即時ビューをいつでも手動に変更できます。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、**ビュー**をダブルクリックします。
3. マテリアライズドビューを右クリックして、**プロパティ**をクリックします。
4. **再表示タイプ**フィールドで、次のいずれかのオプションを選択します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプション	アクション
手動ビューから即時ビューへの変更	即時
即時ビューから手動ビューへの変更	手動

5. *OK* をクリックします。

結果

マテリアライズドビューの再表示タイプが変更されます。基本となるオブジェクトのデータが変更されると、即時ビューが更新されます。手動ビューを再表示すると、手動ビューが更新されます。

次のステップ

手動から即時へ変更したら、ビューを初期化 (再表示) してデータを取り込む必要があります。

関連情報

再表示タイプの手動または即時への設定 [67 ページ]
マテリアライズドビューの制限 [69 ページ]
マテリアライズドビューの初期化 [73 ページ]
インデックスの作成 [40 ページ]

1.1.11.12 高度: マテリアライズドビューのステータスとプロパティ

マテリアライズドビューの可用性と状態は、そのステータスとプロパティから判断できます。

既存のマテリアライズドビューのステータスとプロパティを判断するのに一番良い方法は、sa_materialized_view_info システムプロシージャを使用することです。

また、SQL Central の**ビュー**フォルダを選択して個々のビュー詳細を調べるか、SYSTAB および SYSVIEW システムビューを問い合わせることにより、マテリアライズドビューの情報を表示できます。

このセクションの内容:

マテリアライズドビューのステータス [84 ページ]
マテリアライズドビューには、有効および無効という 2 種類のステータスがあります。

マテリアライズドビューのプロパティ [84 ページ]
マテリアライズドビューのプロパティは、ビューを使用するかどうかを評価するときにオプティマイザによって使用されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

マテリアライズドビューの変更、再表示、トランケート実行時のステータスとプロパティの変更 [85 ページ]
マテリアライズドビューで変更、再表示、トランケートなどの操作を実行すると、ビューのステータスとプロパティに影響を与えます。

1.1.11.12.1 マテリアライズドビューのステータス

マテリアライズドビューには、有効および無効という 2 種類のステータスがあります。

有効

マテリアライズドビューは、コンパイルが正常に実行され、データベースサーバから使用できます。有効にされたマテリアライズドビューにはデータがない場合もあります。たとえば、有効にされたマテリアライズドビューからデータをトランケートすると、有効に変わりますが、初期化されていません。マテリアライズドビューを初期化することはできますが、マテリアライズドビューの定義を満たす基本となるテーブルにデータがない場合は、空になります。これは、初期化されていないためにデータがないマテリアライズドビューと同じではありません。

無効

マテリアライズビューは、ALTER MATERIALIZED VIEW...DISABLE 文の使用などによって明示的に無効化されました。マテリアライズドビューを無効にすると、そのビューのデータとインデックスは削除されます。また、即時ビューを無効にすると、手動ビューに変わります。

ビューが有効であるか、無効であるかを判断するには、ビューのステータスプロパティを返す sa_materialized_view_info システムプロシージャを使用します。

関連情報

マテリアライズドビューのプロパティ [84 ページ]
マテリアライズドビューの有効化または無効化 [75 ページ]

1.1.11.12.2 マテリアライズドビューのプロパティ

マテリアライズドビューのプロパティは、ビューを使用するかどうかを評価するときにオプティマイザによって使用されます。

次のリストでは、sa_materialized_view_info システムプロシージャが返すマテリアライズドビューのプロパティについて説明します。

Status

ビューが有効であるか、無効であるかを示します。

DataStatus

ビュー内のデータのステータスを反映します。たとえば、ビューが初期化されているかどうか、ビューが古いかどうかなどを示します。マテリアライズドビューが最後に再表示されてから基本となるテーブルのデータが変更されると、手動ビューは古くなります。即時ビューが古くなることはありません。

ViewLastRefreshed

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ビューが最後に再表示された時刻を示します。

DateLastModified

ビューが古い場合に、基本となるテーブルのデータが最後に修正された時刻を示します。

AvailForOptimization

オプティマイザがビューを使用できるかどうかを反映します。

RefreshType

ビューが手動ビューであるか、即時ビューであるかを示します。

各プロパティで使用できる値については、sa_materialized_view_info システムプロシージャを使用してください。

手動ビューを即時ビューに変換できるかどうかを示すプロパティはありませんが、sa_materialized_view_can_be_immediate システムプロシージャを使用してこれを判断できます。

関連情報

マテリアライズドビューのステータス [84 ページ]

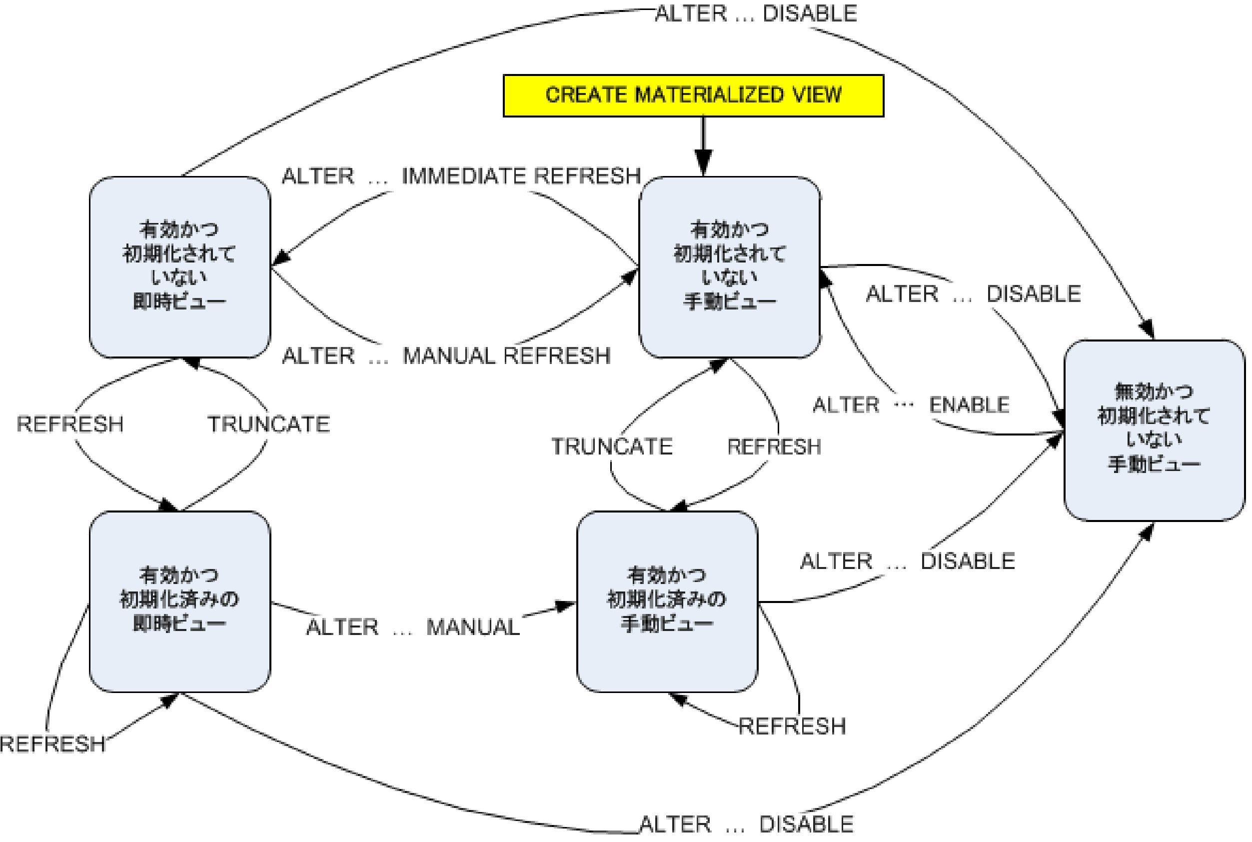
1.1.11.12.3 マテリアライズドビューの変更、再表示、トランケート実行時のステータスとプロパティの変更

マテリアライズドビューで変更、再表示、トランケートなどの操作を実行すると、ビューのステータスとプロパティに影響を与えます。

次の図に、これらのタスクがマテリアライズドビューのステータスと一部のプロパティに与える影響を示します。

この図では、灰色の四角がマテリアライズドビューであり、即時ビューは IMMEDIATE という用語、手動ビューは MANUAL という用語で区別されています。灰色のボックスの間を接続する ALTER という用語は、ALTER MATERIALIZED VIEW の省略です。マテリアライズドビューのステータスを変更するために SQL 文が示されていますが、SQL Central を使用してこれらの操作を実行することもできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.



- マテリアライズドビューを作成すると、そのビューは有効な手動ビューになり、未初期化状態です (データは含まれません)。
- 未初期化状態のビューを再表示すると、初期化された状態になります (データが入力されます)。
- 手動ビューから即時ビューに変更するにはいくつかの手順が必要です。また、即時ビューには追加の制限があります。
- マテリアライズドビューを無効にすると、次のことが行われます。
  - データが削除されます
  - ビューが未初期化状態に戻ります
  - インデックスが削除されます
  - 即時ビューが手動に戻ります

関連情報

再表示タイプの手動または即時への設定 [67 ページ]
マテリアライズドビューの制限 [69 ページ]
マテリアライズドビューのプロパティ [84 ページ]
マテリアライズドビューのステータス [84 ページ]
マテリアライズドビューの再表示タイプの変更 [82 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.1.11.13 高度: マテリアライズドビューに対するデータの古さの制御の設定

マテリアライズドビュー内のデータは、そのビューが参照するテーブルのデータが変更されることによって古くなります。

マテリアライズドビューがオプティマイザによって検討されない場合、古さに原因がある可能性があります。マテリアライズドビューに対する古さのしきい値は、materialized_view_optimization データベースオプションを使用して調整します。

ビューを再表示するイベントまたはトリガに対して指定した間隔を調整することもできます。

クエリがマテリアライズドビューを明示的に参照している場合、そのビューのデータの古さに関係なく、ビューがクエリの処理に使用されます。さらに、materialized_view_optimization データベースオプションの設定を上書きし、マテリアライズドビューを強制的に使用する場合に、SELECT、UPDATE、INSERT などの文の OPTION 句を使用できます。

スナップショットアイソレーションが使用されている場合、トランザクションのスナップショットの開始後にマテリアライズドビューが再表示されると、オプティマイザはそのマテリアライズドビューを使用しません。

1.2 ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数

プロシージャとトリガは、手続き型 SQL 文をデータベースに格納します。

プロシージャとトリガでは、制御文を使用して、SQL 文の繰り返し (LOOP 文) と条件付き実行 (IF 文と CASE 文) ができます。バッチは、データベースサーバにグループとして送られる SQL コマンドのセットです。制御文など、プロシージャとトリガで使用できる機能の多くは、バッチでも使用できます。

警告

ソースコードに対する変更と、ソース (ストアドプロシージャを含む) から作成され、データベースに展開されるオブジェクトに対する変更を追跡する場合は、ソース制御ソフトウェアを使用します。

プロシージャは CALL 文で呼び出され、パラメータを使って値を受け取り、呼び出しを行った環境に結果の値を返します。SELECT 文の FROM 句にプロシージャ名を含めると、プロシージャの結果セットを操作できます。

プロシージャは、呼び出し元に結果セットを返し、他のプロシージャを呼び出すか、またはトリガを起動できます。たとえば、ユーザ定義関数はストアドプロシージャの一種であり、呼び出しを行った環境に 1 つの値を返します。ユーザ定義関数は、渡されたパラメータを変更しないで、クエリや他の SQL 文に使用可能な関数のスコープを拡張します。

トリガは特定のデータベーステーブルに関連付けられます。トリガは、関連するテーブルのローが挿入、更新、削除されるたびに自動的に起動します。トリガでプロシージャを呼び出したり、他のトリガを起動したりはできますが、トリガにパラメータを指定したり、CALL 文で呼び出したりはできません。

ストアド SQL のトラブルシューティングとプロファイリング

ストアドプロシージャをプロファイリングして、SQL Anywhere プロファイラ でパフォーマンスを分析できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

プロシージャ、トリガ、およびユーザ定義関数の利点 [89 ページ]
プロシージャとトリガにより、データベースのセキュリティ、効率、標準化を強化することができます。

プロシージャ [89 ページ]
プロシージャはデータベースで 1 つ以上の特定のタスクを実行します。

ユーザ定義関数 [101 ページ]
ユーザ定義関数は、呼び出しを行った環境に単一の値を返します。

トリガ [107 ページ]
トリガとは、データを修正する文が実行されると自動的に実行されるストアドプロシージャの特別な形式です。

バッチ [119 ページ]
バッチはまとめて送信されてグループとして次々に実行される一連の SQL 文です。

プロシージャ、トリガ、ユーザ定義関数の構造 [122 ページ]
プロシージャ、トリガ、ユーザ定義関数の本文は複合文で構成されています。

制御文 [125 ページ]
プロシージャ、トリガ、またはユーザ定義関数の本文、またはバッチの中には、論理フローや意思決定のための制御文が複数あります。

結果セット [128 ページ]
プロシージャは結果を 1 つまたは複数のローとして返します。

プロシージャ、トリガ、ユーザ定義関数、バッチのカーソル [137 ページ]
カーソルは、結果セットに複数のローがあるクエリまたはストアドプロシージャからローを 1 つずつ取り出します。

エラーと警告の処理 [140 ページ]
アプリケーションプログラムは SQL 文を実行した後、ステータスコード (リターンコード) をチェックできます。ステータスコードは文が正しく実行されたかどうかを示し、エラーの場合はその理由を提示します。

プロシージャ、トリガ、ユーザ定義関数、バッチで使用される EXECUTE IMMEDIATE [152 ページ]
EXECUTE IMMEDIATE 文を使うと、文字列 (引用符で囲む) と変数を使って文を組み立てることができます。

プロシージャ、トリガ、ユーザ定義関数でのトランザクションとセーブポイント [154 ページ]
プロシージャまたはトリガ内の SQL 文は現在のトランザクションの一部です。

プロシージャ、トリガ、ユーザ定義関数、バッチを作成するときのヒント [155 ページ]
プロシージャ、トリガ、ユーザ定義関数、バッチを作成するときに役立つヒントがいくつかあります。

プロシージャ、トリガ、イベント、バッチで使用できる文 [156 ページ]
バッチでは大部分の SQL 文を使用できますが、いくつかの例外があります。

プロシージャ、ファンクション、トリガ、イベント、またはビューの内容を隠す [157 ページ]
SET HIDDEN 句を使用して、プロシージャ、関数、トリガ、イベント、またはビューの内容を隠します。

関連情報

SQL Anywhere のデバッガ [834 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.1 プロシージャ、トリガ、およびユーザ定義関数の利点

プロシージャとトリガにより、データベースのセキュリティ、効率、標準化を強化することができます。

プロシージャとトリガの定義はデータベース内にあり、データベースアプリケーションから分離されています。これには、いくつかの利点があります。

標準化

プロシージャとトリガを使用すると、複数のアプリケーションプログラムで実行するアクションを標準化できます。アクションをコーディングし、将来利用するためにデータベースに格納します。アプリケーションはプロシージャを呼び出すか、トリガを起動するだけで、何度でもそのアクションを実行できます。すべての変更が 1 か所で行われるため、アクションの実装が変更された場合、アクションを使用するすべてのアプリケーションが自動的に新機能を取得します。

効率化

ネットワークデータベースサーバ環境で使用されるプロシージャとトリガは、ネットワーク通信を使用しないでデータベースのデータにアクセスできます。つまり、クライアント上のアプリケーションに実装する場合と比較して、ネットワークのパフォーマンスを低下させることなく高速に実行されます。

プロシージャとトリガを作成すると、自動的に構文チェックを行った後に、システムテーブルに格納されます。アプリケーションが初めてプロシージャを呼び出すか、トリガを起動するときには、システムテーブルからコンパイルされて仮想メモリにロードされ、実行されます。最初に実行された後もプロシージャまたはトリガのコピーがメモリに保持されるため、同じプロシージャまたはトリガの実行を繰り返す場合、すぐに実行できます。また、複数のアプリケーションが同時にプロシージャまたはトリガを使用することも、1 つのアプリケーションが再帰的に使用することもできます。

1.2.2 プロシージャ

プロシージャはデータベースで 1 つ以上の特定のタスクを実行します。

このセクションの内容:

所有者または呼び出し側の権限で実行するプロシージャとファンクション [90 ページ]
　プロシージャまたはファンクションを指定する場合、プロシージャまたはファンクションをその所有者の権限で実行するか、またはそれを呼び出したユーザまたはプロシージャ (呼び出し側) の権限で実行するかを指定できます。

プロシージャの作成 (SQL Central の場合) [96 ページ]
　プロシージャテンプレートを使用してプロシージャを作成するには、プロシージャ作成ウィザードを使用します。

プロシージャの変更 (SQL Central の場合) [97 ページ]
　SQL Central で既存のプロシージャを変更します。

プロシージャの呼び出し (SQL の場合) [98 ページ]
　プロシージャを呼び出し、値を挿入します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

プロシージャのコピー (SQL Central の場合) [99 ページ]
SQL Central を使用して、データベース間または同じデータベース内でプロシージャをコピーします。

プロシージャの削除 (SQL Central の場合) [100 ページ]
プロシージャが必要なくなったときなどに、データベースから削除します。

1.2.2.1 所有者または呼び出し側の権限で実行するプロシージャとファンクション

プロシージャまたはファンクションを指定する場合、プロシージャまたはファンクションをその所有者の権限で実行するか、またはそれを呼び出したユーザまたはプロシージャ (呼び出し側) の権限で実行するかを指定できます。

呼び出し側の特定は常に明白であるとはかぎりません。ユーザがプロシージャを呼び出すこともできますが、プロシージャが別のプロシージャを呼び出すこともできます。このような場合、ログインユーザ (トッププレベルのプロシージャを最初に呼び出したユーザ) と、最初のプロシージャによって呼び出されたプロシージャの所有者の場合もある有効なユーザとは区別されます。プロシージャが呼び出し側の権限で実行される場合、有効なユーザの権限が適用されます。

プロシージャまたはファンクションを作成する場合、CREATE PROCEDURE 文または CREATE FUNCTION 文の SQL SECURITY 句により、プロシージャまたはファンクションを実行するときに適用される権限が、修飾されていないオブジェクトの所有者とともに設定されます。この句では、INVOKER または DEFINER が選択されます。ただし、ユーザは別のユーザが所有するプロシージャまたはファンクションを作成することができます。この場合は、実際は定義者ではなく所有者の権限となります。

プロシージャまたはファンクションを作成するときに、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ます。プロシージャ内のオブジェクトが所有者に対して修飾されていない場合、そのオブジェクトが所有者として実行されるか呼び出し側として実行されるかにより、所有者が異なります。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;
```

別のユーザである user2 がこのプロシージャの実行を試行したときに user2.table1 が存在しない場合は、データベースサーバはエラーを返します。user1.table1 ではなく user2.table1 が存在する場合は、それが使用されます。

呼び出し側の権限を使用してプロシージャまたはファンクションを実行する場合、呼び出し側はプロシージャ、ファンクションまたはシステムプロシージャが動作するデータベースオブジェクトに必要な権限に加えて、そのプロシージャの EXECUTE 権限を持っている必要があります。

プロシージャまたはファンクションを呼び出し側として実行するのか、定義者として実行するのかがわからない場合は、その SQL 定義で SQL SECURITY 句を確認します。

データベースで権限付き操作を実行するプロシージャまたはファンクションに必要な権限を決定する場合は、sp_proc_priv システムプロシージャを使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ユーザコンテキストの決定

SESSION_USER、INVOKING_USER、EXECUTING_USER、および PROCEDURE OWNER 特別値を使用して、プロシージャ実行時のユーザコンテキストを決定します。これらの特別値は、ネストされたプロシージャ (とりわけ SQL SECURITY DEFINER または SQL SECURITY INVOKER として実行するよう設定されている場合) に特に有用です。次のシナリオでは、これらの特別値を使用してユーザコンテキストに関する情報を取得する方法について説明します。

1. 次の文を実行して、テスト用のシナリオを作成します。

```
CREATE USER u1 IDENTIFIED BY pwdforu1;
CREATE USER u2 IDENTIFIED BY pwdforu2;
CREATE USER u3 IDENTIFIED BY pwdforu3;

CREATE PROCEDURE u2.p2()
SQL SECURITY DEFINER
BEGIN
    DECLARE u2_message LONG VARCHAR;
    DECLARE u3_message LONG VARCHAR;

    CALL u3.p3( u3_message );
    SET u2_message = STRING( 'u2.p2: SESSION USER=', SESSION USER,
                             ', INVOKING USER=', INVOKING USER,
                             ', EXECUTING USER=', EXECUTING USER,
                             ', PROCEDURE OWNER=', PROCEDURE OWNER );

    SELECT u2_message AS ret UNION ALL SELECT u3_message;
END;

CREATE PROCEDURE u3.p3( OUT u3_message LONG VARCHAR )
SQL SECURITY INVOKER
BEGIN
    SET u3_message = STRING( 'u3.p3: SESSION USER=', SESSION USER,
                             ', INVOKING USER=', INVOKING USER,
                             ', EXECUTING USER=', EXECUTING USER,
                             ', PROCEDURE OWNER=', PROCEDURE OWNER );
END;

GRANT EXECUTE ON u2.p2 TO u1;
GRANT EXECUTE ON u3.p3 TO u2;
```

2. u2 としてログインし、次の文を実行します。

```
SELECT SESSION USER, INVOKING USER, EXECUTING USER, PROCEDURE OWNER;
```

   この結果では、u1 がプロシージャを実行していないため、PROCEDURE OWNER が NULL でも、SESSION USER、INVOKING USER および EXECUTING USER はすべて u1 であることを示しています。

3. u1 としてログインし、同じ文を実行します。その結果は、u2.p2 内での実行中、以下の内容であることを示しています。
   - ログインユーザが u1 であるため、SESSION USER は u1 です。
   - u2.p2 が u1 により呼び出されているため、INVOKING USER は u1 です。
   - u2.p2 が SQL SECURITY DEFINER プロシージャであるため、EXECUTING USER は u2 です。そのため、プロシージャ内で実行する場合、有効なユーザは u2 に変わります。
   - u2 がプロシージャ u2.p2 を所有しているため、PROCEDURE OWNER は u2 です。

   その結果は、u3.p3 内での実行中、以下の内容であることを示しています。
   - ログインユーザが u1 であるため、SESSION USER は u1 です。
   - u3.p3 が u2.p2 から呼び出され、u2.p2 内で EXECUTING USER が u2 のため、INVOKING USER は u2 です。
   - u3.p3 が SQL SECURITY INVOKER プロシージャのため、EXECUTING USER は u2 です。そのため、実行しているユーザは呼び出し元と変わりません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

○ u3 がプロシージャ u3.p3 を所有しているため、PROCEDURE OWNER は u3 です。

このセクションの内容:

[16.0 より前のバージョンのシステムプロシージャを呼び出し側または定義者として実行する [92 ページ]]
テーブルの変更のような、データベースで権限付きタスクを実行する 16.0 より前のバージョンのソフトウェアに存在するシステムプロシージャには、呼び出し側または定義者 (所有者) のどちらの権限でも実行できるものがあります。

1.2.2.1.1 16.0 より前のバージョンのシステムプロシージャを呼び出し側または定義者として実行する

テーブルの変更のような、データベースで権限付きタスクを実行する 16.0 より前のバージョンのソフトウェアに存在するシステムプロシージャには、呼び出し側または定義者 (所有者) のどちらの権限でも実行できるものがあります。

データベースを作成または初期化する場合、これらの特別なシステムプロシージャを、その所有者 (定義者) の権限で実行するか、または呼び出し側の権限で実行するかを指定できます。

これらのシステムプロシージャを呼び出し側として実行するようデータベースが設定されている場合、すべてのシステムプロシージャは呼び出し側ユーザとして実行されます。システムプロシージャを実行するには、ユーザはそのプロシージャに対する EXECUTE 権限に加え、そのプロシージャの SQL 文が必要とするシステム権限やオブジェクト権限をすべて持つ必要があります。ユーザは、PUBLIC のメンバーなので、EXECUTE 権限を継承しています。

これらのシステムプロシージャを定義者として実行するようデータベースが設定されている場合、すべてのシステムプロシージャは定義者 (通常 dbo ロールまたは SYS ロール) として実行されます。システムプロシージャを実行するには、ユーザが必要とするのはそのプロシージャに対する EXECUTE 権限だけです。この動作は 16.0 より前のデータベースと互換性があります。

> **注記**
>
> ユーザ定義のプロシージャのデフォルトの動作は、呼び出し側/定義者モードの影響を受けることはありません。つまり、ユーザ定義のプロシージャの定義で呼び出し側または定義者が指定されていない場合、プロシージャは定義者の権限で実行されます。

次の方法の 1 つを使用して、データベースでこれらのシステムプロシージャの作成時またはアップグレード時の実行方法を制御できます。

CREATE DATABASE...SYSTEM PROCEDURE AS DEFINER 文

CREATE DATABASE...SYSTEM PROCEDURE AS DEFINER OFF の指定は、データベースサーバが呼び出し側の権限を適用することを意味します。これは、新しいデータベースのデフォルトの動作です。

CREATE DATABASE...SYSTEM PROCEDURE AS DEFINER ON の指定は、データベースサーバが定義者 (所有者) の権限を適用することを意味します。これは、16.0 より前のバージョンのデータベースのデフォルトの動作です。

ALTER DATABASE UPGRADE...SYSTEM PROCEDURE AS DEFINER 文

この句は、CREATE DATABASE 文と同じ動作となります。句が指定されていない場合は、アップグレードされたデータベースの既存の動作が維持されます。たとえば、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た場合、定義者の権限による実行がデフォルトとなります。

-pd オプション、初期化ユーティリティ (dbinit)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベースの作成時に -pd オプションを指定すると、データベースサーバは、これらのシステムプロシージャの実行時に定義者の権限を適用します。-pd オプションを指定しない場合は、呼び出し側の権限の適用がデフォルトの動作となります。

-pd オプション、アップグレードユーティリティ **(dbupgrad)**

データベースのアップグレード時に -pd Y オプションを指定すると、データベースサーバは、これらのシステムプロシージャの実行時に定義者の権限を適用します。

-pd N オプションを指定すると、これらのシステムプロシージャの実行時に、データベースサーバは呼び出し側の権限を適用します。

このオプションが指定されていない場合は、アップグレードされたデータベースの既存の動作が維持されます。

注記

PUBLIC システムロールには、すべてのシステムプロシージャに対する EXECUTE 権限が付与されます。新しく作成されたユーザには、デフォルトで PUBLIC ロールが付与されます。したがって、ユーザはシステムの EXECUTE 権限をすでに持っていることになります。

ユーザ定義関数とプロシージャのデフォルトは、呼び出し側と定義者の決定の影響を受けません。つまり、これらのシステムプロシージャを呼び出し側として実行するよう選択した場合でも、ユーザ定義のプロシージャは定義者のままです。

呼び出し側/定義者設定の影響を受けるプロシージャのリスト

次は、呼び出し側/定義者設定の影響を受けるシステムプロシージャのリストです。これらは、権限付き操作を実行していたデータベース上で 16.0 より前のバージョンの SQL Anywhere のシステムプロシージャです。これらのプロシージャを定義者として実行するようデータベースが設定されている場合、ユーザは、実行する必要のある各プロシージャで EXECUTE 権限のみが必要となります。データベースが INVOKER として実行するよう設定されている場合、ユーザは各プロシージャで EXECUTE 権限を必要としませんが、代わりに各プロシージャを正常に実行するために個別の権限が必要となります。

- sa_audit_string
- sa_clean_database
- sa_column_stats
- sa_conn_activity
- sa_conn_compression_info
- sa_conn_info
- sa_conn_list
- sa_conn_options
- sa_conn_properties
- sa_db_list
- sa_db_properties
- sa_disable_auditing_type
- sa_disk_free_space
- sa_enable_auditing_type
- sa_external_library_unload
- sa_flush_cache
- sa_flush_statistics

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- sa_get_histogram
- sa_get_request_profile
- sa_get_request_times
- sa_get_table_definition
- sa_index_density
- sa_index_levels
- sa_install_feature
- sa_java_loaded_classes
- sa_load_cost_model
- sa_make_object
- sa_materialized_view_can_be_immediate
- sa_procedure_profile
- sa_procedure_profile_summary
- sa_recompile_views
- sa_refresh_materialized_views
- sa_refresh_text_indexes
- sa_remove_tracing_data
- sa_reset_identity
- sa_save_trace_data
- sa_send_udp
- sa_server_option
- sa_set_tracing_level
- sa_table_fragmentation
- sa_table_page_usage
- sa_table_stats
- sa_text_index_vocab_nchar
- sa_unload_cost_model
- sa_user_defined_counter_add
- sa_user_defined_counter_set
- sa_validate
- sa_verify_password
- sp_forward_to_remote_server
- sp_get_last_synchronize_result
- sp_list_directory
- sp_remote_columns
- sp_remote_exported_keys
- sp_remote_imported_keys
- sp_remote_primary_keys
- sp_remote_procedures
- sp_remote_tables
- st_geometry_predefined_srs
- st_geometry_predefined_uom
- xp_cmdshell
- xp_read_file
- xp_sendmail

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- xp_startmail
- xp_startsmtp
- xp_stopmail
- xp_stopsmtp
- xp_write_file

呼び出し側/定義者設定に関係なく呼び出し側の権限で実行されるプロシージャのリスト

権限付き操作を実行する 16.0 より前のバージョンのシステムプロシージャの小さなサブセットでは、呼び出し側/定義者設定に関係なく、タスクを実行するため呼び出し側は追加の権限を持っていることが必要です。これらのプロシージャで必要な追加の権限のリストを表示するには、各プロシージャのマニュアルを参照してください。

- sa_locks
- sa_report_deadlocks
- sa_snapshots
- sa_transactions
- sa_performance_statistics
- sa_performance_diagnostics
- sa_describe_shapefile
- sa_text_index_stats
- sa_get_user_status
- xp_getenv

このセクションの内容:

データベースが使用するセキュリティモデルの決定 (SQL の場合) [95 ページ]
データベースの作成時またはアップグレード時に指定されたセキュリティモデル設定 (呼び出し側と定義者) を、Capabilities データベースプロパティのクエリによって取得します。

1.2.2.1.1.1 データベースが使用するセキュリティモデルの決定 (SQL の場合)

データベースの作成時またはアップグレード時に指定されたセキュリティモデル設定 (呼び出し側と定義者) を、Capabilities データベースプロパティのクエリによって取得します。

コンテキスト

デフォルトでは、新しいデータベースは INVOKER モデルのみを使用して権限付きシステムプロシージャを実行します。これは、権限付き操作を実行する 16.0 より前のバージョンのシステムプロシージャが、ユーザがプロシージャを呼び出した権限で実行されることを意味します。この設定は、データベースの作成時またはアップグレード時に変更できます。この方法を使用して、指定したセキュリティモデル設定 (呼び出し側と定義者) を決定できます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

Interactive SQL で、データベースにログインし、次の SQL 文を実行します。

```
SELECT IF ((HEXTOINT(SUBSTRING(DB_PROPERTY('Capabilities'),
1,LENGTH(DB_PROPERTY('Capabilities'))-20)) & 8) = 8)
THEN 1
ELSE 0
END IF
```

結果

A 1 は、権限付き操作を実行する 16.0 より前のバージョンのシステムプロシージャが、呼び出し側モデルを使用して実行されたことを示します。A 0 は、プロシージャが定義者 (所有者) の権限で実行されたことを示します。

1.2.2.2 プロシージャの作成 (SQL Central の場合)

プロシージャテンプレートを使用してプロシージャを作成するには、プロシージャ作成ウィザードを使用します。

前提条件

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。

外部プロシージャを作成するには、CREATE EXTERNAL REFERENCE システム権限も必要です。

テンポラリプロシージャの作成には権限は必要ありません。

手順

1. SQL Central で、*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. 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。
3. ファイル > 新規 > プロシージャ をクリックします。
4. プロシージャ作成ウィザードの指示に従います。
5. 右ウィンドウ枠で、*[SQL]* タブをクリックして、プロシージャコードを終了します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

新しいプロシージャは、*[プロシージャとファンクション]* に表示されます。このプロシージャはアプリケーションで使用できます。

関連情報

複合文 [126 ページ]
リモートサーバとリモートテーブルのマッピング [663 ページ]
リモートプロシージャの作成 (SQL Central の場合) [696 ページ]

1.2.2.3 プロシージャの変更 (SQL Central の場合)

SQL Central で既存のプロシージャを変更します。

前提条件

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

コンテキスト

SQL Central では、既存のプロシージャの名前を直接変更することはできません。新しい名前のプロシージャを新しく作成し、以前のコードをそこへコピーしてから、元のプロシージャを削除します。

手順

1. SQL Central で、*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. 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。
3. プロシージャを選択します。
4. 次のいずれかの方法を使用して、プロシージャを編集します。
    - 右ウィンドウ枠で、*[SQL]* タブをクリックします。
    - プロシージャを右クリックして、*[新しいウィンドウで編集]* をクリックします。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

> ➡ ヒント
>
> プロシージャごとに別のウィンドウを開いて、プロシージャ間でコードをコピーできます。

- ○ プロシージャのコメントを追加または編集するには、プロシージャを右クリックして、*[プロパティ]* をクリックします。
  データベースドキュメントウィザードを使用して、SQL Anywhere データベースをドキュメント化する場合、これらのコメントを出力に含めることができます。

結果

プロシージャのコードが変更されます。

関連情報

プロシージャの作成 (SQL Central の場合) [96 ページ]
ストアドプロシージャの変換 [547 ページ]

1.2.2.4 プロシージャの呼び出し (SQL の場合)

プロシージャを呼び出し、値を挿入します。

前提条件

そのプロシージャの所有者であるか、そのプロシージャに対する EXECUTE 権限を持っているか、または EXECUTE ANY PROCEDURE システム権限を持っている必要があります。

プロシージャの EXECUTE 権限を付与されたすべてのユーザは、テーブルの権限がなくてもプロシージャを呼び出すことができます。

コンテキスト

プロシージャの呼び出しには CALL 文を使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

次の文を実行してプロシージャを呼び出し、次の値を挿入します。

```
CALL procedure-name( values );
```

この呼び出しの後で、値が追加されたことを確認することもできます。

> 注記
>
> クエリで呼び出すことで結果セットを返すプロシージャを呼び出すことができます。プロシージャの結果セットに対してクエリを実行し、WHERE 句やその他の SELECT 機能を適用して、結果セットを制限できます。

結果

プロシージャが呼び出され、実行されます。

> 例
>
> 次に、NewDepartment プロシージャを呼び出して、部署 Eastern Sales を追加する例を示します。
>
> ```
> CALL NewDepartment( 210, 'Eastern Sales', 902 );
> ```
>
> この呼び出しの後で、実際に新しく部署が追加されたことを確認するために、Departments テーブルを確認できます。
>
> プロシージャの EXECUTE 権限を付与されたすべてのユーザは、Department テーブルの権限がなくても NewDepartment プロシージャを呼び出すことができます。

1.2.2.5 プロシージャのコピー (SQL Central の場合)

SQL Central を使用して、データベース間または同じデータベース内でプロシージャをコピーします。

前提条件

プロシージャをコピーして自分自身を所有者として割り当てる場合は、プロシージャのコピー先のデータベースで、CREATE PROCEDURE システム権限を持っていることが必要です。プロシージャをコピーして別のユーザを所有者として割り当てる場合は、プロシージャのコピー先のデータベースで、CREATE ANY PROCEDURE または CREATE ANY OBJECT のシステム権限を持っていることが必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

同じデータベース内でプロシージャをコピーする場合は、プロシージャの名前を変更するか、コピーしたプロシージャに別の所有者を選択する必要があります。

手順

1. SQL Central で *SQL Anywhere17* プラグインを使用すると、コピーするプロシージャを含むデータベースに接続できます。
2. プロシージャをコピーするデータベースに接続します。
3. 最初のデータベースの左ウィンドウ枠でコピーするプロシージャを選択して、それを 2 番目のデータベースの *[プロシージャとファンクション]* にドラッグします。

結果

新しいプロシージャが作成されて、元のプロシージャのコードがコピーされます。新しいプロシージャにコピーされるのは、プロシージャのコードだけです。権限など、その他のプロシージャのプロパティはコピーされません。

1.2.2.6 プロシージャの削除 (SQL Central の場合)

プロシージャが必要なくなったときなどに、データベースから削除します。

前提条件

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システム権限を持っていることが必要です。

- DROP ANY PROCEDURE
- DROP ANY OBJECT

手順

1. SQL Central で、*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. 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。
3. プロシージャを右クリックして、*[削除]* をクリックします。
4. はいをクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

プロシージャがデータベースから削除されます。

次のステップ

従属データベースオブジェクトの場合は、その定義を変更して削除されたプロシージャへの参照を削除する必要があります。

1.2.3 ユーザ定義関数

ユーザ定義関数は、呼び出しを行った環境に単一の値を返します。

> **注記**
>
> データベースサーバはユーザ定義関数がスレッドセーフであるかどうかについて想定しません。この想定を行うことは、アプリケーション開発者の責任になります。

CREATE FUNCTION の構文は、CREATE PROCEDURE 文の構文と若干異なります。

- IN、OUT、INOUT などのキーワードは必要ありません。すべてのパラメータは IN パラメータです。
- 返されるデータ型を指定するために RETURNS 句が必要です。
- 返される値を指定するために RETURN 文が必要です。
- 名前付きパラメータはサポートされていません。

このセクションの内容:

ユーザ定義関数の作成 [102 ページ]
SQL Central を使用して、ユーザ定義ファンクションを作成します。

ユーザ定義関数の呼び出し [103 ページ]
Interactive SQL を使用して、ユーザ定義の関数を作成します。

ユーザ定義関数の削除 (SQL の場合) [105 ページ]
ユーザ定義関数を削除します。

ユーザ定義関数実行機能の付与 (SQL の場合) [106 ページ]
EXECUTE オブジェクトレベル権限を付与して、ユーザ定義関数を実行する機能を付与します。

ユーザ定義関数に関する詳細情報 [107 ページ]
データベースサーバでは、すべてのユーザ定義関数は、NOT DETERMINISTIC と宣言されないかぎりべき等として扱われます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.3.1 ユーザ定義関数の作成

SQL Central を使用して、ユーザ定義ファンクションを作成します。

前提条件

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 システム権限が必要です。

テンポラリ関数の作成には権限は必要ありません。

コンテキスト

ユーザ定義関数はプロシージャの集まりで、呼び出しを行った環境に単一の値を返します。

手順

1. SQL Central で、*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. 左ウィンドウ枠でプロシージャとファンクションを右クリックし、新規 > ファンクション をクリックします。
3. ファンクション作成ウィザードの指示に従います。
4. 右ウィンドウ枠で、*[SQL]* タブをクリックして、ファンクションコードを終了します。

結果

新しいファンクションは、*[プロシージャとファンクション]* に表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.3.2 ユーザ定義関数の呼び出し

Interactive SQL を使用して、ユーザ定義の関数を作成します。

前提条件

その関数に対する EXECUTE 権限が必要です。

コンテキスト

ユーザ定義関数は、集合関数以外の組み込み関数が使われていればどこでも使用できます。

手順

1. Interactive SQL で、データベースに接続します。
2. ユーザ定義関数を使用して、EXECUTE 文を実行します。

結果

関数が呼び出され、実行されます。

例

例 1: ユーザ定義関数の呼び出し

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;
```

Interactive SQL で次の文を実行すると、姓と名前の入った 2 つのカラムから氏名が返されます。

```
SELECT FullName( GivenName, Surname )
 AS "Full Name"
 FROM Employees;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Full Name
Fran Whitney
Matthew Cobb
Philip Chin
...

Interactive SQL で次の文を実行すると、FullName ユーザ定義関数が使用され、指定された姓と名前から氏名が返されます。

```
SELECT FullName('Jane', 'Smith')
 AS "Full Name";
```

Full Name
Jane Smith

例 2: ローカル変数の宣言

次にローカル変数の宣言の例としてユーザ定義関数を示します。

i 注記

この関数は説明には役立ちますが、多数のローを含む SELECT に使用する場合は、性能が低くなることがあります。たとえば、テーブルに 100000 のローがあり、その中の 10000 のローを返すようなクエリの SELECT リストで関数を使用した場合、関数は 10000 回呼び出されます。同じクエリの WHERE 句に関数を使用した場合は、100000 回呼び出されます。

Customers テーブルには、カナダとアメリカの顧客が含まれます。ユーザ定義関数 Nationality は、Country カラムの入力データに基づいて 3 文字の国コードを生成します。

```
CREATE FUNCTION Nationality( CustomerID INT )
RETURNS CHAR( 3 )
BEGIN
    DECLARE nation_string CHAR(3);
    DECLARE nation country_t;
    SELECT DISTINCT Country INTO nation
    FROM Customers
    WHERE ID = CustomerID;
    IF nation = 'Canada' THEN
            SET nation_string = 'CDN';
    ELSE IF nation = 'USA' OR nation = ' ' THEN
            SET nation_string = 'USA';
        ELSE
            SET nation_string = 'OTH';
        END IF;
    END IF;
RETURN ( nation_string );
END;
```

この例では国名を入れる変数 nation_string を宣言し、SET 文を使用して値を nation_string に入れます。次に nation_string の値を、この関数を呼び出した環境に返します。

次に示すクエリは、Customers テーブルに含まれるカナダの顧客をすべてリストします。

```
SELECT *
FROM Customers
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
WHERE Nationality( ID ) = 'CDN';
```

1.2.3.3 ユーザ定義関数の削除 (SQL の場合)

ユーザ定義関数を削除します。

前提条件

ユーザ定義関数の所有者であるか、または次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。

- DROP ANY PROCEDURE
- DROP ANY OBJECT

手順

1. データベースに接続します。
2. 次のような DROP FUNCTION 文を実行します。

```
DROP FUNCTION function-name;
```

結果

ユーザ定義関数が削除されます。

例

次に、FullName 関数をデータベースから削除する文を示します。

```
DROP FUNCTION FullName;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.3.4 ユーザ定義関数実行機能の付与 (SQL の場合)

EXECUTE オブジェクトレベル権限を付与して、ユーザ定義関数を実行する機能を付与します。

前提条件

ユーザ定義関数の所有者であるか、またはその関数の EXECUTE 権限に対する管理権限を持っていることが必要です。

ユーザ定義関数の所有権はその関数の作成者に属しており、そのユーザが関数を実行する場合には権限は必要ありません。

コンテキスト

関数を作成して、他のユーザがその関数を実行できるようにします。

手順

1. データベースに接続します。
2. 次のような GRANT EXECUTE 文を実行します。

```
GRANT EXECUTE ON function-name TO user-id;
```

結果

被付与者がプロシージャを実行できるようになりました。

例

たとえば、Nationality 関数の作成者が別のユーザに Nationality の使用許可を与える文は、次のようになります。

```
GRANT EXECUTE ON Nationality TO BobS;
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.3.5 ユーザ定義関数に関する詳細情報

データベースサーバでは、すべてのユーザ定義関数は、NOT DETERMINISTIC と宣言されないかぎりべき等として扱われます。

べき等関数は、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。同じパラメータを持つ冪等関数が連続して 2 回呼び出されている場合は、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副次効果は生じません。

関連情報

関数のキャッシュ [209 ページ]

1.2.4 トリガ

トリガとは、データを修正する文が実行されると自動的に実行されるストアドプロシージャの特別な形式です。

トリガは、参照整合性や他の宣言制約では不十分な場合に使います。

検査項目を細かく設定して複雑な参照整合性を設定したり、既存のデータは制約の範囲から外れても許可するが新しいデータはチェックしたりする場合があります。トリガはこのようなときに使用すると便利です。また、データベースにアクセスするアプリケーションとは個別に、データベーステーブルのアクティビティのログを取るときにもトリガを使います。

> **注記**
>
> その後のトリガが起動しない LOAD TABLE、TRUNCATE、WRITETEXT の 3 つの特別な文があります。

トリガを実行する権限

トリガは、関連するテーブルまたはビューの所有者の権限によって実行されます。そのトリガを起動したユーザの ID ではありません。トリガはユーザが直接変更できないテーブルのローを変更できます。

トリガが起動しないようにするには、-gf サーバオプションを指定するか、または fire_triggers オプションを設定します。

トリガのタイプ

サポートされるトリガのタイプは次のとおりです。

BEFORE トリガ

BEFORE トリガは、トリガ元アクションが実行される前に実行されます。BEFORE トリガはテーブルに定義できますが、ビューには定義できません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

AFTER トリガ

AFTER トリガは、トリガ元アクションが完了した後に実行されます。AFTER トリガはテーブルに定義できますが、ビューには定義できません。

INSTEAD OF トリガ

INSTEAD OF トリガは、トリガ元アクションの代わりに実行される条件付きのトリガです。INSTEAD OF トリガはテーブルとビューに定義できます (マテリアライズドビューを除く)。

トリガイベント

トリガを起動するイベントのリストを次に示します。

動作	説明
INSERT	トリガの関連するテーブルに新しいローが挿入されたときに、トリガが起動されます。
DELETE	トリガの関連するテーブル内のローが削除されたときに、トリガが起動されます。
UPDATE	トリガの関連するテーブル内のローが更新されたときに、トリガが起動されます。
UPDATE OF `column-list`	トリガの関連するテーブル内のローが、*column-list* 中のカラムが変更されるなどして更新されたときに、トリガが起動されます。

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。

トリガのタイミング

トリガのレベルには、ローレベルと文レベルがあります。

- ローレベルトリガは、変更されるローごとに一回実行されます。ローレベルトリガは、ローの変更前または変更後に実行されます。
  対象ローの新しいイメージと古いイメージのカラム値は、変数によってトリガから使用可能になります。
- 文レベルトリガは、トリガする文全体の処理が完了した後に実行されます。トリガする文の対象ローは、ローの新しいイメージと古いイメージを表すテンポラリテーブルによってトリガから使用可能になります。SQL Anywhere では、文レベル BEFORE トリガはサポートされていません。

トリガ実行のタイミングは柔軟に設定できるので、実行に応じてカスケード更新または削除の実行が決まるような、参照整合性に依存するトリガに対して有効です。

トリガの実行中にエラーが発生すると、トリガを起動した操作そのものがエラーになります。INSERT、UPDATE、DELETE はアトミックオペレーションです。これらがエラーになると、トリガの結果とトリガが起動したプロシージャを含め、その文のすべての結果がキャンセルされます。

このセクションの内容:

テーブル内でのトリガの作成 (SQL Central の場合) [109 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

トリガ作成ウィザードを使用して、テーブル内にトリガを作成します。

テーブル内でのトリガの作成 (SQL の場合) [110 ページ]
CREATE TRIGGER 文を使用して、テーブル内にトリガを作成します。

トリガの実行 [112 ページ]
指定したテーブルで INSERT、UPDATE、DELETE が行われたときに、トリガが自動的に実行されます。

トリガの変更 [113 ページ]
SQL Central を使用して、トリガの定義を編集します。

トリガの削除 [114 ページ]
SQL Central を使用して、データベースからトリガを削除します。

例: トリガ操作の一時的な無効化 [115 ページ]
ユーザがカラムデータに対して (トリガを起動する) アクションを実行するときに、トリガの操作が一時的に無効になるようにトリガを設定できます。

トリガを実行する権限 [117 ページ]
ユーザはトリガを実行できません。データベースに対するアクションに対応してデータベースサーバがトリガを起動します。

トリガに関する詳細情報 [117 ページ]
競合するトリガが実行されるかどうか、また実行される場合の順序は、トリガのタイプ (BEFORE、INSTEAD OF、または AFTER) とトリガのスコープ (ローレベルまたは文レベル) の 2 点で決まります。

関連情報

データ整合性 [727 ページ]
INSTEAD OF トリガ [118 ページ]
アトミックな複合文 [127 ページ]

1.2.4.1 テーブル内でのトリガの作成 (SQL Central の場合)

トリガ作成ウィザードを使用して、テーブル内にトリガを作成します。

前提条件

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

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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. 左ウィンドウ枠でトリガを右クリックし、 新規 トリガ をクリックします。
3. トリガ作成ウィザードの指示に従います。
4. コードを完了するには、右ウィンドウ枠で、*[SQL]* タブをクリックします。

結果

新しいトリガが作成されます。

関連情報

複合文 [126 ページ]

1.2.4.2 テーブル内でのトリガの作成 (SQL の場合)

CREATE TRIGGER 文を使用して、テーブル内にトリガを作成します。

前提条件

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

コンテキスト

COMMIT と ROLLBACK 文、いくつかの ROLLBACK TO SAVEPOINT 文をトリガ内に使用することはできません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. データベースに接続します。
2. CREATE TRIGGER 文を実行します。

   トリガの本文は複合文、つまり BEGIN 文と END 文に挟まれ、セミコロンで区切られた SQL 文のセットから構成されています。

結果

新しいトリガが作成されます。

例

例 1: ローレベルの INSERT トリガ

次にローレベルの INSERT トリガの例を示します。新しい従業員の生年月日が正しく入力されたかどうかをチェックします。

```
CREATE TRIGGER check_birth_date
   AFTER INSERT ON Employees
REFERENCING NEW AS new_employee
FOR EACH ROW
BEGIN
     DECLARE err_user_error EXCEPTION
   FOR SQLSTATE '99999';
   IF new_employee.BirthDate > 'June 6, 2001' THEN
          SIGNAL err_user_error;
   END IF;
END;
```

注記

SQL Anywhere のサンプルデータベースに check_birth_date というトリガがすでにある可能性があります。このトリガがある場合に上記の SQL 文を実行しようとすると、トリガの定義が既存のトリガと矛盾していることを示すエラーが表示されます。

このトリガは、Employees テーブルに新しいローが追加されると起動されます。2001 年 6 月 6 日以降の生年月日に対応する新しいローを検知し、エラーにします。

フレーズ REFERENCING NEW AS new_employee は、トリガコード中の文がエイリアス new_employee を使用して、新しいローのデータを参照できるようにします。

エラーが発生すると、トリガ元の文、およびトリガによる前の変更内容がすべて取り消されます。

Employees テーブルに複数のローを追加する INSERT 文の場合は、新しいローごとに check_birth_date トリガが起動されます。どれか 1 つのローでトリガが失敗すると、INSERT 文のすべての結果がロールバックされます。

ローを追加した後でなく、追加する前にトリガが起動されるようにするには、例文の 2 行目を次のように変更します。

```
BEFORE INSERT ON Employees
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

REFERENCING NEW 句は追加されるローの値を参照します。この句はトリガが起動されるタイミング (BEFORE と AFTER) には影響されません。

トリガではなく、宣言参照整合性や検査制約を使用して整合性を確保する方が簡単な場合があります。たとえば、上記の例でカラム検査制約を使用すると、さらに効率が良く、簡潔になります。

```
CHECK (@col <= 'June 6, 2001')
```

例 2: ローレベルの **DELETE** トリガの例

次に示す CREATE TRIGGER 文は、ローレベルの DELETE トリガを定義します。

```
CREATE TRIGGER mytrigger
BEFORE DELETE ON Employees
REFERENCING OLD AS oldtable
FOR EACH ROW
BEGIN
   ...
END;
```

REFERENCING OLD 句は、トリガが起動されるタイミング (BEFORE または AFTER) に影響されず、エイリアス oldtable を使用して、削除されるローの値を削除トリガコードが参照できるようにします。

例 3: 文レベルの **UPDATE** トリガの例

文レベルの UPDATE トリガを作成する CREATE TRIGGER 文の例を次に示します。

```
CREATE TRIGGER mytrigger AFTER UPDATE ON Employees
REFERENCING NEW AS table_after_update
            OLD AS table_before_update
FOR EACH STATEMENT
BEGIN
   ...
END;
```

REFERENCING NEW 句と REFERENCING OLD 句は、UPDATE トリガのコマンド文が更新の前と後の両方の値を参照できるようにします。テーブルエイリアス table_after_update は、新しいローのカラムを参照し、テーブルエイリアス table_before_update は古いローのカラムを参照します。

REFERENCING NEW 句と REFERENCING OLD 句は、文レベルとローレベルのトリガで少し異なる意味を持ちます。文レベルではテーブルが対象になりますが、ローレベルでは変更されるローが対象になります。

関連情報

複合文 [126 ページ]

1.2.4.3 トリガの実行

指定したテーブルで INSERT、UPDATE、DELETE が行われたときに、トリガが自動的に実行されます。

ローレベルトリガは、ローが影響を受けるごとに起動され、文レベルトリガは、文全体が一度に起動されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

INSERT、UPDATE、DELETE がトリガを起動すると、トリガのタイプ (BEFORE または AFTER) によって、次の順序で操作が行われます。

1. BEFORE トリガが起動します。
2. 追加などの操作そのものが実行されます。
3. 参照動作を行います。
4. AFTER トリガが起動します。

i 注記

CREATE TRIGGER 文を使用してトリガを作成するときにトリガのタイプを指定しなかった場合は、デフォルトで AFTER になります。

手順の途中で、プロシージャまたはトリガの内部で処理されないエラーが発生すると、それより以前の手順は取り消され、それ以降の手順は実行されません。トリガを起動した操作そのものも失敗となります。

1.2.4.4 トリガの変更

SQL Central を使用して、トリガの定義を編集します。

前提条件

コメントを追加または編集するには、次のいずれかのシステム権限を持っている必要があります。

- COMMENT ANY OBJECT
- ALTER ANY TRIGGER
- ALTER ANY OBJECT
- CREATE ANY TRIGGER
- CREATE ANY OBJECT

コードを編集するには、ALTER ANY OBJECT システム権限または ALTER ANY TRIGGER システム権限および次の条件の 1 つを持っている必要があります。

- 基礎となるテーブルの所有者です。
- ALTER ANY TABLE システム権限
- 基礎となるテーブルに対する ALTER 権限

コンテキスト

SQL Central では、既存のトリガの名前を直接変更することはできません。代わりに、新しい名前を付けて新しくトリガを作成し、このトリガに以前のコードをコピーしてから、元のトリガを削除します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. SQL Central で、*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. 左ウィンドウ枠で、トリガをダブルクリックします。
3. トリガを選択します。
4. 次のいずれかの方法を使用して、トリガを変更します。

オプション	アクション
コードの編集	トリガを右クリックして、*[新しいウィンドウで編集]* をクリックするか、右ウィンドウ枠の *[SQL]* タブでコードを編集できます。 ➜ ヒント プロシージャごとに別のウィンドウを開いて、トリガ間でコードをコピーできます。
コメントの追加	トリガのコメントを追加または編集するには、トリガを右クリックして、*[プロパティ]* をクリックします。 データベースドキュメントウィザードを使用して、SQL Anywhere データベースをドキュメント化する場合、これらのコメントを出力に含めることができます。

結果

トリガのコードが変更されます。

関連情報

ストアドプロシージャの変換 [547 ページ]

1.2.4.5 トリガの削除

SQL Central を使用して、データベースからトリガを削除します。

前提条件

そのトリガの所有者であるか、または次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。

- DROP ANY TRIGGER
- DROP ANY OBJECT

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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. 左ウィンドウ枠で、トリガをダブルクリックします。
3. トリガを選択して、編集 > 削除 をクリックします。
4. はい をクリックします。

結果

トリガがデータベースから削除されます。

次のステップ

従属データベースオブジェクトの場合は、その定義を変更して削除されたトリガへの参照を削除する必要があります。

1.2.4.6 例: トリガ操作の一時的な無効化

ユーザがカラムデータに対して (トリガを起動する) アクションを実行するときに、トリガの操作が一時的に無効になるようにトリガを設定できます。

その場合でも、事前定義の接続変数を含むプロシージャを使用して、トリガを起動してその操作を実行できます。ユーザは、トリガが起動されてもトリガ操作を実行しないで、カラムに対して INSERT、ALTER、または DELETE を実行できます。

注記

ローレベルのトリガを使用している場合は、WHEN 句を使用して、トリガをいつ起動するかを指定します。

例

例: 1 つのトリガの操作の一時的な無効化

この例では、接続変数が存在するかどうかに基づいてトリガの操作を無効にします。

1. 接続変数の状態を確認してトリガ論理が有効かどうかを判断する AFTER INSERT トリガを作成します。変数が存在しない場合、トリガの操作は有効です。

```
CREATE TRIGGER myTrig AFTER INSERT
REFERENCING NEW AS new-name
FOR EACH STATEMENT
BEGIN
   DECLARE @execute_trigger integer;
   IF varexists('enable_trigger_logic') = 1 THEN
     SET @execute_trigger = enable_trigger_logic;
   ELSE
     SET @execute_trigger = 1;
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    END IF;
    IF @execute_trigger = 1 THEN
       ... -your-trigger-logic
    END IF;
END;
```

2. 次のコードを文に追加し、手順 1 で作成したトリガを呼び出します。文は、接続変数を使用してトリガをいつ無効にするかを制御します。文は無効にするコードを囲んでいる必要があります。

```
...
  IF varexists('enable_trigger_logic') = 0 THEN
       CREATE VARIABLE enable_trigger_logic INT;
  END IF;
  SET enable_trigger_logic = 0;
   ... execute-your-code-that-you-do-not-want-triggers-to-run
   SET enable_trigger_logic = 1;
   ... now-your-trigger-logic-will-do-its-work
```

例: 複数のトリガの操作を一時的に無効にする

この例では、例 1 の接続変数の方法を使用して、複数のトリガの操作を制御します。複数のトリガを有効および無効にするために呼び出すことができる 2 つのプロシージャを作成します。トリガ操作が有効かどうかを確認するために使用することができる関数も作成します。

1. トリガ操作を無効にするために呼び出すことができるプロシージャを作成します。動作は接続変数の値に基づいています。

```
CREATE PROCEDURE sp_disable_triggers()
BEGIN
       IF VAREXISTS ('enable_trigger_logic') = 0 THEN
          CREATE VARIABLE enable_trigger_logic INT;
       END IF;
       SET enable_trigger_logic = 0;
END;
```

2. トリガ操作を有効にするために呼び出すことができるプロシージャを作成します。動作は接続変数の値に基づいています。

```
CREATE PROCEDURE sp_enable_triggers()
BEGIN
        IF VAREXISTS ('enable_trigger_logic') = 0 THEN
          CREATE VARIABLE enable_trigger_logic INT;
        END IF;
      SET enable_trigger_logic = 1;
END;
```

3. トリガ操作が有効かどうかを判別するために呼び出すことができる関数を作成します。

```
CREATE FUNCTION f_are_triggers_enabled()
RETURNS INT
BEGIN
       IF VAREXISTS ('enable_trigger_logic') = 1 THEN
         RETURN enable_trigger_logic;
       ELSE
          RETURN 1;
       END IF;
END;
```

4. 操作を制御するトリガに IF 句を追加します。

```
IF f_are_triggers_enabled() = 1 THEN
     ... your-trigger-logic
END IF;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

5. トリガ操作を有効にするために手順 2 で作成したプロシージャを呼び出します。

```
CALL sp_enable_triggers();
... execute-code-where-trigger-logic-runs
```

6. トリガ操作を無効にするために手順 1 で作成したプロシージャを呼び出します。

```
CALL sp_disable_triggers();
... execute-your-code-where-trigger-logic-is-disabled
```

1.2.4.7 トリガを実行する権限

ユーザはトリガを実行できません。データベースに対するアクションに対応してデータベースサーバがトリガを起動します。

トリガが実行される場合は、トリガに関連する権限があり、その動作を実行する権利を定義します。

トリガは、トリガが定義されているテーブルの所有者の権限を使用して実行します。トリガを起動する原因となったユーザの権限や、トリガを作成したユーザの権限ではありません。

トリガがテーブルを参照するときは、そのテーブルの所有者名を特に指定しないで、テーブル作成者のロールメンバーシップを使います。たとえば、user_1.Table_A にあるトリガが Table_B を参照し、Table_B の所有者の名前を指定しないとします。この場合、Table_B が user_1 によって作成されたか、user_1 が Table_B の所有者であるロールの (直接または間接的に) メンバーでなければなりません。どちらの条件も満たされない場合は、トリガを起動すると、データベースサーバがテーブルが見つからないことを示すメッセージを返します。

また、user_1 はトリガに指定された操作を実行するための権限を持っていなければなりません。

1.2.4.8 トリガに関する詳細情報

競合するトリガが実行されるかどうか、また実行される場合の順序は、トリガのタイプ (BEFORE、INSTEAD OF、または AFTER) とトリガのスコープ (ローレベルまたは文レベル) の 2 点で決まります。

UPDATE 文は、複数のテーブルのカラム値を変更できます。トリガ起動の順序は各テーブルで同じですが、テーブルが更新される順序は保証されません。

ローレベルのトリガの場合、BEFORE トリガが実行されてから INSTEAD OF トリガが実行され、その後に AFTER トリガが実行されます。特定のローのローレベルのトリガがすべて実行されてから、後続のローのトリガが実行されます。

文レベルのトリガの場合、INSTEAD OF トリガが実行されてから AFTER トリガが実行されます。文レベルの BEFORE トリガはサポートされていません。

文レベルとローレベルの AFTER トリガが競合する場合は、ローレベルの AFTER トリガがすべて完了してから文レベルのトリガが実行されます。

文レベルとローレベルの INSTEAD OF トリガが競合する場合、ローレベルのトリガは実行されません。

AFTER STATEMENT トリガに対して作成される OLD および NEW テンポラリテーブルは、基本となるベーステーブルと同じスキーマを持ち、カラム名とデータ型は同じです。ただし、これらのテーブルにはプライマリキー、外部キー、またはインデックスはありません。OLD および NEW テンポラリテーブル内のローの順序は保証されず、ベーステーブルのローが元々更新された順序と一致しない可能性があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

INSTEAD OF トリガ [118 ページ]
INSTEAD OF トリガは、トリガが実行されるとトリガ元アクションはスキップされ、代わりに指定されたアクションが実行される点で、BEFORE トリガや AFTER トリガと異なります。

1.2.4.8.1 INSTEAD OF トリガ

INSTEAD OF トリガは、トリガが実行されるとトリガ元アクションはスキップされ、代わりに指定されたアクションが実行される点で、BEFORE トリガや AFTER トリガと異なります。

INSTEAD OF トリガに固有の機能や制限を次に示します。

- INSTEAD OF トリガは、特定のテーブルのトリガイベントごとに 1 つだけ指定できます。
- INSTEAD OF トリガはテーブルまたはビューに定義できます。ただし、INSTEAD OF トリガはマテリアライズドビューには定義できません。マテリアライズドビューには INSERT 文、DELETE 文、UPDATE 文などの DML 操作を実行できないからです。
- INSTEAD OF トリガを定義するときは、ORDER 句または WHEN 句は指定できません。
- INSTEAD OF トリガは、UPDATE OF `column-list` トリガイベントには定義できません。
- INSTEAD OF トリガが再帰を実行するかどうかは、トリガのターゲットがベーステーブルであるか、ビューであるかで異なります。ビューの場合は再帰が発生しますが、ベーステーブルの場合は発生しません。たとえば、INSTEAD OF トリガによって、トリガが定義されているベーステーブルに対して DML 操作が実行されても、これらの操作でトリガは実行されません (BEFORE トリガや AFTER トリガを含む)。ターゲットがビューの場合は、ビューに対して実行された操作ですべてのトリガが実行されます。
- テーブルに INSTEAD OF トリガが定義されている場合、ON EXISTING 句を含む INSERT 文をテーブルに対して実行できません。実行しようとすると、SQLE_INSTEAD_TRIGGER エラーが返されます。
- WITH CHECK OPTION を指定して定義されているか、WITH CHECK OPTION を指定して定義されている別のビューにネストされており、INSTEAD OF INSERT トリガが定義されているビューには、INSERT 文を実行できません。UPDATE 文と DELETE 文も同様です。実行しようとすると、SQLE_CHECK_TRIGGER_CONFLICT エラーが返されます。
- 位置付け更新、位置付け削除、PUT 文、またはワイド挿入操作の結果として INSTEAD OF トリガが実行されると、SQLE_INSTEAD_TRIGGER_POSITIONED エラーが返されます。

INSTEAD OF トリガを使用した更新不可のビューの更新

INSTEAD OF トリガを使用すると、本質的には更新不可能なビューに INSERT 文、UPDATE 文、または DELETE 文を実行できます。トリガの本文で、対応する INSERT、UPDATE、DELETE 文を実行する意味を定義します。たとえば、次のビューを作成するとします。

```
CREATE VIEW V1 ( Surname, GivenName, State )
   AS SELECT DISTINCT Surname, GivenName, State
         FROM Contacts;
```

DISTINCT キーワードによって、派生の関係で V1 が更新不可能になるので、V1 のローは削除できません。つまり、データベースサーバでは、V1 からローを削除する意味を明確に特定できません。ただし、V1 への削除操作を実装する INSTEAD OF

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

DELETE トリガを定義することはできます。たとえば、次のトリガでは、V1 からローを削除すると、Surname、GivenName、State が指定されている Contacts からすべてのローが削除されます。

```
CREATE TRIGGER V1_Delete
  INSTEAD OF DELETE ON V1
  REFERENCING OLD AS old_row
  FOR EACH ROW
BEGIN
    DELETE FROM Contacts
      WHERE Surname = old_row.Surname
        AND GivenName = old_row.GivenName
        AND State = old_row.State
END;
```

V1_Delete トリガを定義すると、V1 からローを削除できます。さらに、V1 で INSERT 文と UPDATE 文を実行させる他の INSTEAD OF トリガを定義することもできます。

INSTEAD OF DELETE トリガが定義されたビューが別のビューにネストされている場合は、DELETE に対する更新可能性を確認するためにベーステーブルのように処理されます。INSERT 操作と UPDATE 操作も同様です。前の例の続きで、別のビューを作成します。

```
CREATE VIEW V2 ( Surname, GivenName ) AS
  SELECT Surname, GivenName from V1;
```

V1_Delete トリガがなければ、V2 からローを削除することはできません。これは、派生の関係で V1 は更新不可能なので、V2 も更新不可能になるからです。しかし、V1 に INSTEAD OF DELETE トリガを定義すれば、V2 からローを削除できます。V2 からローが削除されるたびに V1 からローが削除され、V1_Delete トリガが実行されます。

INSTEAD OF トリガをネストされているビューに定義する場合は、これらのトリガが実行されるときに、意図しない影響がある可能性があるため、注意してください。意図する動作を明示的にするには、ネストされているビューを参照するすべてのビューに INSTEAD OF トリガを定義します。

次のトリガを V2 に定義し、DELETE 文の意図した動作を実行できます。

```
CREATE TRIGGER V2_Delete
  INSTEAD OF DELETE ON V2
  REFERENCING OLD AS old_row
  FOR EACH ROW
BEGIN
    DELETE FROM Contacts
      WHERE Surname = old_row.Surname
        AND GivenName = old_row.GivenName
END;
```

V2_Delete トリガによって、V1 に対する INSTEAD OF DELETE トリガが削除または変更されても、V2 に対する削除操作の動作は変わりません。

1.2.5 バッチ

バッチはまとめて送信されてグループとして次々に実行される一連の SQL 文です。

プロシージャで使われる制御文 (CASE、IF、LOOP など) はバッチでも使えます。バッチが BEGIN/END で囲まれた複合文で構成される場合、ホスト変数、変数のローカル宣言、カーソル、テンポラリテーブル、例外をバッチに含めることもできます。ホスト変数参照は、次の制限付きでバッチ内で使用できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ホスト変数を参照できるのはバッチ内の 1 文だけです。
- ホスト変数を使用する文の前に、結果セットを返す文を入れることはできません。

バッチの使用を明確に示すために、BEGIN/END を使うことをお奨めします。

バッチ内の文はセミコロンで区切ることができます。その場合、バッチは Watcom SQL 構文に準拠します。文を区切るためにセミコロンを使用しない複数文のバッチは、Transact-SQL 構文に準拠します。バッチの構文によって、バッチ内で使用できる文とバッチ内でのエラーの処理方法が決まります。

多くの点で、バッチはストアドプロシージャに似ていますが、いくつかの違いがあります。

- バッチには名前がありません。
- バッチにはパラメータを使用できません。
- バッチは永続的にデータベースに保存されません。
- バッチは異なる接続で共有できません。

簡単なバッチは、デリミタのない SQL 文のセットで、次の行に go という単語が続きます。次の例では、Eastern Sales という部署を作成し、Massachusetts のすべての営業担当者を Eastern Sales に転送します。これは Transact-SQL バッチの例です。

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' )
UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA'
COMMIT
go
```

go という単語は Interactive SQL によって認識され、前の文は 1 つのバッチとしてサーバに送信されます。

次の例は外見は似ていますが、Interactive SQL での処理は全く異なります。この例では、Transact-SQL 構文を使用しません。各文はセミコロンで区切られています。Interactive SQL はセミコロンで区切られた各文を個別にサーバに送信します。この場合は、バッチとしては処理されません。

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' );
UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA';
COMMIT;
```

Interactive SQL でバッチとして処理するには、`BEGIN ... END` を使用して、複合文に変更します。次の構文は、前の例を修正したものです。複合文に含まれる 3 つの文は、バッチとしてサーバに送信されます。

```
BEGIN
  INSERT
  INTO Departments ( DepartmentID, DepartmentName )
  VALUES ( 220, 'Eastern Sales' );
  UPDATE Employees
  SET DepartmentID = 220
  WHERE DepartmentID = 200
  AND State = 'MA';
  COMMIT;
END
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

この例の場合、サーバがバッチと個別のどちらで文を実行しても結果は同じになります。ただし、結果が異なる場合もあります。次の例を考えます。

```
DECLARE @CurrentID INTEGER;
SET @CurrentID = 207;
SELECT Surname FROM Employees
  WHERE EmployeeID=@CurrentID;
```

Interactive SQL を使用してこの例を実行すると、データベースサーバが変数が見つからないことを示すエラーを返します。このエラーは、Interactive SQL が 3 つの文を個別にサーバに送信するために発生します。これらの文はバッチとしては実行されません。このようなエラーに対処するには、複合文を使用して Interactive SQL が強制的に 3 つの文をバッチとしてサーバに送信するようにします。次の例では、複合文を使用しています。

```
BEGIN
  DECLARE @CurrentID INTEGER;
  SET @CurrentID = 207;
  SELECT Surname FROM Employees
    WHERE EmployeeID=@CurrentID;
END
```

一連の文を BEGIN と END で囲んだ場合、Interactive SQL は強制的に文をバッチとして処理します。

IF 文は複合文の別の例です。Interactive SQL は、次の文を 1 つのバッチとしてサーバに送信します。

```
IF EXISTS(   SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
ELSE
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF
```

別の方法で SQL 文を作成して実行した場合、この例の動作は適用されません。たとえば、ODBC を使用するアプリケーションでは、セミコロンで区切られた一連の文をバッチとして作成および実行できます。

Interactive SQL の文とサーバ向けの SQL 文が混在している場合は、注意が必要です。次の例に、Interactive SQL の文と SQL 文が混在する場合に発生し得る問題を示します。この例では、Interactive SQL の OUTPUT 文が複合文に組み込まれているため、その他のすべての文と一緒にバッチとしてサーバに送信され、構文エラーが発生します。

```
IF EXISTS(   SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
   OUTPUT TO 'c:¥¥temp¥¥query.txt';
ELSE
   MESSAGE 'The Employees table does not exist'
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    TO CLIENT;
END IF
```

OUTPUT 文の正しい配置は、下記のとおりです。

```
IF EXISTS(   SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
    SELECT   Surname AS LastName,
            GivenName AS FirstName
    FROM Employees;
    SELECT Surname, GivenName
    FROM Customers;
    SELECT Surname, GivenName
    FROM Contacts;
ELSE
    MESSAGE 'The Employees table does not exist'
    TO CLIENT;
END IF;
OUTPUT TO 'c:¥¥temp¥¥query.txt';
```

関連情報

Transact-SQL のバッチ [546 ページ]

1.2.6 プロシージャ、トリガ、ユーザ定義関数の構造

プロシージャ、トリガ、ユーザ定義関数の本文は複合文で構成されています。

複合文は、一連の SQL 文を囲む BEGIN と END で構成されています。各文はセミコロンで区切ります。

このセクションの内容:

プロシージャのパラメータ宣言 [123 ページ]
プロシージャのパラメータは、CREATE PROCEDURE 文でリストとして記述します。

プロシージャにパラメータを渡す方法 [123 ページ]
ストアドプロシージャパラメータのデフォルト値は、CALL 文の 2 通りの形式のどちらでも使用できます。

関数にパラメータを渡す方法 [124 ページ]
ユーザ定義関数は CALL 文で呼び出すのではなく、組み込み関数と同じように使用します。

関連情報

複合文 [126 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.6.1　プロシージャのパラメータ宣言

プロシージャのパラメータは、CREATE PROCEDURE 文でリストとして記述します。

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。パラメータは有効なデータ型で、キーワード IN、OUT、INOUT のいずれかのプレフィクスが付いています。デフォルトでは、パラメータは INOUT パラメータです。これらのキーワードには、次のような意味があります。

IN

引数はプロシージャに値を提供する式です。

OUT

引数はプロシージャから値を与えられる変数です。

INOUT

引数はプロシージャに値を提供する変数で、プロシージャから新しい値を与えられることもあります。

CREATE PROCEDURE 文中のプロシージャパラメータにはデフォルト値を設定できます。デフォルト値は定数で、NULL でもかまいません。たとえば、次に示すプロシージャは、IN パラメータのデフォルトとして NULL を指定しています。これは意味のないクエリを実行するのを避けるためです。

```
CREATE PROCEDURE CustomerProducts( IN customer_ID INTEGER DEFAULT NULL )
RESULT ( product_ID INTEGER,
         quantity_ordered INTEGER )
BEGIN
   IF customer_ID IS NULL THEN
      RETURN;
   ELSE
      SELECT Products.ID, sum( SalesOrderItems.Quantity )
      FROM   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;
```

次に示す文は DEFAULT NULL を割り当て、プロシージャはクエリを実行せずに RETURN 操作を実行します。

```
CALL CustomerProducts();
```

1.2.6.2　プロシージャにパラメータを渡す方法

ストアドプロシージャパラメータのデフォルト値は、CALL 文の 2 通りの形式のどちらでも使用できます。

CREATE PROCEDURE 文の引数リストの末尾にオプションのパラメータがある場合、これらは CALL 文で省略できます。次に示すのは、INOUT パラメータを 3 つ持つプロシージャの例です。

```
CREATE PROCEDURE SampleProcedure(
       INOUT var1 INT DEFAULT 1,
                    INOUT var2 int DEFAULT 2,
                    INOUT var3 int DEFAULT 3 )
...
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の例では、プロシージャを呼び出す環境で、プロシージャに渡す数値を格納するための接続スコープ変数を 3 つ設定してあるものと想定しています。

```
CREATE VARIABLE V1 INT;
CREATE VARIABLE V2 INT;
CREATE VARIABLE V3 INT;
```

次に示すように、SampleProcedure プロシージャは最初のパラメータを指定するだけで呼び出せます。この場合、*var2* と *var3* にはデフォルト値が使用されます。

```
CALL SampleProcedure( V1 );
```

また、次のように、プロシージャは最初のパラメータには DEFAULT 値を使用して、2 番目のパラメータのみを指定することによっても呼び出せます。

```
CALL SampleProcedure( DEFAULT, V2 );
```

オプションの引数を使ってプロシージャを呼び出すよりも柔軟な方法は、パラメータに名前を付けて渡すという方法です。このとき、SampleProcedure プロシージャは次のように呼び出すことができます。

```
CALL SampleProcedure( var1 = V1, var3 = V3 );
```

または次のようになります。

```
CALL SampleProcedure( var3 = V3, var1 = V1 );
```

i 注記

プロシージャの呼び出し時に、INOUT および OUT パラメータに対してデータベーススコープ変数を使用することはできません。ただし、IN パラメータにデータベーススコープ変数を使用することは可能です。

1.2.6.3 関数にパラメータを渡す方法

ユーザ定義関数は CALL 文で呼び出すのではなく、組み込み関数と同じように使用します。

たとえば、次の例では FullName 関数を使用して従業員の名前を取り出します。

例

Interactive SQL で次のクエリを実行します。

```
SELECT FullName( GivenName, Surname ) AS Name
   FROM Employees;
```

次の結果が表示されます。

Name
Fran Whitney
Matthew Cobb

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Name
Philip Chin
Julie Jordan
...

注記

- デフォルトパラメータは呼び出し関数でも使用できます。ただしパラメータは、名前を付けて関数に渡すことはできません。
- パラメータは参照ではなく、値で渡されます。関数がそのパラメータの値を変更しても、その変更は関数を呼び出した環境には戻されません。
- ユーザ定義関数では出力パラメータは使用できません。
- ユーザ定義関数は結果セットを返すことはできません。

関連情報

ユーザ定義関数の作成 [102 ページ]

1.2.7 制御文

プロシージャ、トリガ、またはユーザ定義関数の本文、またはバッチの中には、論理フローや意思決定のための制御文が複数あります。

使用可能な制御文は、次のとおりです。

制御文	構文

複合文	`BEGIN [ ATOMIC ]`
	`   statement-list`
	`END`

条件実行: IF	`IF condition THEN`
	`   statement-list`
	`ELSEIF condition THEN`
	`   statement-list`
	`ELSE`
	`   statement-list`
	`END IF`

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

制御文	構文

条件実行: CASE	`CASE expression`
	`WHEN value THEN`
	`   Statement-list`
	`WHEN value THEN`
	`   Statement-list`
	`ELSE`
	`   Statement-list`
	`END CASE`

繰り返し: WHILE、LOOP	`WHILE condition LOOP`
	`   Statement-list`
	`END LOOP`

繰り返し: FOR カーソルループ	`FOR loop-name`
	`   AS cursor-name CURSOR FOR`
	`   select-statement`
	`DO`
	`   Statement-list`
	`END FOR`

中断: LEAVE	`LEAVE label`
CALL	`CALL procname( arg, ... )`

このセクションの内容:

複合文 [126 ページ]
  プロシージャまたはトリガの本文は複合文です。

複合文での宣言 [127 ページ]
  複合文中のローカル宣言は、キーワード BEGIN のすぐ後に続きます。

アトミックな複合文 [127 ページ]
  「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。

1.2.7.1 複合文

プロシージャまたはトリガの本文は複合文です。

複合文はキーワード BEGIN で始まり、キーワード END で終わります。複合文はバッチでも使用できます。複合文はネストが可能で、他の制御文とともにプロシージャ、トリガ、またはバッチの実行フローを定義します。

複合文は、SQL 文のセットをまとめて 1 つの単位として扱えるようにします。複合文の中の SQL 文はセミコロンで区切ります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.7.2 複合文での宣言

複合文中のローカル宣言は、キーワード BEGIN のすぐ後に続きます。

このローカル宣言は複合文中にのみ存在します。複合文に次のものを宣言できます。

- 変数
- カーソル
- テンポラリテーブル
- 例外処理 (エラー識別子)

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。ローカル宣言は、複合文中から呼び出された他のプロシージャからは見えません。

1.2.7.3 アトミックな複合文

「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、変更されたすべてのローが元の状態に戻ります。したがって、UPDATE 文はアトミックです。

複合文でないすべての SQL 文はアトミックです。BEGIN キーワードの後にキーワード ATOMIC を追加して、複合文をアトミックにすることができます。

```
BEGIN ATOMIC
   UPDATE Employees
   SET ManagerID = 501
   WHERE EmployeeID = 467;
    UPDATE Employees
   SET BirthDate = 'bad_data';
END
```

この例の 2 つの UPDATE 文は、アトミックな複合文の一部です。これら 2 つの文は、1 つの文として更新を完了するか、両方ともエラーになります。最初の UPDATE 文はエラーなしで完了するとします。次の UPDATE 文は BirthDate カラムに割り当てた値を日付に変換できないため、エラーになります。

このアトミックな複合文はエラーになり、UPDATE 文の結果は両方とも取り消されます。現在実行中のトランザクションがコミットされても、この複合文中の文は両方ともその効果をもたらしません。

アトミックな複合文が成功すると、現在実行中のトランザクションがコミットされた場合のみ、複合文中で実行された変更は有効になります。アトミックな複合文が成功しても、その文で発生したトランザクションがロールバックされた場合は、アトミックな複合文もロールバックされます。アトミックな複合文の開始時に、セーブポイントが設定されます。文でエラーが発生すると、そのセーブポイントにロールバックされます。

アトミックな複合文がオートコミット (非連鎖) モードで実行されると、文の実行が完了するまでコミットモードが手動 (連鎖) に変更されます。手動モードでは、アトミックな複合文内で DML 文を実行しても、即座に COMMIT や ROLLBACK は実行されません。アトミックな複合文が正常に完了すると、COMMIT 文が実行されます。正常に完了しない場合は ROLLBACK 文が実行されます。

COMMIT 文と ROLLBACK 文、および一部の ROLLBACK TO SAVEPOINT 文は、アトミックな複合文内で使用できません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

プロシージャ、トリガ、ユーザ定義関数でのトランザクションとセーブポイント [154 ページ]
例外処理とアトミックな複合文 [148 ページ]

1.2.8 結果セット

プロシージャは結果を 1 つまたは複数のローとして返します。

単一のローのデータからなる結果は、プロシージャへの引数として返すことができます。複数のローのデータからなる結果は、結果セットとして返されます。また、プロシージャは RETURN 文で 1 つの値を返すこともできます。

このセクションの内容:

RETURN 文を使って値を返す [129 ページ]
RETURN 文は、呼び出しを行った環境に 1 つの整数値を返した後、すぐにプロシージャを終了します。

結果をプロシージャのパラメータとして返す方法 [129 ページ]
プロシージャは、プロシージャのパラメータで呼び出しを行った環境に結果を返すことができます。

プロシージャからの結果セットで返される情報 [131 ページ]
プロシージャは結果セットで情報を返すことができます。

複数の結果セットを返す [133 ページ]
Interactive SQL を使用して、プロシージャから複数の結果セットを返します。

プロシージャの変数結果セット [134 ページ]
RESULT 句を省略すると、実行方法に応じて、さまざまなカラム数やカラム型を使った異なる結果セットを返すプロシージャを記述できます。

SYSPROCPARM システムビューで古くなった結果セットおよびパラメータ [135 ページ]
SYSPROCPARM システムビューには、プロシージャやファンクションのパラメータ、結果セット、戻り値名、およびタイプが格納されています。これらがテーブル、ビュー、プロシージャなどの変更された別のオブジェクトから抽出された場合、古くなってしまう場合があります。

関連情報

プロシージャ [89 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.8.1　RETURN 文を使って値を返す

RETURN 文は、呼び出しを行った環境に 1 つの整数値を返した後、すぐにプロシージャを終了します。

手順

1. 次の文を実行します。

```
RETURN expression
```

2. 式の値が、呼び出しを行った環境に返されます。返ってきた値を変数に保存するには、CALL 文の拡張機能を使います。

```
CREATE VARIABLE returnval INTEGER;
returnval = CALL variable/procedure-name? myproc();
```

結果

値が返され、変数として保存されます。

1.2.8.2　結果をプロシージャのパラメータとして返す方法

プロシージャは、プロシージャのパラメータで呼び出しを行った環境に結果を返すことができます。

次の文を使用して、プロシージャ内でパラメータと変数に値を割り当てることができます。

- SET 文
  次に示すプロシージャは、SET 文を使用して OUT パラメータに値を割り当てて返します。次の文を実行するには、CREATE PROCEDURE システム権限を持っていることが必要です。

```
CREATE PROCEDURE greater(
   IN a INT,
   IN b INT,
   OUT c INT )
BEGIN
   IF a > b THEN
      SET c = a;
   ELSE
      SET c = b;
   END IF ;
END;
```

- INTO 句を持つ SELECT 文
  「シングルロークエリ」がデータベースから取り出すローの数は多くても 1 つだけです。このタイプのクエリは SELECT 文に INTO 句を組み合わせて作成します。INTO 句は SELECT リストの後に続き、FROM 句より前に指定します。この句には、SELECT リストの各項目の値を受け取るための変数のリストが含まれます。変数は、SELECT リストの項目数と同じ数だけ用意します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SELECT 文が実行されると、データベースサーバは SELECT 文の結果を取り出して、変数に入れます。クエリの結果、複数のローが取り出されると、データベースサーバはエラーを返します。複数のローを返すクエリにはカーソルを使用します。
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、警告が返されます。
SELECT 文を実行するには、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。

例

例 1: プロシージャを作成し、**SELECT...INTO** 文を使用してその結果を選択します。

1. Interactive SQL を起動して、SQL Anywhere サンプルデータベースに接続します。CREATE PROCEDURE システム権限、および Employee テーブルに対する SELECT 権限または SELECT ANY TABLE システム権限のどちらかを持っていることが必要です。
2. *SQL 文*ウィンドウ枠で、次の文を実行し、従業員の平均給与を OUT パラメータとして返すプロシージャ (AverageSalary) を作成します。

```
CREATE PROCEDURE AverageSalary( OUT average_salary NUMERIC(20,3) )
BEGIN
   SELECT AVG( Salary )
   INTO average_salary
   FROM GROUPO.Employees;
END;
```

3. プロシージャの結果を格納する変数を作成します。この場合、出力変数は小数点以下 3 桁の数値となります。

```
CREATE VARIABLE Average NUMERIC(20,3);
```

4. 作成した変数を使ってプロシージャを呼び出し、結果を格納します。

```
CALL AverageSalary( Average );
```

5. プロシージャが正しく作成され、実行された場合、Interactive SQL の履歴タブにエラーは表示されません。
6. 次の文を実行して変数の値を検査します。

```
SELECT Average;
```

7. 出力変数 Average の値を見ます。結果ウィンドウ枠の結果タブに、この変数の値 49988.623 が表示されます。これが従業員の給与の平均値です。

例 2: シングルロー **SELECT** 文の結果を返します。

1. Interactive SQL を起動して、SQL Anywhere サンプルデータベースに接続します。CREATE PROCEDURE システム権限、および Customers テーブルに対する SELECT 権限または SELECT ANY TABLE システム権限のどちらかを持っていることが必要です。
2. 指定した顧客によって行われた発注の数を返すには、次の文を実行します。

```
CREATE PROCEDURE OrderCount(
   IN customer_ID INT,
   OUT Orders INT )
BEGIN
   SELECT COUNT(SalesOrders.ID)
      INTO Orders
   FROM GROUPO.Customers
      KEY LEFT OUTER JOIN SalesOrders
   WHERE Customers.ID = customer_ID;
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

3. このプロシージャは、Interactive SQL で次の文を使ってテストします。次の文は ID が 102 の顧客からの注文の回数を返します。

```
CREATE VARIABLE orders INT;
CALL OrderCount ( 102, orders );
SELECT orders;
```

例 2 の注意

- customer_ID パラメータは IN パラメータとして宣言されます。このパラメータには、プロシージャに渡される顧客 ID が格納されます。
- Orders パラメータは OUT パラメータとして宣言されます。これは変数 orders の値を呼び出し元の環境に返します。
- 変数 Orders はプロシージャの引数リストで宣言されているので、DECLARE 文は必要ありません。
- SELECT 文は 1 つのローを返して、変数 Orders に入れます。

関連情報

プロシージャからの結果セットで返される情報 [131 ページ]

1.2.8.3 プロシージャからの結果セットで返される情報

プロシージャは結果セットで情報を返すことができます。

RESULT 句の変数の数は、SELECT 文のリスト項目の数に一致しなければなりません。データ型が一致しない場合は、可能であれば自動的にデータ型の変換が行われます。SELECT 文のリスト項目の名前は、RESULT 句の変数の名前と一致する必要はありません。

RESULT 句は CREATE PROCEDURE 文の一部であり、文デリミタは付きません。

ビューでプロシージャの結果セットを変更するには、基本となるテーブルに対する適切な権限がユーザに必要です。

ストアドプロシージャまたはユーザ定義関数が結果を返す場合、出力パラメータまたは戻り値を同時に返すことはできません。

デフォルトでは、Interactive SQL は最初の結果セットのみを返します。Interactive SQL でプロシージャが結果の複数のローを返せるようにするには、オプションウィンドウの結果タブで複数の結果セットを表示オプションを設定します。

例

例 1

次に示すプロシージャは、注文した顧客のリストと、注文の合計額を返します。

Interactive SQL で次の文を実行します。

```
CREATE PROCEDURE ListCustomerValue()
RESULT ( "Company" CHAR(36), "Value" INT )
BEGIN
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    SELECT CompanyName,
       CAST( SUM(  SalesOrderItems.Quantity *
                   Products.UnitPrice )
                   AS INTEGER ) AS value
    FROM Customers
       INNER JOIN SalesOrders
       INNER JOIN SalesOrderItems
       INNER JOIN Products
    GROUP BY CompanyName
    ORDER BY value DESC;
END;
```

`CALL ListCustomerValue ( );` を実行すると、次の結果セットが返されます。

Company	Value
The Hat Company	5016
The Igloo	3564
The Ultimate	3348
North Land Trading	3144
Molly's	2808
...	...

例 2

次に示すプロシージャは、ある部署の従業員 1 人 1 人の給与をセットにして返します。Interactive SQL で次の文を実行します。

```
CREATE PROCEDURE SalaryList( IN department_id INT )
RESULT ( "Employee ID" INT, Salary NUMERIC(20,3) )
BEGIN
    SELECT EmployeeID, Salary
     FROM Employees
     WHERE Employees.DepartmentID = department_id;
END;
```

RESULT 句内の名前はクエリの結果と対応付けられ、表示される結果のカラムの見出しに使われます。

研究開発部 (部署 ID 100) の従業員の給与一覧を表示するには、次の文を実行します。

```
CALL SalaryList( 100 );
```

結果ウィンドウ枠に次の結果セットが表示されます。

Employee ID	Salary
102	45700.000
105	62000.000
160	57490.000
243	72995.000
...	...

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.8.4 複数の結果セットを返す

Interactive SQL を使用して、プロシージャから複数の結果セットを返します。

コンテキスト

デフォルトでは、Interactive SQL は複数の結果セットを返しません。

手順

1. Interactive SQL で、データベースに接続します。
2. ツール > オプション をクリックします。
3. *SQL Anywhere* をクリックします。
4. 結果タブで、すべての結果セットを表示をクリックします。
5. *OK* をクリックします。

結果

このオプションを有効にすると、Interactive SQL には複数の結果セットが表示されます。設定内容はすぐに反映され、無効にされるまで後続のセッションすべてについて有効になります。

例

次の例は、すべての従業員、顧客、連絡先の名前をリストするプロシージャです。

```
CREATE PROCEDURE ListPeople()
RESULT ( Surname CHAR(36), GivenName CHAR(36) )
BEGIN
   SELECT Surname, GivenName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

プロシージャ定義に RESULT 句を含める場合には、結果セットはそれに合わせなければなりません。つまり、結果セットは SELECT 文のリストと同じ数の項目を持ち、データ型はすべて RESULT 句にリストされたデータ型に自動的に変換可能な型である必要があります。

RESULT 句を省略した場合は、プロシージャは返されるカラムの数や型がさまざまに異なる結果セットを返すことができます。

関連情報

プロシージャの変数結果セット [134 ページ]

1.2.8.5 プロシージャの変数結果セット

RESULT 句を省略すると、実行方法に応じて、さまざまなカラム数やカラム型を使った異なる結果セットを返すプロシージャを記述できます。

プロシージャでは RESULT 句は省略可能です。変数結果セット機能を使用しない場合は、性能を高めるために RESULT 句を使用してください。

たとえば、次のプロシージャは、変数として Y を入力した場合は 2 カラムを、それ以外の場合は 1 カラムを返します。

```
CREATE PROCEDURE Names( IN formal char(1) )
BEGIN
   IF formal = 'y' THEN
      SELECT Surname, GivenName
      FROM Employees
   ELSE
      SELECT GivenName
      FROM Employees
   END IF
END;
```

クライアントアプリケーションで使用しているインタフェースによっては、プロシージャでの変数結果セットの使用に制限があります。

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) する必要があります。

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、SELECT 文の FROM 句に WITH 句を含めることをお奨めします。または、DESCRIBE 文で WITH VARIABLE RESULT 句を使用できます。WITH VARIABLE RESUL 句は、各 OPEN 文の後にプロシージャ呼び出しを記述するべきかどうかを判断するのに使用できます。

ODBC

変数結果セットプロシージャは ODBC アプリケーションで使用できます。SQL Anywhere ODBC ドライバは、変数結果セットを正しく記述します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Open Client アプリケーション

Open Client アプリケーションは、変数結果セットプロシージャを使用できます。SQL Anywhere は、変数結果セットを正しく記述します。

1.2.8.6 SYSPROCPARM システムビューで古くなった結果セットおよびパラメータ

SYSPROCPARM システムビューには、プロシージャやファンクションのパラメータ、結果セット、戻り値名、およびタイプが格納されています。これらがテーブル、ビュー、プロシージャなどの変更された別のオブジェクトから抽出された場合、古くなってしまう場合があります。

プロシージャに SELECT 文が含まれていると、SYSPROCPARM 内の値が古くなる一因となる場合があります。SELECT 文で参照されるカラムが変更されるときに、プロシージャの結果セットのカラム数やカラムタイプ数が変わります。プロシージャやファンクションで table_name.column_name%TYPE 構文を使用しており、参照されるカラムが変更されると、結果セット、パラメータ、戻り値タイプが古くなる場合もあります。

古いプロシージャやファンクション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、SYSPROCPARM が常に更新されます。

- プロシージャやファンクションが変更されて以降、参照されています。
- プロシージャやファンクションに RESULT 句が含まれます。または、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ありません。

プロシージャやファンクションの基になっているオブジェクトの変更直後に SYSPROCPARM を更新するには、関連するプロシージャやファンクションで ALTER PROCEDURE...RECOMPILE 文を実行します。

SYSPROCPARM の結果セット情報が正確に決定されない場合

次のタイプのプロシージャには、作成または変更された直後であっても、SYSPROCPARM システムビューに正確な値が表示されない場合があります。

再帰プロシージャ

例:

```
CREATE PROCEDURE p_recurse ( IN @recurse_depth INT )
BEGIN
  IF ( @recurse_depth>1 ) THEN
    CALL p_recurse ( @recurse_depth -1 );
  ELSE
    CALL p ( );
  END IF;
END;
```

10 以上の深さレベルでネストされた呼び出しがあり、RESULT 句がないプロシージャ

たとえば、プロシージャ p が SELECT 文を返すと、プロシージャ p2 が p を、プロシージャ p3 が p2 を呼び出し、プロシージャ p11 による p10 の呼び出しまで行われます。そのため、プロシージャ p11 の SYSPROCPARM 情報は正確ではない場合があります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

いくつかの結果セットのうち 1 つを返すか、または複数の結果セットを返し、**RESULT** 句のないプロシージャ

正確な結果セット、カラム名、タイプ情報を決定するには、このタイプのプロシージャに対する呼び出しでカーソルが開かれたら、カーソルを記述してください。Embedded SQL で DESCRIBE...CURSOR NAME 文を使用します。その他の API では、CALL 文が実行されるか開かれると自動的に発生します。

例

次の例では、プロシージャやファンクションの基になっているテーブルに変更が加えられたために SYSPROCPARM システムビューが古くなってしまった場合に、チェックポイント中に更新を行う方法を説明します。

1. テーブルを作成し、その後、そのテーブルに依存する多数のプロシージャと 1 つのファンクションを作成します。

```
CREATE TABLE t ( pk INTEGER PRIMARY KEY, col INTEGER );
```

```
CREATE PROCEDURE p ( )
BEGIN
  SELECT col FROM t;
END;
```

```
CREATE PROCEDURE p2 ( )
BEGIN
  CALL p ();
END;
```

```
CREATE PROCEDURE p_const ( ) RESULT ( col t.col%TYPE )
BEGIN
  SELECT 5;
END;
```

```
CREATE PROCEDURE p_no_result ( IN @pk T.pk%TYPE, OUT @col t.col%TYPE ) NO
RESULT SET
BEGIN
  SELECT col INTO @col FROM t WHERE pk=@pk;
END;
```

```
CREATE PROCEDURE p_all ( )
BEGIN
  SELECT * FROM t;
END;
```

```
CREATE FUNCTION f() RETURNS t.col%TYPE
BEGIN
  DECLARE @ret t.col%TYPE;
  SET @ret = ( SELECT FIRST col FROM t ORDER BY pk );
  RETURN ( @ret );
END;
```

2. SYSPROCPARM システムビューで、現在のパラメータ、結果セット、およびプロシージャ p の戻り値名とタイプを表示するには、次の文を実行します。

```
SELECT * FROM SYS.SYSPROCPARM
WHERE proc_id = ( SELECT proc_id FROM SYS.SYSPROCEDURE
  WHERE creator = ( SELECT user_id FROM SYS.SYSUSER WHERE user_name = CURRENT
USER )
  AND proc_name = 'p' )
ORDER BY parm_id;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SSYSPROCPARM のプロシージャ情報は、プロシージャやファンクションが作成または変更されると、ただちに更新されます。上述のクエリで、'p' を関連する任意のプロシージャやファンクションの名前に置き換えることができます。

3. 次の文を実行してテーブル t を変更します。

```
ALTER TABLE t ALTER col tinyint;
```

テーブル t を変更すると、作成したプロシージャとファンクションに次の変更が発生するため、SYSPROCPARM が古くなる原因となります。
   - プロシージャ p、p2、p_const、および p_all の結果カラムタイプの変更
   - p_no_result のパラメータタイプの変更
   - ファンクション f の戻り値の変更

   手順 2 の SYSPROCPARM のクエリを再実行します。特に domain_id、width、base_type_str カラムについて、システムビューが古くなっています。
4. 古くなっているプロシージャの 1 つにアクセスし、チェックポイントを実行して、SYSPROCPARM を更新します。

```
CALL p2 ( );
CHECKPOINT;
```

i 注記

パフォーマンスの低下を招く恐れがあるため、運用環境ではチェックポイントを実行しないことをお奨めします。

プロシージャ p2 およびプロシージャ p の両方で、SYSPROCPARM の値が更新されます。プロシージャ p2 を呼び出すと、プロシージャ p2 とプロシージャ p の両方にアクセスするためです。

1.2.9 プロシージャ、トリガ、ユーザ定義関数、バッチのカーソル

カーソルは、結果セットに複数のローがあるクエリまたはストアドプロシージャからローを 1 つずつ取り出します。

カーソルは、クエリまたはプロシージャに対するハンドルまたは識別子で、結果セットの中の現在の位置を示します。

このセクションの内容:

カーソル管理 [137 ページ]
カーソル管理は、プログラミング言語の点でファイル管理に似ています。

SELECT 文のカーソル [138 ページ]
SELECT 文でカーソルを使用できます。

プロシージャ、トリガ、ユーザ定義関数、バッチ内の位置付け更新 [140 ページ]
更新可能なカーソルを SELECT 文に対して使用できます。

1.2.9.1 カーソル管理

カーソル管理は、プログラミング言語の点でファイル管理に似ています。

カーソル管理については、次の手順に従います。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1. DECLARE 文を使って、SELECT 文またはプロシージャにカーソルを宣言します。
2. OPEN 文を使って、カーソルを開きます。
3. FETCH 文を使って、カーソルから結果をローごとに取り出します。
4. ローが見つかりませんという警告が、結果セットの最後に表示されます。
5. CLOSE 文を使ってカーソルを閉じます。

デフォルトでは、カーソルは (COMMIT 文または ROLLBACK 文の) トランザクションの最後で自動的に閉じられます。WITH HOLD 句を使って開いたカーソルは、明示的に閉じられるまで、後続のトランザクションで開いた状態になります。

1.2.9.2 SELECT 文のカーソル

SELECT 文でカーソルを使用できます。

下記の例では、ListCustomerValue プロシージャで使用するのと同じクエリに基づいて、ストアドプロシージャ言語の特徴を示します。

```
CREATE PROCEDURE TopCustomerValue(
      OUT TopCompany CHAR(36),
      OUT TopValue INT )
BEGIN
   -- 1. Declare the "row not found" exception
   DECLARE err_notfound
      EXCEPTION FOR SQLSTATE '02000';
   -- 2.  Declare variables to hold
   --     each company name and its value
   DECLARE ThisName CHAR(36);
   DECLARE ThisValue INT;
   -- 3.  Declare the cursor ThisCompany
   --      for the query
   DECLARE ThisCompany CURSOR FOR
   SELECT CompanyName,
         CAST( sum( SalesOrderItems.Quantity *
               Products.UnitPrice ) AS INTEGER )
         AS value
   FROM Customers
      INNER JOIN SalesOrders
      INNER JOIN SalesOrderItems
      INNER JOIN Products
   GROUP BY CompanyName;
   -- 4. Initialize the values of TopValue
   SET TopValue = 0;
   -- 5. Open the cursor
   OPEN ThisCompany;
   -- 6. Loop over the rows of the query
   CompanyLoop:
   LOOP
      FETCH NEXT ThisCompany
         INTO ThisName, ThisValue;
      IF SQLSTATE = err_notfound THEN
         LEAVE CompanyLoop;
      END IF;
      IF ThisValue > TopValue THEN
         SET TopCompany = ThisName;
         SET TopValue = ThisValue;
      END IF;
   END LOOP CompanyLoop;
   -- 7. Close the cursor
   CLOSE ThisCompany;
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

注記

この TopCustomerValue プロシージャには、次の特徴があります。

- 例外が宣言されます。この例外は、プロシージャの後でクエリの結果のループが完了するときに通知されます。
- クエリの各ローの結果を入れる 2 つのローカル変数 ThisName と ThisValue が宣言されます。
- カーソル ThisCompany が宣言されます。SELECT 文は会社名とその会社からの注文の合計額のリストを作成します。
- ループで使うため、TopValue の初期値は 0 に設定されています。
- ThisCompany カーソルが開きます。
- LOOP 文はクエリの各ローをループして、各会社の名前を変数 ThisName と ThisValue に入れます。ThisValue が現在の最大値よりも大きい場合、TopCompany と TopValue は ThisName と ThisValue にリセットされます。
- プロシージャの最後にカーソルは閉じられます。
- SELECT 文に ORDER BY 値 DESC 句を追加して、ループを使わずにこのプロシージャを作成することもできます。その場合、カーソルの最初のローのみをフェッチする必要があります。

TopCompanyValue プロシージャ内の LOOP 構文は標準的な形式であり、最後のローを処理した後に終了します。FOR ループを使うと、このプロシージャをさらに簡潔な形式に書き換えることができます。FOR 文は 1 つの文に、上記のプロシージャの複数の要素を組み込みます。

```
CREATE PROCEDURE TopCustomerValue2(
      OUT TopCompany CHAR(36),
      OUT TopValue INT )
BEGIN
   -- 1. Initialize the TopValue variable
   SET TopValue = 0;
   -- 2. Do the For Loop
   FOR CompanyFor AS ThisCompany
      CURSOR FOR
      SELECT CompanyName AS ThisName,
         CAST( sum( SalesOrderItems.Quantity *
               Products.UnitPrice ) AS INTEGER )
         AS ThisValue
      FROM Customers
         INNER JOIN SalesOrders
         INNER JOIN SalesOrderItems
         INNER JOIN Products
      GROUP BY ThisName
   DO
      IF ThisValue > TopValue THEN
         SET TopCompany = ThisName;
         SET TopValue = ThisValue;
      END IF;
   END FOR;
END;
```

関連情報

プロシージャからの結果セットで返される情報 [131 ページ]
エラーと警告の処理 [140 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.9.3 プロシージャ、トリガ、ユーザ定義関数、バッチ内の位置付け更新

更新可能なカーソルを SELECT 文に対して使用できます。

次の例では、ストアドプロシージャ言語を使用して、ローに対して位置付け更新を実行する更新可能カーソルを示しています。

```
CREATE PROCEDURE UpdateSalary(
  IN employeeIdent INT,
  IN salaryIncrease NUMERIC(10,3) )
BEGIN
-- Procedure to increase (or decrease) an employee's salary
  DECLARE err_notfound
      EXCEPTION FOR SQLSTATE '02000';
  DECLARE oldSalary NUMERIC(20,3);
  DECLARE employeeCursor
    CURSOR FOR SELECT Salary from Employees
               WHERE EmployeeID = employeeIdent
    FOR UPDATE;
  OPEN employeeCursor;
  FETCH employeeCursor INTO oldSalary FOR UPDATE;
  IF SQLSTATE = err_notfound THEN
    MESSAGE 'No such employee' TO CLIENT;
  ELSE
    UPDATE Employees SET Salary = oldSalary + salaryIncrease
      WHERE CURRENT OF employeeCursor;
  END IF;
  CLOSE employeeCursor;
END;
```

上記のストアドプロシージャを呼び出すには、次の文を入力してください。

```
CALL UpdateSalary( 105, 220.00 );
```

1.2.10 エラーと警告の処理

アプリケーションプログラムは SQL 文を実行した後、ステータスコード (リターンコード) をチェックできます。ステータスコードは文が正しく実行されたかどうかを示し、エラーの場合はその理由を提示します。

プロシージャを呼び出す CALL 文にも同じメカニズムを適用して、成功またはエラーを判断できます。

エラーのレポートには、SQLCODE または SQLSTATE のどちらかのステータス説明を使用します。

SQL 文が実行されると、SQLCODE および SQLSTATE と呼ばれる特別なプロシージャ変数に値が入ります。この特別値は、文の実行中に異常な状況が発生したかどうかを示します。SQLCODE と SQLSTATE の値は、IF 文を SQL 文の後に置いてチェックできます。文の結果に応じて、適切な処置を実行できます。

たとえば、SQLSTATE 変数はローが正しくフェッチされたかどうかを示すのに使用できます。TopCustomerValue プロシージャでは、SELECT 文のすべてのローが処理されたかどうかを確認するために SQLSTATE テストが使用されています。

このセクションの内容:

デフォルトのエラー処理 [141 ページ]
プロシージャ内でエラー処理が指定されていない場合、データベースサーバはデフォルト設定を使用して、プロシージャの実行中に発生したエラーを処理します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ON EXCEPTION RESUME を使用したエラー処理 [142 ページ]
: ON EXCEPTION RESUME 句が CREATE PROCEDURE 文に含まれている場合、エラーが発生すると、プロシージャによって次の文が検査されます。

デフォルトの警告処理 [144 ページ]
: エラーは警告とは異なる方法で処理されます。

例外ハンドラ [144 ページ]
: 特定の種類のエラーは呼び出しを行った環境へ戻すよりも、プロシージャまたはトリガの内部で捕捉して処理した方が良い場合があります。これは例外ハンドラを使用して行います。

例: 例外ハンドラによって呼び出し可能なエラーログプロシージャの作成 [150 ページ]
: 例外ハンドラでのアプリケーション間で統一されたエラーログに使用できる、エラーログプロシージャを定義できます。

1.2.10.1 デフォルトのエラー処理

プロシージャ内でエラー処理が指定されていない場合、データベースサーバはデフォルト設定を使用して、プロシージャの実行中に発生したエラーを処理します。

さまざまな動作に例外ハンドラを使用できます。

警告の処理はエラーの処理とは少し異なります。

エラーを処理するには、特に指定しない限り、次の 2 つの方法があります。

デフォルトのエラー処理

プロシージャかトリガがエラーを起こしたときに、呼び出しを行った環境にエラーコードが返されます。

ON EXCEPTION RESUME

CREATE PROCEDURE 文に ON EXCEPTION RESUME 句が含まれていれば、プロシージャはエラーを起こした箇所の次の文から実行を再開します。

ON EXCEPTION RESUME を使用するプロシージャの正確な動作は、on_tsql_error オプション設定によって指定します。

デフォルトのエラー処理

通常、プロシージャまたはトリガの SQL 文がエラーを起こすと、そのプロシージャまたはトリガは実行を停止し、SQLCODE と SQLSTATE に適切な値が入った状態でアプリケーションに制御が戻されます。これは最初の文から直接または間接的に呼び出されたプロシージャまたはトリガでエラーが発生したときも同じです。また、トリガの場合、トリガを起動した操作も取り消され、エラーがアプリケーションに返されます。

次の例のプロシージャは、アプリケーションからプロシージャ OuterProc を呼び出し、OuterProc が InnerProc を呼び出して、そこでエラーが発生した場合の処理を示します。

```
CREATE OR REPLACE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc' TO CLIENT;
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
      DECLARE column_not_found EXCEPTION FOR SQLSTATE '52003';
      MESSAGE 'Hello from InnerProc' TO CLIENT;
      SIGNAL column_not_found;
      MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc' TO CLIENT;
END;
CALL OuterProc();
```

Interactive SQL の履歴タブには、次のメッセージが表示されます。

```
Hello from OuterProc
Hello from InnerProc
```

InnerProc 内の DECLARE 文は、サーバが認識しているエラー条件に関連して事前に定義された SQLSTATE 値のうち、1 つの値の記号名を宣言します。

MESSAGE ... TO CLIENT 文は Interactive SQL の履歴タブにメッセージを送ります。

SIGNAL 文は InnerProc プロシージャ内から、エラー条件を生成します。

InnerProc の SIGNAL 文の後に続く文は実行されません。InnerProc は呼び出しを行った環境 (この場合はプロシージャ OuterProc) に即座に制御を戻します。OuterProc の CALL 文の後に続く文は実行されません。エラーは呼び出しを行った環境に戻され、そこで処理されます。たとえば、Interactive SQL はエラーメッセージをメッセージウィンドウに表示してエラーの処理を行います。

TRACEBACK 関数は、エラーが起きたときに実行していた文の一覧を圧縮した形で作成します。次のように、SA_SPLIT_LIST システムプロシージャを使用すると、TRACEBACK 関数の結果を分割できます。

```
SELECT * FROM SA_SPLIT_LIST( TRACEBACK(), '¥n' )
```

関連情報

例外ハンドラ [144 ページ]
デフォルトの警告処理 [144 ページ]

1.2.10.2 ON EXCEPTION RESUME を使用したエラー処理

ON EXCEPTION RESUME 句が CREATE PROCEDURE 文に含まれている場合、エラーが発生すると、プロシージャによって次の文が検査されます。

その文がエラーを処理する場合、プロシージャの実行が続行され、エラーを発生させた文の次の文から処理が再開されます。エラーが発生したとき、呼び出しを行った環境に制御を戻しません。

on_tsql_error オプション設定を使用して、ON EXCEPTION RESUME を使用するプロシージャの動作を変更できます。

エラー処理文には、次のようなものがあります。

- IF
- SELECT @variable =

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- CASE
- LOOP
- LEAVE
- CONTINUE
- CALL
- EXECUTE
- SIGNAL
- RESIGNAL
- DECLARE
- SET VARIABLE

次に示すプロシージャは、アプリケーションからプロシージャ OuterProc を呼び出し、OuterProc が InnerProc を呼び出して、そこでエラーが発生した場合の処理を示します。例文は、以前に使用したプロシージャを基にしています。

```
CREATE OR REPLACE PROCEDURE OuterProc()
ON EXCEPTION RESUME
BEGIN
   DECLARE res CHAR(5);
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   SET res = SQLSTATE;
   IF res = '52003' THEN
      MESSAGE 'SQLSTATE set to ', res, ' in OuterProc' TO CLIENT;
   END IF
END;
CREATE OR REPLACE PROCEDURE InnerProc()
ON EXCEPTION RESUME
BEGIN
   DECLARE column_not_found EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from InnerProc' TO CLIENT;
   SIGNAL column_not_found;
   MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc' TO CLIENT;
END;
CALL OuterProc();
```

Interactive SQL の履歴タブには、次のメッセージが表示されます。

```
Hello from OuterProc
Hello from InnerProc
SQLSTATE set to 52003 in OuterProc
```

実行パスを次に示します。

1. OuterProc は InnerProc を実行して呼び出します。
2. InnerProc では、SIGNAL 文がエラーを通知します。
3. MESSAGE 文はエラー処理文ではないため、制御は OuterProc に返され、メッセージは表示されません。
4. OuterProc では、エラーに続く文が SQLSTATE の値を 'res' という変数に割り当てます。これはエラー処理文なので、実行は継続され、OuterProc メッセージが表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.10.3 デフォルトの警告処理

エラーは警告とは異なる方法で処理されます。

デフォルトのエラー処理では、SQLSTATE と SQLCODE に値を設定してエラー発生時の呼び出し元環境に制御を戻しますが、デフォルトの警告処理では、SQLSTATE と SQLCODE に値を設定してプロシージャの実行を続けます。

次のプロシージャ例は、デフォルトの警告処理を示します。

この場合、ローが見つからないことを示す条件が SIGNAL 文によって生成されます。これはエラーではなく警告です。

```
CREATE OR REPLACE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc' TO CLIENT;
END;
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
   DECLARE row_not_found EXCEPTION FOR SQLSTATE '02000';
   MESSAGE 'Hello from InnerProc' TO CLIENT;
   SIGNAL row_not_found;
   MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc' TO CLIENT;
END;
CALL OuterProc();
```

Interactive SQL の履歴タブには、次のメッセージが表示されます。

```
Hello from OuterProc
Hello from InnerProc
SQLSTATE set to 02000 in InnerProc
SQLSTATE set to 00000 in OuterProc
```

両方のプロシージャとも、警告によって SQLSTATE に値 (02000) が設定された後も実行を続けました。

InnerProc で 2 番目の MESSAGE 文を実行すると、警告がリセットされます。SQL 文が正常に実行されると、SQLSTATE が 00000 に、SQLCODE が 0 にリセットされます。プロシージャがエラー状態を保存する必要がある場合、エラーまたは警告の原因となった文の実行直後に、値を割り当てる必要があります。

関連情報

デフォルトのエラー処理 [141 ページ]

1.2.10.4 例外ハンドラ

特定の種類のエラーは呼び出しを行った環境へ戻すよりも、プロシージャまたはトリガの内部で捕捉して処理した方が良い場合があります。これは例外ハンドラを使用して行います。

例外ハンドラは、複合文の EXCEPTION 部分で定義します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

複合文でエラーが起きた場合、例外ハンドラが実行されます。警告では、例外ハンドラは実行されません。ネストされた複合文の中でエラーが起きた場合、また、複合文の中から起動されたプロシージャやトリガの中でエラーが起きた場合も、例外処理コードが実行されます。

中断エラー SQL_INTERRUPT、SQLSTATE 57014 の例外ハンドラには、ROLLBACK や ROLLBACK TO SAVEPOINT などの中断のできない文だけを含めます。例外ハンドラに、接続の中断時に呼び出される中断可能な文を含めると、データベースサーバは最初の中断可能な文で例外ハンドラを停止し、中断エラーを返します。

例外ハンドラでは、文が失敗した原因を判断するために、SQLSTATE または SQLCODE 特別値を使用できます。代わりに、ERRORMSG 関数を引数なしで使用すると、SQLSTATE に関連付けられたエラー条件が返されます。この情報を指定できるのは、各 WHEN 句の最初の文のみであり、文は複合文にはできません。

次の例では、カラムが見つからないエラーが InnerProc 内の例外ハンドラによって処理されます。デモンストレーションのため、SIGNAL 文によってエラーを人為的に発生させています。

OuterProc にも例外ハンドラが含まれています。

```
CREATE OR REPLACE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc (no exception)' TO CLIENT;
   EXCEPTION
      WHEN OTHERS THEN
        MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc (exception)' TO CLIENT;
        RESIGNAL ;
END;
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
   DECLARE column_not_found EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from InnerProc' TO CLIENT;
   SELECT  'OK';
   SIGNAL column_not_found;
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in InnerProc (no exception)' TO CLIENT;
   EXCEPTION
      WHEN column_not_found THEN
        MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc (exception)' TO CLIENT;
        --RESIGNAL;
      WHEN OTHERS THEN
        RESIGNAL;
END;
CALL OuterProc();
```

この例を Interactive SQL を使用して実行すると、結果タブに OK という結果が表示されます。履歴タブには、次のメッセージが表示されます。

```
Hello from OuterProc
Hello from InnerProc
SQLSTATE set to 52003 in InnerProc (exception)
SQLSTATE set to 00000 in OuterProc (no exception)
```

EXCEPTION 句は、1 つ以上の例外ハンドラの開始を宣言します。EXCEPTION 以降の行はエラーが起きないかぎり実行されません。各 WHEN 句は例外名 (DECLARE 文で宣言される) と、その例外が起こったときに実行される 1 つ以上の文を指定します。

WHEN OTHERS THEN 句は、発生した例外が先行の WHEN 句に現れないときに実行される文を定義します。

上記の例では、RESIGNAL 文は例外を上位レベルの例外ハンドラに渡します。例外ハンドラの中に WHEN OTHERS THEN が指定されていない場合、処理されない例外に対するデフォルトのアクションはすべて RESIGNAL となります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

RESIGNAL 文からコメントインジケータを削除すれば、column_not_found 例外を OuterProc に戻すことができます。すると、OuterProc プロシージャ内の例外ハンドラが起動されることになります。

追加の注意事項

- InnerProc の SIGNAL 文に続く行ではなく、EXCEPTION ハンドラが実行されます。
- 見つからないカラムに関するエラーが発生したため、エラー処理のための MESSAGE 文が実行され、SQLSTATE は 0 にリセットされます (エラーがないことを示します)。
- 例外処理コードが実行された後、制御は OuterProc に戻され、OuterProc はエラーがなかったものとして続行されます。
- ON EXCEPTION RESUME は明示的な例外処理と一緒には使用しないでください。ON EXCEPTION RESUME が含まれていると、例外処理コードは実行されません。
- エラーに対するエラー処理コードが RESIGNAL 文である場合、制御は OuterProc プロシージャに戻され、SQLSTATE の値は 52003 に設定されたままになります。これは、InnerProc にエラー処理コードがないのと同じです。OuterProc にはエラー処理コードがないため、プロシージャはエラーになります。

このセクションの内容:

RESIGNAL による例外処理 [147 ページ]
ユーザ定義のストアドプロシージャに、RESIGNAL を使用して例外を呼び出し元に戻す例外ハンドラが含まれている場合、呼び出し元のプロシージャが結果セットを得られないことがあります。結果が得られるかどうかは、そのユーザ定義ストアドプロシージャの呼び出し方法によります。

例外処理とアトミックな複合文 [148 ページ]
アトミック複合文でエラーが発生し、その文にエラー処理用の例外ハンドラが実装されている場合、複合文はアクティブな例外なしで完了し、例外より前の変更は取り消されません。

例外処理とネストされた複合文 [148 ページ]
エラーを引き起こした文に続くコードが実行されるのは、プロシージャ定義に ON EXCEPTION RESUME 句が含まれる場合のみです。

関連情報

デフォルトのエラー処理 [141 ページ]
複合文 [126 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.10.4.1 RESIGNAL による例外処理

ユーザ定義のストアドプロシージャに、RESIGNAL を使用して例外を呼び出し元に戻す例外ハンドラが含まれている場合、呼び出し元のプロシージャが結果セットを得られないことがあります。結果が得られるかどうかは、そのユーザ定義ストアドプロシージャの呼び出し方法によります。

ユーザ定義のストアドプロシージャに含まれる SELECT 文が別のストアドプロシージャを呼び出し、そのプロシージャで例外が発生した場合、どうなるでしょうか。

エラーが発生し、例外ハンドラが存在する場合、ユーザ定義のストアドプロシージャに含まれているのが SELECT 文か CALL 文かで、その実行は異なります。

この状況について、次の例で説明します。

```
CREATE OR REPLACE PROCEDURE OuterProc()
RESULT ( res CHAR(5) )
BEGIN
  -- CALL InnerProc();
  SELECT res FROM InnerProc();
  EXCEPTION
    WHEN OTHERS THEN
      MESSAGE 'Exception in OuterProc' TO CLIENT;
      RESIGNAL;
END;
CREATE OR REPLACE PROCEDURE InnerProc()
RESULT ( res CHAR(5) )
BEGIN
  SELECT 'OK_1' UNION ALL SELECT 'OK_2';
  SET tst = '3';
  EXCEPTION
    WHEN OTHERS THEN
      MESSAGE 'Exception in InnerProc' TO CLIENT;
      RESIGNAL;
END;
CALL InnerProc();
CALL OuterProc();
```

Interactive SQL を使用して `CALL InnerProc()` 文を実行すると、エラーが発生し、結果セットが次のように表示されます。

```
OK_1
OK_2
```

Interactive SQL の履歴タブには、次のメッセージが表示されます。

```
Exception in InnerProc
```

これは次のような流れで実行されます。

1. InnerProc が実行され、SELECT 文によって結果が 2 行生成されます。
2. 変数 `tst` が定義されていないため SET 文がエラーとなり、EXCEPTION ブロックが実行されます。
3. MESSAGE 文によって、Interactive SQL にメッセージテキストが送信されます。
4. RESIGNAL 文によって、エラーが呼び出し元 (CALL 文) に戻されます。
5. Interactive SQL はエラーを表示し、続いて結果セットを表示します。

Interactive SQL を使用して `CALL OuterProc()` 文を実行すると、エラーが発生し、結果セットは生成されません。

Interactive SQL の履歴タブを見ても、InnerProc のメッセージしか表示されていません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

これは次のような流れで実行されます。

1. OuterProc が結果セットを生成するので、クライアントはクライアント側のカーソルを開いてこの結果セットを処理する必要があります。
2. カーソルが開かれると、OuterProc は最初の結果セットを得るための文 (SELECT 文) に達するところまで実行されます。ここで、文の実行準備が行われます (ただし実行はされません)。
3. データベースサーバは停止し、制御がクライアントに戻されます。
4. クライアントは結果セットの最初の行をフェッチしようとします。最初の行を得るため、制御はサーバに戻ります。
5. サーバは、準備済みの文を実行します (これは、プロシージャの実行とは無関係に行われます)。
6. サーバは、結果セットの最初の行を得るため InnerProc を実行し、ここで例外が発生します (例外は InnerProc 内の EXCEPTION 文によって捕捉され、送り返されます)。プロシージャの実行は実際上クライアントによって行われるので、例外はクライアントに戻され、OuterProc の EXCEPTION には捕捉されません。

SQL Anywhere は結果セットをオンデマンド方式で生成しますが、DBMS によってはプロシージャを論理エンドポイントまで完全に実行し、全体の結果セットをすべて生成してから制御をクライアントに戻すものもあるかもしれないので注意してください。

OuterProc 内の SELECT 文を CALL 文に変更すれば、結果セット全体が OuterProc 内で生成され、OuterProc の例外ハンドラが起動されます。

1.2.10.4.2 例外処理とアトミックな複合文

アトミック複合文でエラーが発生し、その文にエラー処理用の例外ハンドラが実装されている場合、複合文はアクティブな例外なしで完了し、例外より前の変更は取り消されません。

例外ハンドラが発生したエラーを処理しない場合、または別のエラー (RESIGNAL によるエラーを含む) を発生させた場合、アトミックな複合文で行われた変更は取り消されます。

1.2.10.4.3 例外処理とネストされた複合文

エラーを引き起こした文に続くコードが実行されるのは、プロシージャ定義に ON EXCEPTION RESUME 句が含まれる場合のみです。

ネストされた複合文を使用すると、エラーの後にどの文が実行され、どの文が実行されないのかを制御できます。

次の例は、ネストされた複合文をどのように使用してフローを制御するかを示します。

```
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
   BEGIN
      DECLARE column_not_found EXCEPTION FOR SQLSTATE VALUE '52003';
      MESSAGE 'Hello from InnerProc' TO CLIENT;
      SIGNAL column_not_found;
      MESSAGE 'Line following SIGNAL' TO CLIENT;
      EXCEPTION
         WHEN column_not_found THEN
            MESSAGE 'Column not found handling' TO CLIENT;
         WHEN OTHERS THEN
            RESIGNAL;
   END;
   MESSAGE 'Outer compound statement' TO CLIENT;
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
CALL InnerProc();
```

Interactive SQL の履歴タブには、次のメッセージが表示されます。

```
Hello from InnerProc
Column not found handling
Outer compound statement
```

エラーを引き起こす SIGNAL 文が検出されると、制御は複合文の例外ハンドラに渡されて、`Column not found handling` メッセージが出力されます。次に制御は外部複合文に戻され、`Outer compound statement` メッセージが出力されます。

内部複合文で "カラムが見つかりません" (SQLSTATE) 以外のエラーが検出されると、例外ハンドラは RESIGNAL 文を実行します。RESIGNAL 文は、呼び出しを行った環境に制御を直接戻します。外部複合文の残りの文は実行されません。

例

次の例は、EXCEPTION、RESIGNAL、ネストされた BEGIN 文を使用するプロシージャに関する 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
    BEGIN
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
        EXCEPTION
            WHEN OTHERS THEN
                CALL proc2();
    END
    EXCEPTION
        WHEN OTHERS THEN
            CALL error_reporting_procedure();
END;
CREATE OR REPLACE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;
CREATE OR REPLACE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;
CALL proc1();
```

`proc1` プロシージャが呼び出されると、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

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の例は、RESIGNAL およびネストされた BEGIN TRY/CATCH 文を使用するプロシージャに関する 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();
```

proc1 プロシージャが呼び出されると、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

1.2.10.5 例: 例外ハンドラによって呼び出し可能なエラーログプロシージャの作成

例外ハンドラでのアプリケーション間で統一されたエラーログに使用できる、エラーログプロシージャを定義できます。

1. 次のテーブルを作成して、エラーログプロシージャが実行されるたびにエラー情報のログを作成します。

```
CREATE TABLE IF NOT EXISTS error_info_table (
   idx INTEGER,
   ln UNSIGNED INTEGER,
   code INTEGER,
   state CHAR(5),
   err_msg CHAR(256),
   name CHAR(257),
   err_stack LONG VARCHAR,
   traceback LONG VARCHAR
);
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
CREATE TABLE IF NOT EXISTS error_stack_trace_table (
   idx UNSIGNED SMALLINT NOT NULL,
   stack_level UNSIGNED SMALLINT NOT NULL,
   user_name VARCHAR(128),
   proc_name VARCHAR(128),
   line_number UNSIGNED INTEGER NOT NULL,
   is_resignal BIT NOT NULL, PRIMARY KEY (idx, stack_level)
);
```

2. error_info_table と error_stack_trace_table にエラー情報のログを作成して、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書き込む、次のプロシージャを作成します。

```
CREATE OR REPLACE PROCEDURE error_report_proc ( IN location_indicator INTEGER )
NO RESULT SET
BEGIN
   INSERT INTO error_info_table VALUES (
     location_indicator,
     ERROR_LINE(),
     ERROR_SQLCODE(),
     ERROR_SQLSTATE(),
     ERROR_MESSAGE(),
     ERROR_PROCEDURE(),
     ERROR_STACK_TRACE(),
     TRACEBACK()
   );
   INSERT INTO error_stack_trace_table
     SELECT location_indicator, *
     FROM sa_error_stack_trace() ;
   MESSAGE 'The error message is '|| ERROR_MESSAGE() ||' and the stack trace is
'|| ERROR_STACK_TRACE()
   TYPE WARNING TO CONSOLE ;
END;
```

3. 次のようなプロシージャを作成して、例外ハンドラからエラーログプロシージャを起動します。

```
CREATE OR REPLACE PROCEDURE MyProc()
BEGIN
   DECLARE column_not_found
     EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from MyProc.' TO CLIENT;
   SIGNAL column_not_found;
   MESSAGE 'Line following SIGNAL.' TO CLIENT;
EXCEPTION
WHEN column_not_found THEN
   MESSAGE 'Column not found handling.' TO CLIENT;
   CALL error_report_proc();
END ;
```

関連情報

例外ハンドラ [144 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.11 プロシージャ、トリガ、ユーザ定義関数、バッチで使用される EXECUTE IMMEDIATE

EXECUTE IMMEDIATE 文を使うと、文字列 (引用符で囲む) と変数を使って文を組み立てることができます。

次に示すのは、テーブルを作成する EXECUTE IMMEDIATE 文を含むプロシージャの例です。

```
CREATE PROCEDURE CreateTableProcedure(
      IN tablename CHAR(128) )
BEGIN
   EXECUTE IMMEDIATE 'CREATE TABLE '
   || tablename
   || '( column1 INT PRIMARY KEY )'
END;
```

EXECUTE IMMEDIATE 文は、結果セットを返すクエリで使用できます。文が結果セットを返すことを指定するには、EXECUTE IMMEDIATE 文で WITH RESULT SET ON 句を使用します。デフォルトの動作では、文は結果セットを返しません。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。

次のプロシージャを考えてみます。

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

プロシージャ定義に RESULT SET 句が含まれていないと、データベースサーバ側ではプロシージャによって結果セットが生成されるのかどうかを判断しようとします。この EXECUTE IMMEDIATE 文は、結果セットを生成しないように設定されています。そのため、データベースサーバ側では、プロシージャには結果セットのカラムがないと判断されるため、このプロシージャの SYSPROCPARM システムビューにはローが表示されません。このプロシージャの CALL に DESCRIBE を指定しても、結果カラムは返されません。カーソルを開くかどうか、文を実行するかどうかを決定するために、Embedded SQL アプリケーションによってこの情報が使用された場合には、文が実行されて、エラーが返されます。

2 番目の例では、上記のプロシージャの修正バージョンを考えてみます。

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1';
END;
```

ここでは、WITH RESULT SET ON 句が指定されているため、SYSPROCPARM システムビューにはこのプロシージャのローが存在しています。プロシージャが EXECUTE IMMEDIATE を使用しているため、データベースサーバ側では、実際に結果セットがどのようになるかを把握できません。ただし、結果セットが存在することは予測できるため、データベースサーバは SYSPROCPARM に、expression という名前で SMALLINT 型の結果セット用ダミーカラムを定義することによって、結果セットに対応していることを示します。作成される結果セット用ダミーカラムは *1 つ* だけです。サーバ側では、EXECUTE IMMEDIATE 文が使用されているときに、各結果セットのカラム数や型について判断できません。そのため、さらに少し修正したバージョンの例を考えてみます。

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1, 2, 3';
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ここでは、SELECT 文は 3 つのカラムの結果セットを返しますが、サーバの SYSPROCPARM システムビューには、1 つのローがあるだけです。そこで、次のクエリを見てください。

```
SELECT * FROM test_result_clause();
```

このクエリは、SQLCODE -866 エラーになります。これは、実行時の結果セットの定義が SYSPROCPARM に事前に用意された定義と一致しないためです。

上記のクエリを実行するには、次に示すように、結果セットのカラムの名前と型を明示的に指定します。

```
SELECT * FROM test_result_clause() WITH (x INTEGER, y INTEGER, z INTEGER);
```

WITH RESULT SET ON が指定されている場合は、実行時に、データベースサーバは EXECUTE IMMEDIATE 文を処理し、結果セットを返します。ただし、WITH RESULT SET OFF が指定されているか、または句が省略されている場合でも、データベースサーバは解析された文字列引数の最初の文の型については、同じように認識することができます。その文が SELECT 文であれば、結果セットを返します。それでは、前述した 2 番目の例を見てください。

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

Interactive SQL からこのプロシージャを呼び出すとエラーにはなりません。ここで、1 つの SELECT 文ではなく、バッチを含むようにプロシージャを変更するとします。

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF
    'begin declare v int; set v=1; select v; end';
END;
```

その場合には、test_result_clause プロシージャを呼び出すと、エラー (SQLCODE -946, SQLSTATE 09W03) が発生します。

この最後の例では、プロシージャ内で SELECT 文を EXECUTE IMMEDIATE 文の引数として構築して、そのプロシージャから結果セットを返すようにする方法を示しています。

```
CREATE PROCEDURE DynamicResult(
   IN Columns LONG VARCHAR,
   IN TableName CHAR(128),
   IN Restriction LONG VARCHAR DEFAULT NULL )
BEGIN
    DECLARE Command LONG VARCHAR;
    SET Command = 'SELECT ' || Columns || ' FROM ' || TableName;
    IF ISNULL( Restriction,'') <> '' THEN
         SET Command = Command || ' WHERE ' || Restriction;
    END IF;
    EXECUTE IMMEDIATE WITH RESULT SET ON Command;
END;
```

このプロシージャは次のように呼び出されます。

```
CALL DynamicResult(
   'table_id,table_name',
   'SYSTAB',
   'table_id <= 10');
```

これは次のような結果になります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

table_id	table_name
1	ISYSTAB
2	ISYSTABCOL
3	ISYSIDX
...	...

上記の CALL では、プロシージャで EXECUTE IMMEDIATE が使用されていても、正しく結果セットが返されます。ODBC などの一部のサーバ API では、PREPARE-DESCRIBE-EXECUTE-OR-OPEN を使用して要求が結合されており、文が結果セットを返すかどうかに応じて、文を実行するのか、または文を開くのかが決定されます。文を開く必要がある場合には、API またはアプリケーションは DESCRIBE CURSOR を続けて発行することによって、プロシージャが作成されたときに構築された SYSPROCPARM システムビューの内容に依存するのではなく、実際の結果セットの内容を把握できるようになります。この方法は、DBISQL と DBISQLC の両方で使用されています。この場合、上記のプロシージャの CALL はエラーにならずに実行されます。ただし、文の DESCRIBE の結果に依存するアプリケーションインタフェースでは、一部の文を処理できない場合があります。

ATOMIC 複合文の中では、COMMIT を伴う EXECUTE IMMEDIATE 文は使用できません。COMMIT 文はこのコンテキストでは許可されていません。

1.2.12 プロシージャ、トリガ、ユーザ定義関数でのトランザクションとセーブポイント

プロシージャまたはトリガ内の SQL 文は現在のトランザクションの一部です。

1 つのトランザクション内で複数のプロシージャを呼び出すことや、1 つのプロシージャ内に複数のトランザクションを持つことができます。

アトミックな文中では COMMIT と ROLLBACK は許可されません。

トリガはアトミックな文である INSERT、UPDATE、DELETE によって起動さます。COMMIT と ROLLBACK はトリガまたはトリガから呼び出されたプロシージャ内では許可されません。

プロシージャまたはトリガではセーブポイントを使用できますが、ROLLBACK TO SAVEPOINT 文はアトミックオペレーションが開始される以前のセーブポイントを参照することはできません。また、アトミックオペレーション内のすべてのセーブポイントは、その操作が終了したときに解除されます。

関連情報

トランザクションと独立性レベル [757 ページ]
アトミックな複合文 [127 ページ]
トランザクション内のセーブポイント [762 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.2.13 プロシージャ、トリガ、ユーザ定義関数、バッチを作成するときのヒント

プロシージャ、トリガ、ユーザ定義関数、バッチを作成するときに役立つヒントがいくつかあります。

SQL 文デリミタを変更する必要性のチェック

プロシージャを作成するときに、文デリミタを変更する必要はありません。ただし、他のブラウズツールを使用してプロシージャやトリガの作成とテストを行う場合には、文デリミタをセミコロンから他の文字に変更する必要があります。

プロシージャ内の各文はセミコロンで終わります。ブラウズするアプリケーションが CREATE PROCEDURE 文自体を解析するには、文デリミタにセミコロン以外の文字を使用する必要があります。

文デリミタを変更する必要があるアプリケーションを使用する場合は、文デリミタとして 2 つのセミコロン (;;) を使用するか、複数の文字によるデリミタが許可されないシステムであれば疑問符 (?) が適切です。

プロシージャの中で文を区切る

プロシージャ内の各文はセミコロンで終わらせます。最後の文はセミコロンがなくてもかまいませんが、各文の後ろにはセミコロンを付けることを習慣にしてください。

CREATE PROCEDURE 文は本体である複合文と、RESULT 指定の両方を含みます。キーワード BEGIN または END の後と、RESULT 句の後には、セミコロンは必要ありません。

プロシージャ内のテーブル名には完全修飾名を使用する

プロシージャ内でテーブルを参照する場合は、必ずテーブルの所有者 (作成者) の名前をプレフィクスとしてテーブル名に付けてください。

プロシージャがテーブルを参照するときは、プロシージャ作成者のロールメンバーシップを使い、所有者の名前は指定しません。たとえば、user_1 が作成したプロシージャが Table_B を参照し、Table_B の所有者の名前を指定しないとします。この場合、Table_B が user_1 によって作成されたか、user_1 が Table_B の所有者であるロールの (直接または間接的に) メンバーでなければなりません。どちらの条件も満たされない場合は、プロシージャが呼び出されると、「`table not found`」というメッセージが表示されます。

FROM 句でテーブルの相関名を使用すると、長い完全修飾名を入力しないですみます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

プロシージャの中で日付と時刻を指定する

プロシージャは、日付と時刻を文字列としてデータベースに送ります。この文字列は date_order データベースオプションの現在の設定に従って変換されます。異なる接続はこのオプションを異なる値に設定することもあるので、文字列が間違った日付に変換されたり、まったく変換できなかったりすることがあります。

プロシージャ内で日付文字列を使用するときには、あいまいでない日付形式 (`yyyy-mm-dd` または `yyyy/mm/dd`) を使用します。date_order データベースオプションの設定に関係なく、サーバはこれらの文字列を日付として正しく解釈します。

プロシージャの入力引数が正しく渡されていることを検証する

入力引数を検証する 1 つの方法は、MESSAGE 文を使って Interactive SQL の履歴タブにパラメータ値を表示することです。たとえば、次に示すプロシージャは、入力パラメータの *var* の値を表示します。

```
CREATE PROCEDURE message_test( IN var char(40) )
BEGIN
   MESSAGE var TO CLIENT;
END;
```

デバッガを使用して、プロシージャの入力引数が正常に渡されたことを確認することもできます。

関連情報

レッスン 2: バグの診断 [838 ページ]

1.2.14 プロシージャ、トリガ、イベント、バッチで使用できる文

バッチでは大部分の SQL 文を使用できますが、いくつかの例外があります。

- ALTER DATABASE (構文に依存する)
- CONNECT
- CREATE DATABASE
- CREATE DECRYPTED FILE
- CREATE ENCRYPTED FILE
- DISCONNECT
- DROP CONNECTION
- DROP DATABASE
- FORWARD TO
- INPUT、OUTPUT などの Interactive SQL 文
- PREPARE TO COMMIT
- STOP SERVER

COMMIT、ROLLBACK、SAVEPOINT 文はプロシージャ、トリガ、バッチで使用できますが、若干の制限があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

バッチで使用される SELECT 文 [157 ページ]
バッチには 1 つまたは複数の SELECT 文を含めることができます。

関連情報

プロシージャ、トリガ、ユーザ定義関数でのトランザクションとセーブポイント [154 ページ]

1.2.14.1 バッチで使用される **SELECT** 文

バッチには 1 つまたは複数の SELECT 文を含めることができます。

次に例を示します。

```
IF EXISTS( SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
END IF;
```

結果セットのエイリアスは、最初の SELECT 文でのみ必要です。サーバはバッチ中の最初の SELECT 文を結果セットの記述に使用するからです。

各クエリの後には、次の結果セットを取り出すための RESUME 文が必要です。

1.2.15 プロシージャ、ファンクション、トリガ、イベント、またはビューの内容を隠す

SET HIDDEN 句を使用して、プロシージャ、関数、トリガ、イベント、またはビューの内容を隠します。

前提条件

オブジェクトの所有者であるか、ALTER ANY OBJECT システム権限を持っているか、次の権限の 1 つを持っていることが必要です。

プロシージャとファンクション

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ALTER ANY PROCEDURE システム権限

ビュー

ALTER ANY VIEW システム権限

イベント

MANAGE ANY EVENT システム権限

トリガ

- ALTER ANY TRIGGER システム権限
- 基礎となるテーブルに対する ALTER 権限と CREATE ANY OBJECT システム権限
- ビューに対するトリガについては、ALTER ANY TRIGGER と ALTER ANY VIEW のシステム権限を持っていることが必要です。

コンテキスト

プロシージャ、ファンクション、トリガ、イベント、およびビュー内に含まれるロジックを公開せずにアプリケーションとデータベースを配布する場合は、ALTER PROCEDURE、ALTER FUNCTION、ALTER TRIGGER、ALTER EVENT、ALTER VIEW 文の SET HIDDEN 句を使用して、これらのオブジェクトの内容を隠すことができます。

SET HIDDEN 句は、関連オブジェクトを使用可能な状態に保ちながら、その内容を難読化して読み取れないようにします。また、アンロードして、別のデータベースに再ロードすることもできます。

修正を元に戻すことはできません。修正すると、オブジェクトの元のテキストが削除されます。オブジェクトの元のソースをデータベースの外部に保存しておく必要があります。

デバッガによるデバッグではプロシージャ定義が表示されず、SQL Anywhere プロファイラ にもソースが表示されません。

i 注記

preserve_source_format データベースオプションを On に設定すると、データベースサーバは、プロシージャ、ビュー、トリガ、イベントの CREATE 文と ALTER 文によってフォーマットされたソースを保存し、それを適切なシステムビューのソースカラムに配置します。この場合、オブジェクト定義とソース定義の両方が隠されます。

ただし、preserve_source_format データベースオプションを On に設定しても、SET HIDDEN 句でオブジェクトの元のソース定義を削除できます。

手順

SET HIDDEN 句を持つ適切な ALTER 文を使用します。

オプション	アクション
個別のオブジェクトを隠す	1 つのプロシージャ、ファンクション、トリガ、イベント、またはビューを隠すには、SET HIDDEN 句を持つ適切な ALTER 文を実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプション	アクション
特定のタイプのすべてのオブジェクトを隠す	すべてのプロシージャ、ファンクション、トリガ、イベント、またはビューを隠すには、SET HIDDEN 句を持つ適切な ALTER 文をループで実行します。

結果

オートコミットが実行されます。オブジェクト定義は表示されません。ただし、オブジェクトは直接参照でき、クエリ処理中に使用できることは変わりません。

例

すべてのプロシージャを隠すには、次のループを実行します。

```
BEGIN
    FOR hide_lp as hide_cr cursor FOR
        SELECT proc_name, user_name
        FROM SYS.SYSPROCEDURE p, SYS.SYSUSER u
        WHERE p.creator = u.user_id
        AND p.creator NOT IN (0,1,3)
    DO
        MESSAGE 'altering ' || proc_name;
        EXECUTE IMMEDIATE 'ALTER PROCEDURE "' ||
            user_name || '"."' || proc_name
            || '" SET HIDDEN'
    END FOR
END;
```

1.3 クエリとデータ修正

データベースのデータの問い合わせや修正を行うために、多くの機能が用意されています。

このセクションの内容:

クエリ [160 ページ]
クエリはデータベースからデータを要求します。

全文検索 [248 ページ]
テーブルに対して全文検索を実行できます。

チュートリアル: テーブルデータの行列変換 [352 ページ]
クエリの FROM 句で PIVOT 句を使用して、テーブル式のテーブルデータを行列変換します。

クエリ結果の要約、グループ化、ソート [355 ページ]
クエリ結果のグループ化およびソートを可能にするプロシージャ、文、句が、いくつかサポートされています。

ジョイン: 複数テーブルからのデータ検索 [376 ページ]
複数のテーブルから関連データを取り出すには、SQL JOIN 演算子を使用してジョイン操作を行います。

共通テーブル式 [420 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

共通テーブル式は、WITH 句を使用して定義します。WITH 句は、SELECT 文内の SELECT キーワードに先行します。

OLAP のサポート [435 ページ]
OLAP (オンライン分析処理) では、単一の SQL 文内で複雑なデータ分析を実行できます。また、データベースでクエリの量を減らすことでパフォーマンスを向上しながら、結果の値を増やすことができます。

サブクエリの使用 [474 ページ]
リレーショナルデータベースを使用すると、複数のテーブルに関連するデータを保存できます。ジョインを使用して関連するテーブルからデータを抽出できるほか、サブクエリを使用しても抽出できます。

データ操作文 [498 ページ]
データの追加、変更、削除に使用する文はデータ操作文と呼ばれ、ANSI SQL のデータ操作言語 (DML) 文部分のサブセットとなっています。

1.3.1 クエリ

クエリはデータベースからデータを要求します。

この処理はデータ取得とも呼ばれます。SQL クエリはすべて、SELECT 文を使用して表現されます。SELECT 文は、1 つ以上のテーブルですべてのローまたはそのサブセットを検索するとき、および 1 つ以上のテーブルですべてのカラムまたはそのサブセットを検索するときに使用します。

このセクションの内容:

SELECT 文と問い合わせ [161 ページ]
SELECT 文は、クライアントアプリケーションで使用する情報をデータベースから取り出します。

クエリの述部 [162 ページ]
述部は条件式であり、論理演算子 AND や OR と組み合わせて、WHERE 句、HAVING 句、または ON 句に条件のセットを構成します。

SQL クエリ [164 ページ]
このマニュアルの SELECT 文とその他の SQL 文の表記では、それぞれの句を個別の行で示し、SQL キーワードを大文字で示します。

SELECT リスト:カラムの指定 [165 ページ]
SELECT リストは、通常はカンマで区切った一連のカラム名か、またはすべてのカラムを表すアスタリスク演算子で構成されています。

FROM 句: テーブルの指定 [175 ページ]
テーブル、ビュー、またはストアドプロシージャのデータを返すすべての SELECT 文には、FROM 句が必須です。

WHERE 句: ローの指定 [179 ページ]
SELECT 文の WHERE 句は、ローの取得時にデータベースサーバが適用する必要がある検索条件を指定します。

ORDER BY 句: 結果の順序付け [192 ページ]
特に指定しないかぎり、データベースサーバは、テーブルのローを意味のない順序で返します。

ORDER BY のパフォーマンスを改善するインデックス [194 ページ]
インデックスを使用すると、データベースサーバがより効率的にテーブルを検索できるようになります。

クエリの集合関数 [195 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

集合関数および GROUP BY 句を使用すると、テーブル内の、個別のローではなくローグループのプロパティを反映するデータの内容を検査するのに役立ちます。

高度: クエリ処理のフェーズ [199 ページ]
注釈フェーズから実行完了まで、文にはいくつかのフェーズがあります

高度: クエリ最適化 [202 ページ]
最適化は、クエリの適切なアクセスプランを生成するのに重要な処理です。

高度: クエリ実行プラン [210 ページ]
実行プランは、データベースサーバがデータベース内の文に関連する情報にアクセスするために使用する一連のステップです。

高度: クエリ実行時の並列処理 [246 ページ]
クエリ実行に対して、クエリ間とクエリ内の 2 種類の並列処理があります。

関連情報

Query Processing Based on SQL Anywhere 12.0.1 Architecture

1.3.1.1 SELECT 文と問い合わせ

SELECT 文は、クライアントアプリケーションで使用する情報をデータベースから取り出します。

SELECT 文はクエリとも呼ばれます。情報は、結果セットとしてクライアントアプリケーションに配信されます。クライアントは、配信された結果セットを処理できます。たとえば、Interactive SQL では、結果セットが [結果] ウィンドウ枠に表示されます。結果セットは、データベースのテーブルと同様に一連のローで構成されます。

SELECT 文には、返される結果のスコープを定義する句が含まれます。次の SELECT 構文では、各行が別々の句です。ここではごく一般的な句を示します。

```
SELECT select-list
[ FROM table-expression ]
[ WHERE search-condition ]
[ GROUP BY column-name ]
[ HAVING search-condition ]
[ ORDER BY { expression | integer } ]
```

次に、SELECT 文の各句について説明します。

- SELECT 句は、どのカラムを取得するかを指定します。この句は、SELECT 文で必須の句です。
- FROM 句は、どのテーブルからカラムを検索するかを指定します。この句は、テーブルからデータを取り出すすべてのクエリに必要です。SELECT 文に FROM 句を使用しない場合は、異なる意味になります。
  ほとんどのクエリはテーブルを操作しますが、クエリを使用して、ビュー、他のクエリ (派生テーブル)、ストアドプロシージャの結果セットなど、カラムとローを持つ他のオブジェクトからデータを取り出せます。
- WHERE 句は、参照するテーブルのローを指定します。
- GROUP BY 句を使用するとデータを集約できます。
- HAVING 句は、集合データが収集されるローを指定します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ORDER BY 句は、結果セット内のローをソートします。(デフォルトでは、リレーショナルデータベースから返されるローの順序に意味はありません)。

これらの句の大半は省略可能ですが、SELECT 文に記述するときは、正しい順序で記述してください。

関連情報

クエリ結果の要約、グループ化、ソート [355 ページ]

1.3.1.2 クエリの述部

述部は条件式であり、論理演算子 AND や OR と組み合わせて、WHERE 句、HAVING 句、または ON 句に条件のセットを構成します。

SQL では、UNKNOWN と評価される述部が FALSE として解釈されます。

インデックスを使用してテーブルからローを取り出すことができる述部を、検索引数可能 **(sargable)** であるといいます。この名前は、*search argument-able* というフレーズからとったものです。定数、他のカラム、または式とカラムとの比較を伴う述部は、検索引数可能です。

次に示す文の述部は、検索引数可能です。データベースサーバは、Employees テーブルのプライマリインデックスを使用して、この文を効率的に評価できます。

```
SELECT *
FROM Employees
WHERE Employees.EmployeeID = 102;
```

最適なアクセスプランでは、これは次のように表示されます。`Employees<Employees>`

反対に、次に示す述部は、検索引数可能ではありません。EmployeeID カラムはプライマリインデックスでインデックスが付けられていますが、結果にはすべてのローまたは 1 つを除くすべてのローが格納されるため、このインデックスを使用しても計算速度は上がりません。

```
SELECT *
FROM Employees
where Employees.EmployeeID <> 102;
```

最適なアクセスプランでは、これは次のように表示されます。`Employees<seq>`

同様に、名前が k という文字で終わるすべての従業員を検索する場合、インデックスは役に立ちません。この結果を計算するには、それぞれのローを個別に調べるしかありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関数

通常、カラム名に対する関数を持つ述部は、検索引数可能ではありません。たとえば、次に示すクエリにはインデックスは使用されません。

```
SELECT *
FROM SalesOrders
WHERE YEAR ( OrderDate ) ='2000';
```

クエリを書き換えて検索引数可能にすると、関数を使用しなくても済みます。たとえば、上記のクエリを次のように書き換えることができます。

```
SELECT *
FROM SalesOrders
WHERE OrderDate > '1999-12-31'
AND OrderDate < '2001-01-01';
```

関数値を計算カラムに格納し、このカラムのインデックスを構築すると、関数を使用するクエリが検索引数可能になります。計算カラムは、テーブル内の他のカラムから値を取得するカラムです。たとえば、注文の日付を保持するカラム OrderDate がある場合は、OrderDate カラムから抽出した年の値を保持する計算カラム OrderYear を作成できます。

```
ALTER TABLE SalesOrders
ADD OrderYear INTEGER
COMPUTE ( YEAR( OrderDate ) );
```

次に、カラム OrderYear のインデックスを通常どおり追加できます。

```
CREATE INDEX IDX_year
ON SalesOrders ( OrderYear );
```

次の文を実行すると、データベースサーバは、情報を保持するインデックスカラムがあることを認識し、そのインデックスを使用してクエリに応答します。

```
SELECT * FROM SalesOrders
WHERE YEAR( OrderDate ) = '2000';
```

計算カラムのドメインは、カラムが置換されるように、COMPUTE 式のドメインと同じにします。上記の例では、`YEAR( OrderDate )` が integer ではなく string を返した場合には、オプティマイザは式の計算カラムを置換しません。その結果、必要なローの取り出しにインデックス IDX_year を使用できなくなってしまいます。

例

次に示す各例では、属性 *x* と *y* は、単一テーブルのそれぞれのカラムです。属性 *z* は、別のテーブルに格納されています。このような属性のそれぞれにインデックスが 1 つ存在することが前提です。

検索引数可能	検索引数不可能
x = 10	*x* < > 10
x IS NULL	
x IS NOT NULL	
x > 25	*x* = 4 OR *y* = 5
x = *z*	*x* = *y*

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

検索引数可能	検索引数不可能
x IN (4, 5, 6)	*x* NOT IN (4, 5, 6)
x LIKE 'pat%'	*x* LIKE '%tern'
x = 20 - 2	*x* + 2 = 20
`X` IS NOT DISTINCT FROM `Y`+1	
`X` IS DISTINCT FROM `Y`+1	

述部が検索引数可能かどうか明らかでない場合があります。この場合、述部を、検索引数可能になるように書き換えることができます。各例では、u はアルファベット順で t の後にくるという事実を利用し、述部 *x* LIKE 'pat%' を *x* >= 'pat' and *x* < 'pau' に書き換えることが可能です。この形式では、属性 `x` のインデックスを使用して、一定範囲内の値を検索できます。幸い、データベースサーバでは、この特殊な変形を自動で行います。

テーブルでのインデックス検索に使用される検索引数可能な述部は、マッチング述部です。WHERE 句には、多くのマッチング述部が含まれることがあります。最も適切な述部はアクセスプランによって異なります。オプティマイザは、アクセスプランの変更を検討するとき、マッチング述部の選択を再評価します。

関連情報

計算カラム [21 ページ]

1.3.1.3 SQL クエリ

このマニュアルの SELECT 文とその他の SQL 文の表記では、それぞれの句を個別の行で示し、SQL キーワードを大文字で示します。

これは文を読みやすくするためであり、必須条件ではありません。SQL キーワードは大文字/小文字のいずれでも入力でき、文のどこでも改行できます。

キーワードと改行

たとえば、次の SELECT 文は、Contacts テーブルからカリフォルニア州内の連絡先の名前と姓を検索します。

```
SELECT GivenName, Surname
FROM Contacts
WHERE State = 'CA';
```

読みやすくはありませんが、この文を次のように入力しても有効です。

```
SELECT GivenName,
Surname from Contacts
WHERE State
 = 'CA';
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

文字列と識別子の大文字と小文字の区別

SQL Anywhere データベースでは、テーブル名、カラム名などの識別子は、大文字と小文字を区別しません。

文字列はデフォルトでは大文字と小文字が区別されないので、'CA'、'ca'、'cA'、'Ca' は同じです。ただし、大文字と小文字を区別するデータベースを作成する場合は、文字列中の大文字/小文字が重要になります。SQL Anywhere サンプルデータベースでは大文字と小文字を区別しません。

識別子の修飾

どのオブジェクトのことを指しているのかはっきりしない場合には、データベース識別子の名前を修飾します。たとえば、SQL Anywhere サンプルデータベースにはカラム City を含んだテーブルがいくつかあるので、City への参照をテーブル名で修飾する必要があります。より規模の大きいデータベースでは、テーブルの所有者の名前を使用してテーブルを識別する場合もあります。

```
SELECT Contacts.City
FROM Contacts
WHERE State = 'CA';
```

ここでの例で示すのは単一テーブルのクエリですから、構文のモデルや例に出てくるカラム名を、カラムが属しているテーブルや所有者の名前で修飾することは通常はありません。

これらの要素は読みやすさを考慮して省略してあるものなので、修飾しても決して間違いではありません。

結果セットのローの順序

結果セットのローの順序に意味はありません。データベースからローが返される順序に保証はなく、またその順序に意味はありません。ローを特定の順序で取り出す場合は、クエリで順序を指定する必要があります。

関連情報

大文字と小文字の区別 [535 ページ]

1.3.1.4 SELECT リスト:カラムの指定

SELECT リストは、通常はカンマで区切った一連のカラム名か、またはすべてのカラムを表すアスタリスク演算子で構成されています。

より一般的には、SELECT リストには、1 つ以上の式がカンマで区切られた形で記述されます。リストの最終カラムの後や、リストにカラムが 1 つしかない場合、カンマはありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SELECT リストの一般的な構文は次のようになります。

```
SELECT expression [, expression  ]..
```

リスト中に、有効な識別子としての規則を満たしていないテーブルやカラムを記述する場合は、それらの識別子を二重引用符で囲んでください。

SELECT リストの式に記述することができるのは、* (すべてのカラム)、カラム名のリスト、文字列、カラムの見出し、算術演算子のある式です。また、集合関数も使用できます。

このセクションの内容:

テーブルからのすべてのカラムの選択 [166 ページ]
SELECT 文の中のアスタリスク (*) には特殊な意味があり、FROM 句で指定されたすべてのテーブルにあるすべてのカラム名を表します。

テーブルからの特定のカラムの選択 [168 ページ]
SELECT 文によって取り出されるカラムは、SELECT キーワードに続けてカラムをリストすることで制限できます。

クエリ結果にある名前が変更されたカラム [170 ページ]
クエリ結果にあるカラムは、いくつかの異なる方法で名前を変更することができます。

クエリ結果の文字列 [171 ページ]
文字列を一重引用符で囲み、それを SELECT リストの他の要素からカンマで区切ると、その文字列をクエリ結果に表示できます。

SELECT リストの計算値 [172 ページ]
SELECT リストの式は、数値カラムのデータを使用して計算を行うことができるため、カラム名や文字列だけではなく、より複雑にできます。

重複したクエリ結果の削除 [175 ページ]
DISTINCT キーワードは、SELECT 文の結果から重複するローを削除します。

関連情報

クエリ結果の要約、グループ化、ソート [355 ページ]

1.3.1.4.1 テーブルからのすべてのカラムの選択

SELECT 文の中のアスタリスク (*) には特殊な意味があり、FROM 句で指定されたすべてのテーブルにあるすべてのカラム名を表します。

1 つのテーブルのカラムをすべて参照したいときにアスタリスクを使用すると、入力する時間が節約でき、入力ミスを防ぐことができます。

SELECT * を使用すると、テーブルが作成されたときに定義された順で、カラムが返されます。

1 つのテーブルのカラムをすべて選択するための構文は次のとおりです。

```
SELECT *
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM table-expression;
```

SELECT * は、1 つのテーブルに現在登録されているカラムをすべて検索するので、カラムの追加、削除、名前の変更など、テーブル構造の変更によって、SELECT * の結果も自動的に変わります。カラムを個別にリストする方が、結果の正確さを確保できます。

例

次の文は Departments テーブルのすべてのカラムを検索します。WHERE 句はありません。そのため、この文はテーブルのすべてのローを検索します。

```
SELECT *
FROM Departments;
```

結果は次のようになります。

DepartmentID	DepartmentName	DepartmentHeadID
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
..	..	..

SELECT キーワードの後ろにテーブルのカラム名をすべて並べても、まったく同じ結果になります。

```
SELECT DepartmentID, DepartmentName, DepartmentHeadID
FROM Departments;
```

次のクエリに示すように、カラム名と同様、* もテーブル名で修飾できます。

```
SELECT Departments.*
FROM Departments;
```

例

ストアドプロシージャに、プロシージャから結果セットをフェッチする際にクエリに * を使用すると、ストアドプロシージャが予期しない結果を返す可能性があります。

たとえば、inner_proc および outer_proc という 2 つのプロシージャを作成します。outer_proc プロシージャで * を使用して、inner_proc プロシージャから結果をフェッチします。

```
CREATE OR REPLACE PROCEDURE inner_proc()
RESULT ( a INT, b INT )
BEGIN
DECLARE not_used INT;
SET not_used = 1;
SELECT 1 AS a, 2 AS b;
END;
```

```
CREATE OR REPLACE PROCEDURE outer_proc()
BEGIN
DECLARE RESULT LONG VARCHAR; -- not used
SELECT * FROM inner_proc();
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
END;
```

次の文を実行します。

```
SELECT * FROM outer_proc()
```

結果は、a b および 1 2 です。

ここで、inner_proc プロシージャを変更して、2 つではなく 3 つのカラムを返すようにします。

```
CREATE OR REPLACE PROCEDURE inner_proc()
RESULT ( a INT, b INT, c INT )
BEGIN
DECLARE not_used INT;
SET not_used = 1;
SELECT 1 as a, 2 as b, 3 as c;
END;
```

次の文をもう一度実行します。

```
SELECT * FROM outer_proc()
```

結果はやはり、a b および 1 2 です。

inner_proc プロシージャの変更後、outer_proc プロシージャは自動的には再コンパイルされません。そのため、outer_proc プロシージャは inner_proc プロシージャが依然として 2 つのカラムを返すものと見なし、上述の最終結果となっています。

inner_proc プロシージャからフェッチするすべてのプロシージャを再コンパイルし、* を使用することが解決法の 1 つとなります。次に例を示します。

```
ALTER PROCEDURE outer_proc RECOMPILE;
```

その他の解決法として、参照プロシージャが inner_proc プロシージャの新しい定義を登録するために、データベースを再起動します。

1.3.1.4.2 テーブルからの特定のカラムの選択

SELECT 文によって取り出されるカラムは、SELECT キーワードに続けてカラムをリストすることで制限できます。

次に例を示します。

```
SELECT Surname, GivenName
FROM Employees;
```

射影と制限

射影とは、テーブル内のカラムのサブセットです。制限 (選択とも言う) は、いくつかの条件に基づいたテーブル内のローのサブセットとのことです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば、次の SELECT 文では、サンプルデータベースで価格が $15 より高い製品すべての名前と価格が検索されます。

```
SELECT Name, UnitPrice
FROM Products
WHERE UnitPrice > 15;
```

このクエリでは、射影 (`SELECT Name, UnitPrice`) と制限 (`WHERE UnitPrice > 15`) を使用しています。

カラムの順番の再調整

カラム名をリストする順番で、カラムが表示される順番が決まります。次の 2 つの例は、表示におけるカラム順の指定方法を示しています。2 つとも、Departments テーブルの 5 つのロー全部から部署名と ID を検出して表示します。ただし、順番が異なります。

```
SELECT DepartmentID, DepartmentName
FROM Departments;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing
..	..

```
SELECT DepartmentName, DepartmentID
FROM Departments;
```

DepartmentName	DepartmentID
R & D	100
Sales	200
Finance	300
Marketing	400
..	..

ジョイン

ジョインは、各テーブルのカラムの値を比較して、2 つ以上のテーブル内のローをリンクします。たとえば、1 ダースを超える数が出荷されたすべての注文項目について、注文項目 ID 番号と製品名を次のように選択できます。

```
SELECT SalesOrderItems.ID, Products.Name
FROM Products JOIN SalesOrderItems
WHERE SalesOrderItems.Quantity > 12;
```

Products テーブルと SalesOrderItems テーブルは、この両テーブル間の外部キー関係に基づいてジョインされます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

ジョイン: 複数テーブルからのデータ検索 [376 ページ]

1.3.1.4.3 クエリ結果にある名前が変更されたカラム

クエリ結果にあるカラムは、いくつかの異なる方法で名前を変更することができます。

デフォルトでは、結果セットの各カラムの見出しは SELECT リストに指定されている式の名前です。式がカラム値の場合、カラムの見出しはカラム名になります。Embedded SQL の場合、DESCRIBE 文を使用して、カーソルによって返される各式の名前を指定できます。また、他のアプリケーションインタフェースでも、インタフェース固有のメカニズムを使用して、結果セットの各カラム名に対するクエリがサポートされています。sa_describe_query システムプロシージャでは、任意の SQL クエリの結果セットのカラム名を指定する、インタフェースに依存しない方法が提供されます。

クエリの SELECT リスト内の式の名前を上書きするには、次のようにエイリアスを使用します。

```
SELECT column-name [ AS ] alias
```

エイリアスを作成すると結果が読みやすくなります。たとえば、次のように、部署リストの DepartmentName を Department に変更できます。

```
SELECT DepartmentName AS Department,
   DepartmentID AS "Identifying Number"
FROM Departments;
```

Department	Identifying Number
R & D	100
Sales	200
Finance	300
Marketing	400
..	..

使用法

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
- 角カッコ
- 逆引用符

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

エイリアスでのスペースとキーワードの使用

上の例では、DepartmentID の "Identifying Number" エイリアスには空白が含まれているため、二重引用符で囲まれています。キーワードまたは特殊文字をエイリアスとして使用する場合も、二重引用符を使用します。たとえば、次のクエリは引用符がないと無効です。

```
SELECT DepartmentName AS Department,
   DepartmentID AS "integer"
FROM Departments;
```

ネームスペースの遮断

エイリアスは、定義されている SELECT ブロックの任意の場所に使用できます。エイリアスは他の SELECT リスト式にも使用できます。この場合は、追加のエイリアスが定義されます。循環的なエイリアスの参照は許可されません。式に指定されたエイリアスが SELECT ブロックのネームスペースにある変数名またはカラム名と同じ場合は、エイリアス定義によってそのカラムまたは変数が遮断されます。次に例を示します。

```
SELECT DepartmentID AS DepartmentName
FROM Departments
WHERE DepartmentName = 'Marketing'
```

この文を実行すると、「値 'Marketing' をデータ型を numeric に変換できません。」というエラーが返されます。これは、クエリの WHERE 句の等号述部が、文字列リテラル "Marketing" と整数カラム DepartmentID を比較しようとしたものの、データ型に互換性がないためです。

> **i 注記**
>
> カラム名を参照する場合、テーブル名によってカラム名を明示的に修飾できます。たとえば、Departments.DepartmentID として、エイリアスと名前が競合しないようにします。

Transact-SQL との互換性

Adaptive Server Enterprise では、SELECT リスト項目のエイリアスを識別するために、ANSI/ISO SQL 標準の AS キーワードと等号の使用の両方がサポートされています。

関連情報

Transact-SQL クエリのサポート [540 ページ]

1.3.1.4.4 クエリ結果の文字列

文字列を一重引用符で囲み、それを SELECT リストの他の要素からカンマで区切ると、その文字列をクエリ結果に表示できます。

文字列を引用符で囲むには、その前に引用符をもう 1 つ記述します。次に例を示します。

```
SELECT 'The department''s name is' AS "Prefix",
   DepartmentName AS Department
FROM Departments;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Prefix	Department
The department's name is	R & D
The department's name is	Sales
The department's name is	Finance
The department's name is	Marketing
The department's name is	Shipping

1.3.1.4.5 SELECT リストの計算値

SELECT リストの式は、数値カラムのデータを使用して計算を行うことができるため、カラム名や文字列だけではなく、より複雑にできます。

算術演算

SELECT リストで実行できる数値演算を説明するには、まずサンプルデータベース内の製品の名前、在庫数、単価をリストすることから始めます。

```
SELECT Name, Quantity, UnitPrice
FROM Products;
```

Name	Quantity	UnitPrice
Tee Shirt	28	9
Tee Shirt	54	14
Tee Shirt	75	14
Baseball Cap	112	9
..	..	..

どの製品も在庫数が 10 になったときに在庫を補充するとします。次のクエリは、各製品をあといくつ売ったら再発注するかをリストします。

```
SELECT Name, Quantity - 10
   AS "Sell before reorder"
FROM Products;
```

Name	Sell before reorder
Tee Shirt	18
Tee Shirt	44
Tee Shirt	65
Baseball Cap	102

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Name	Sell before reorder
..	..

カラムの値を組み合わせることもできます。次のクエリは、在庫中の各製品の総額をリストします。

```
SELECT Name, Quantity * UnitPrice AS "Inventory value"
FROM Products;
```

Name	Inventory value
Tee Shirt	252.00
Tee Shirt	756.00
Tee Shirt	1050.00
Baseball Cap	1008.00
..	..

1 つの式に算術演算子が 2 つ以上ある場合は、乗法、除法、モジュロをまず計算し、その後で減法と加法を計算します。1 つの式のすべての算術演算子の優先度が同じ場合、計算は左から右に順番に行われます。カッコに入っている式は、他のすべての計算より優先されます。

たとえば、次の SELECT 文は、在庫中の各製品の総額を計算し、次にその値から 5 ドル引きます。

```
SELECT Name, Quantity * UnitPrice - 5
FROM Products;
```

確実に正しい結果を得るためには、可能な限りカッコを使用します。次のクエリは前述のクエリと同じ意味で、同じ結果を生成しますが、構文の精度が高くなります。

```
SELECT Name, ( Quantity * UnitPrice ) - 5
FROM Products;
```

算術演算は、演算の結果をそのデータ型で表現できないことによってオーバーフローする場合があります。オーバーフローが発生すると、値の代わりにエラーが返されます。

文字列の演算

文字列連結演算子を使用して文字列を連結できます。|| (ANSI/ISO SQL 標準で定義されている) または + (Adaptive Server Enterprise でサポート) を連結演算子として使用します。たとえば、次の文では、結果の Surname と GivenName の値を取り出して、連結しています。

```
SELECT EmployeeID, GivenName || ' ' || Surname AS Name
FROM Employees;
```

EmployeeID	Name
102	Fran Whitney
105	Matthew Cobb

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

EmployeeID	Name
129	Philip Chin
148	Julie Jordan
..	..

日付と時刻の演算

日付カラムと時刻カラムでも演算子を使用できますが、そのためには一般的に関数を使用することになります。

計算カラムについての注意事項

カラムのエイリアスを指定できる

デフォルトでは、カラム名は SELECT リストにリストされる式ですが、計算カラムの場合、式では煩雑でわかりにくくなります。

その他の演算子を使用できる

乗算演算子はカラムを結合するために使用できます。標準的な算術演算子、論理演算子、文字列演算子など、その他の演算子も使用できます。

たとえば、次のクエリは、すべての顧客のフルネームをリストします。

```
SELECT ID, (GivenName || ' ' || Surname ) AS "Full name"
FROM Customers;
```

|| 演算子は文字列を連結しています。このクエリの場合、カラムのエイリアスにはスペースが付いているので、二重引用符で囲みます。この規則は、カラムのエイリアスだけでなく、データベース内のテーブル名やその他の識別子にも適用されます。

関数を使用できる

カラムを結合するだけでなく、さまざまな組み込み関数を使用して、必要な結果を得ることができます。

たとえば、次のクエリは製品名を大文字でリストします。

```
SELECT ID, UCASE( Name )
FROM Products;
```

ID	UCASE(Products.name)
300	TEE SHIRT
301	TEE SHIRT
302	TEE SHIRT
400	BASEBALL CAP
..	..

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

クエリ結果にある名前が変更されたカラム [170 ページ]

1.3.1.4.6 重複したクエリ結果の削除

DISTINCT キーワードは、SELECT 文の結果から重複するローを削除します。

DISTINCT を指定しないと、重複ローを含むすべてのローが表示されます。オプションとして、SELECT リストの前に ALL を指定すると、すべてのローが表示されます。SQL のその他の実装との互換性のために、SQL Anywhere 構文では、ALL を使用して明示的にすべてのローを要求できるようになっています。ALL がデフォルトです。

たとえば、DISTINCT を指定しないで Contacts テーブルのすべての都市を検索した場合は、60 個のローが表示されます。

```
SELECT City
FROM Contacts;
```

DISTINCT を使用すれば、重複するエントリを削除できます。次のクエリは 16 個のローだけ返します。

```
SELECT DISTINCT City
FROM Contacts;
```

NULL 値は互いに区別されない

DISTINCT キーワードは、複数の NULL 値を互いに重複するものとして処理します。つまり、SELECT 文に DISTINCT が記述されていると、どんなに多くの NULL 値が検出された場合でも、結果には 1 つの NULL しか返されません。

1.3.1.5 FROM 句: テーブルの指定

テーブル、ビュー、またはストアドプロシージャのデータを返すすべての SELECT 文には、FROM 句が必須です。

FROM 句には、2 つ以上のテーブルをリンクする JOIN 条件を記述できるほか、他のクエリ (派生テーブル) へのジョインを記述できます。

テーブル名の修飾

FROM 句では、テーブルとビューについては常に、次のようなフルネームでの構文を作成できます。

```
SELECT select-list
FROM owner.table-name;
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブル名、ビュー名、またはプロシージャ名を修飾する必要があるのは、そのオブジェクトが現在の接続のユーザ ID と異なるユーザ ID によって所有されている場合、または所有者のユーザ ID が現在の接続のユーザ ID が所属するロール名と異なる場合だけです。

相関名の使用

テーブル名に相関名を指定すると、読みやすさが向上し、テーブル名を参照する箇所で毎回完全な名前を入力する手間が省けます。相関名は、次のように、FROM 句でテーブル名の後に入力して割り当てます。

```
SELECT d.DepartmentID, d.DepartmentName
FROM Departments d;
```

相関名が使用される場合、たとえば WHERE 句の中など、そのテーブルに対する他のすべての参照には、テーブル名ではなく、必ず相関名を使用してください。相関名は有効な識別子としての規則に従っている必要があります。

派生テーブルのクエリ

派生テーブルは、クエリ式の評価によって、1 つまたは複数のテーブルから直接または間接的に派生されるテーブルです。派生テーブルは、SELECT 文の FROM 句に定義します。

派生テーブルのクエリは、ビューのクエリと同様に動作します。つまり、派生テーブルの値は、派生テーブル定義の評価時に決定されます。ただし、派生テーブルは、その定義がデータベースに保存されないという点で、ビューと異なります。また、派生テーブルは実体化されず、それが定義されているクエリ外からは参照できないという点で、ベーステーブルやテンポラリテーブルと異なります。

次のクエリでは、各部署の最高給与を取得するために、派生テーブル (my_derived_table) を使用しています。派生テーブルのデータは、Employees テーブルにジョインして、その給与の従業員の姓を取得できます。

```
SELECT Surname,
   my_derived_table.maximum_salary AS Salary,
   my_derived_table.DepartmentID
FROM Employees e,
  ( SELECT MAX( Salary ) AS maximum_salary, DepartmentID
      FROM Employees
      GROUP BY DepartmentID ) my_derived_table
  WHERE e.Salary = my_derived_table.maximum_salary
  AND e.DepartmentID = my_derived_table.DepartmentID
ORDER BY Salary DESC;
```

Surname	Salary	DepartmentID
Shea	138948.00	300
Scott	96300.00	100
Kelly	87500.00	200
Evans	68940.00	400
Martinez	55500.80	500

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の例では、Products テーブルの項目をランクする派生テーブル (MyDerivedTable) を作成し、派生テーブルを問い合わせて、最も低価格の 3 つの項目を返しています。

```
SELECT TOP 3 *
       FROM ( SELECT Description,
                     Quantity,
                     UnitPrice,
                     RANK() OVER ( ORDER BY UnitPrice ASC )
                 AS Rank
                 FROM Products ) AS MyDerivedTable
ORDER BY Rank;
```

テーブル以外のオブジェクトのクエリ

FROM 句内の最も一般的な要素は、テーブル名です。ただし、テーブルに似た構造を持つ (すなわち、適切に定義されたローとカラムを持つ) 他のデータベースオブジェクトからローを問い合わせることも可能です。たとえば、ビューを問い合わせることも、結果セットを返すストアドプロシージャを問い合わせることもできます。

たとえば、次の文は ShowCustomerProducts というストアドプロシージャの結果セットを問い合わせます。

```
SELECT *
FROM ShowCustomerProducts( 149 );
```

このセクションの内容:

DML 文に対する SELECT 文 [177 ページ]
DML 文 (INSERT、UPDATE、DELETE、または MERGE) をクエリの FROM 句内のテーブル式として使用することができます。

関連情報

ジョイン: 複数テーブルからのデータ検索 [376 ページ]

1.3.1.5.1 DML 文に対する SELECT 文

DML 文 (INSERT、UPDATE、DELETE、または MERGE) をクエリの FROM 句内のテーブル式として使用することができます。

文に `dml-derived-table` を含めると、DESCRIBE では無視されます。OPEN 時には、UPDATE 文が最初に実行され、結果がテンポラリテーブルに格納されます。テンポラリテーブルでは、文によって変更されるテーブルのカラム名が使用されます。変更された値は、REFERENCING 句の相関名を使用して参照できます。OLD または FINAL を指定することによって、クエリで参照される更新されたテーブルに、一連のユニークなカラム名を指定する必要がなくなります。`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。

たとえば、次のクエリでは、UPDATE 文に対して SELECT 文を使用し、以下にリストする操作を実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- サンプルデータベース内のすべての製品の値段を 7% 上げるように更新する
- 影響を受ける製品とその注文のうち、2000 年 4 月 10 日から 2000 年 5 月 21 日までの間に出荷され、注文数が 36 より多いものをリストする

```
SELECT old_products.ID, old_products.name, old_products.UnitPrice AS OldPrice,
       final_products.UnitPrice AS NewPrice, SOI.ID AS OrderID, SOI.Quantity
FROM
( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
     REFERENCING ( OLD AS old_products FINAL AS final_products )
  JOIN SalesOrderItems AS SOI ON SOI.ProductID = old_products.ID
WHERE SOI.ShipDate BETWEEN '2000-04-10' AND '2000-05-21'
      AND SOI.QUANTITY > 36
ORDER BY old_products.ID;
```

次のクエリは、MERGE 文と UPDATE 文の両方を使用しています。modified_employees テーブルは、ステータスが変更された従業員のコレクションを表しています。MERGE 文では、給与が 3% 上がった従業員と modified_employees テーブルに含まれている従業員の従業員 ID と名前をマージしています。このクエリで OPTION 句に指定されているオプション設定は、UPDATE 文と MERGE 文の両方に適用されます。

```
CREATE TABLE modified_employees
( EmployeeID INTEGER PRIMARY KEY, Surname VARCHAR(40), GivenName VARCHAR(40) );
MERGE INTO modified_employees AS me
USING (SELECT modified_employees.EmployeeID,
              modified_employees.Surname,
              modified_employees.GivenName
       FROM (
          UPDATE Employees
          SET Salary = Salary * 1.03
          WHERE ManagerID = 501)
            REFERENCING (FINAL AS modified_employees) ) AS dt_e
       ON dt_e.EmployeeID = me.EmployeeID
WHEN MATCHED THEN SKIP
WHEN NOT MATCHED THEN INSERT
OPTION( optimization_level=1, isolation_level=2 );
```

クエリでの複数テーブルの使用

クエリ内で複数の `dml-derived-table` 引数を使用する場合、UPDATE 文の実行順序は保証されません。次の文は、サンプルデータベースの Products テーブルと SalesOrderItems テーブルの両方を更新し、その操作を含むジョインに基づく結果を生成します。

```
SELECT old_products.ID, old_products.name, old_products.UnitPrice AS OldPrice,
       final_products.UnitPrice AS NewPrice,
       SalesOrders.ID AS OrderID, SalesOrders.CustomerID,
       old_order_items.Quantity,
       old_order_items.ShipDate AS OldShipDate,
       final_order_items.ShipDate AS RevisedShipDate
FROM
( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
      REFERENCING ( OLD AS old_products FINAL AS final_products )
  JOIN
( UPDATE SalesOrderItems
  SET ShipDate = DATEADD( DAY, 6, ShipDate )
  WHERE ShipDate BETWEEN  '2000-04-10' AND '2000-05-21' )
      REFERENCING ( OLD AS old_order_items FINAL AS final_order_items )
        ON (old_order_items.ProductID = old_products.ID)
  JOIN SalesOrders ON ( SalesOrders.ID = old_order_items.ID )
WHERE old_order_items.Quantity > 36
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ORDER BY old_products.ID;
```

結果の実体化を伴わないテーブルの使用

REFERENCING (NONE) 句を使用すると、結果を実体化しないで UPDATE 文を埋め込むことができます。この場合は、UPDATE 文の結果が空となるため、意図した結果が返されるようにクエリを記述します。空ではない結果が返されるようにするには、外部ジョインの NULL 入力側に dml-derived-table を指定します。次に例を示します。

```
SELECT 'completed' AS finished, ( SELECT COUNT( * ) FROM Products ) AS product_total
FROM SYS.DUMMY LEFT OUTER JOIN
    ( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
      REFERENCING ( NONE ) ON 1=1;
```

また、集合演算子 (UNION、EXCEPT、または INTERSECT) のいずれかを使用するクエリ式の一部として dml-derived-table を使用することによって、空ではない結果が返されるようにすることもできます。次に例を示します。

```
SELECT 'completed' AS finished, ( SELECT COUNT( * ) FROM Products ) AS product_total
FROM SYS.DUMMY
UNION ALL
SELECT 'dummy', 1 /* This query specification returns the empty set */
FROM ( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
      REFERENCING ( NONE );
```

関連情報

データ操作文 [498 ページ]

1.3.1.6 WHERE 句: ローの指定

SELECT 文の WHERE 句は、ローの取得時にデータベースサーバが適用する必要がある検索条件を指定します。

検索条件は述部とも呼ばれます。一般的なフォーマットは次のとおりです。

```
SELECT select-list
FROM table-list
WHERE search-condition
```

WHERE 句の探索条件には、次のものがあります。

比較演算子

(=、<、> など) たとえば、給料が $50,000 を上回る従業員全員をリストする場合は、次のようになります。

```
SELECT Surname
   FROM Employees
   WHERE Salary > 50000;
```

範囲

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

(BETWEEN と NOT BETWEEN) たとえば、給料が $40,000 〜 $60,000 の従業員をすべてリストする場合は、次のようになります。

```
SELECT Surname
   FROM Employees
   WHERE Salary BETWEEN 40000 AND 60000;
```

リスト

(IN、NOT IN) たとえば、オンタリオ州、ケベック州、またはマニトバ州の顧客をすべてリストする場合は、次のようになります。

```
SELECT CompanyName, State
   FROM Customers
   WHERE State IN( 'ON', 'PQ', 'MB');
```

文字の一致

(LIKE と NOT LIKE) たとえば、電話番号が 415 ではじまる顧客をすべてリストする場合は、次のようになります (データベースでは、電話番号は文字列として保存されています)。

```
SELECT CompanyName, Phone
   FROM Customers
   WHERE Phone LIKE '415%';
```

不定の値

(IS NULL と IS NOT NULL) たとえば、マネージャがいる部署をすべてリストする場合は、次のようになります。

```
SELECT DepartmentName
   FROM Departments
   WHERE DepartmentHeadID IS NOT NULL;
```

組み合わせ

(AND、OR) たとえば、給料が $50,000 を上回り、名前が A で始まるすべての従業員をリストする場合は、次のようになります。

```
SELECT GivenName, Surname
   FROM Employees
   WHERE Salary > 50000
   AND GivenName like 'A%';
```

このセクションの内容:

WHERE 句での比較演算子 [181 ページ]
WHERE 句で比較演算子が使用できます。

WHERE 句での範囲 [182 ページ]
BETWEEN キーワードは包括的範囲を指定します。包括的範囲には、その範囲の間の値だけではなく、上限値と下限値も含まれます。

WHERE 句でのリスト [183 ページ]
IN キーワードは、値のリストのいずれかに一致する値を選択するためのキーワードです。

WHERE 句での文字列のパターン一致 [184 ページ]
WHERE 句でパターン一致を使用すると、検索条件を拡張することができます。

文字列と引用符 [187 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

文字データや日付データを入力したり、検索したりするときは、一重引用符で囲む必要があります。

不定の値: NULL [187 ページ]
: カラムに NULL 値がある場合は、ユーザまたはアプリケーションがそのカラムに何も入力していないことを意味します。

カラム値を NULL と比較する [188 ページ]
: IS NULL 検索条件を使用すると、カラム値を NULL と比較したり、その比較の結果に基づいてカラム値を選択したりするなど、特定の動作を実行できます。

NULL のプロパティ [189 ページ]
: NULL 値のプロパティは、いくつかの方法で拡張することができます。

条件を接続する論理演算子 [190 ページ]
: 論理演算子 AND、OR、NOT を使用して、WHERE 句で複数の検索条件を接続します。

日付を比較する探索条件 [191 ページ]
: 演算子 =、<、および > を使用して、日付を比較することができます。

音によるローの一致 [192 ページ]
: SOUNDEX 関数を使用すると、ローを発音で対応させることができます。

1.3.1.6.1 WHERE 句での比較演算子

WHERE 句で比較演算子が使用できます。

演算子は次の構文に従います。

```
WHERE expression comparison-operator expression
```

比較についての注意事項

ソート順

文字データの比較の場合、< はソート順の前の方、> はソート順の後の方であるという意味です。ソート順は、データベースを作成するときに選択した照合順によって決まります。データベースに対して dbinfo コマンドラインユーティリティを実行すると、照合順を確認できます。

```
dbinfo -c "uid=DBA;pwd=sql"
```

SQL Central からもデータベースのプロパティの詳細情報タブに移動して、照合順を確認することができます。

後続ブランク

データベースの作成時には、比較処理のために後続ブランクを無視するかどうかを示します。

デフォルトでは、後続ブランクを無視しないでデータベースを作成します。たとえば、'Dirk' は 'Dirk ' と同じではありません。後続ブランクが無視されるように、ブランクを埋め込んだデータベースを作成できます。

日付の比較

日付を比較するときには、< はより古いことを意味し、> はより新しいことを意味します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

大文字と小文字の区別

データベースの作成時に、文字列比較が大文字と小文字を区別するかどうかを指定します。

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。たとえば、'Dirk' は 'DIRK' と同じです。大文字と小文字を区別するように、データベースを作成できます。

次は比較演算子を使用する SELECT 文です。

```
SELECT *
   FROM Products
   WHERE Quantity < 20;
SELECT E.Surname, E.GivenName
   FROM Employees E
   WHERE Surname > 'McBadden';
SELECT ID, Phone
   FROM Contacts
   WHERE State  != 'CA';
```

NOT 演算子

NOT 演算子は式を否定します。次の 2 つのクエリはどちらも、単価が $10 以下の T シャツ (Tee Shirt) と野球帽 (BaseBall Cap) をすべて検索します。ただし、否定の論理演算子 (NOT) と否定の比較演算子 (!>) では、位置が異なることに注意してください。

```
SELECT ID, Name, Quantity
   FROM Products
   WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
   AND NOT UnitPrice > 10;
SELECT ID, Name, Quantity
   FROM Products
   WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
   AND UnitPrice !> 10;
```

1.3.1.6.2 WHERE 句での範囲

BETWEEN キーワードは包括的範囲を指定します。包括的範囲には、その範囲の間の値だけではなく、上限値と下限値も含まれます。

NOT BETWEEN を使用すると、この範囲内にないローをすべて検出できます。

例

- 次のクエリは、$10 以上 $15 以下の価格の製品をすべてリストします。

```
SELECT Name, UnitPrice
   FROM Products
   WHERE UnitPrice BETWEEN 10 AND 15;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Name	UnitPrice
Tee Shirt	14
Tee Shirt	14
Baseball Cap	10
Shorts	15

- 次のクエリは、$10 未満か、$15 を超える製品をすべてリストします。

```
SELECT Name, UnitPrice
   FROM Products
   WHERE UnitPrice NOT BETWEEN 10 AND 15;
```

Name	UnitPrice
Tee Shirt	9
Baseball Cap	9
Visor	7
Visor	7
..	..

1.3.1.6.3 WHERE 句でのリスト

IN キーワードは、値のリストのいずれかに一致する値を選択するためのキーワードです。

式は定数またはカラム名であり、リストは、定数のセット、またはより一般的には、サブクエリです。

たとえば、IN を使用せずに、オンタリオ州、マニトバ州、またはケベック州内の顧客すべての名前と州名をリストする場合は、次のようなクエリを入力します。

```
SELECT CompanyName, State
   FROM Customers
   WHERE State = 'ON' OR State = 'MB' OR State = 'PQ';
```

ただし、得られる結果は IN を使用したときと同じになります。IN キーワードに続く項目は、カンマで区切り、カッコで囲んでください。文字、日付、時刻の値の前後に一重引用符を入力します。次に例を示します。

```
SELECT CompanyName, State
   FROM Customers
   WHERE State IN( 'ON', 'MB', 'PQ');
```

おそらく、IN キーワードの最も重要な用途は、サブクエリとも呼ばれる、ネストされたクエリの中で使用される場合です。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.6.4 WHERE 句での文字列のパターン一致

WHERE 句でパターン一致を使用すると、検索条件を拡張することができます。

SQL では、パターンの検索には LIKE キーワードを使用します。パターン一致では、ワイルドカード文字を使用して、文字のさまざまな組み合わせに対応します。

LIKE キーワードは、その後に入力された文字列がマッチングパターンであると指定します。LIKE は文字データとともに使用します。

LIKE の構文は、次のとおりです。

```
expression [ NOT ] LIKE match-expression
```

一致する式は、次の特殊記号を含むマッチ式と比較されます。

記号	意味
%	文字数が 0 以上の文字列に一致します。
_	文字 1 個に一致します。
[specifier]	カッコ内の指定子の形式は、次のとおりです。 範囲 範囲のフォームは rangespec1-rangespec2 です。rangespec1 は文字の範囲の始点を指し、ハイフンは範囲、rangespec2 は文字範囲の終点を指します。 セット セットには、任意の順序での不連続な値の集合が含まれます。たとえば、[a2bR] などです。 範囲 [a-f] と、セット [abcdef] と [fcbdae] は、同じ値セットを返します。
[^specifier]	指定子の前にある脱字記号 (^) は非包含を表します。[^a-f] は a ～ f の範囲ではないとの意味で、[^a2bR] は a、2、b、R ではないとの意味です。

カラムデータを、定数、変数、またはテーブルに示されているワイルドカード文字を含むその他のカラムに一致させることができます。定数を使用するときは、マッチ文字列と文字列を一重引用符で囲みます。

例

次の例はすべて、Contacts テーブルの Surname カラムで LIKE を使用しています。クエリの形式は次のとおりです。

```
SELECT Surname
   FROM Contacts
   WHERE Surname LIKE match-expression;
```

最初の例は次のように入力します。

```
SELECT Surname
   FROM Contacts
   WHERE Surname LIKE 'Mc%';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

マッチ式	説明	戻り値
'Mc%'	Mc で始まるすべての名前を検索します。	`McEvoy`
'%er'	er で終わるすべての名前を検索します。	`Brier, Miller, Weaver, Rayner`
'%en%'	en を含むすべての名前を検索します。	`Pettengill, Lencki, Cohen`
'_ish'	ish で終わるすべての 4 文字の名前を検索します。	`Fish`
'Br[iy][ae]r'	Brier、Bryer、Briar、または Bryar を検索します。	`Brier`
'[M-Z]owell'	M ～ Z の範囲の 1 文字で始まり、owell で終わるすべての名前を検索します。	`Powell`
'M[^c]%'	M で始まり、2 番目の文字が c ではないすべての名前を検索します。	`Moore, Mulley, Miller, Masalsky`

ワイルドカードには LIKE が必須

ワイルドカード文字を LIKE なしで使用すると、パターンとは解釈されず、文字列リテラルと解釈されます。つまり、ワイルドカード文字そのものの値を表すことになります。次のクエリは、415% の 4 文字だけから成る電話番号を検出しようとします。415 で始まる電話番号は検出しません。

```
SELECT Phone
   FROM Contacts
   WHERE Phone = '415%';
```

日付値と時刻値での LIKE の使用

LIKE は、DATE、TIME、TIMESTAMP、TIMESTAMP WITH TIME ZONE の各フィールドに対して使用できます。ただし、LIKE 述部は文字データでのみ使用できます。日付値や時刻値で LIKE を使用すると、これらの値は、DATE、TIME、TIMESTAMP、TIMESTAMP WITH TIME ZONE の各データ型の対応するオプション設定を使用して、暗黙的に CHAR または VARCHAR にキャストされ、値がフォーマットされます。

日付/時刻	VARCHAR への CAST で使用
DATE	date_format
TIME	time_format

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

日付/時刻	VARCHAR への CAST で使用
TIMESTAMP	timestamp_format
TIMESTAMP WITH TIME ZONE	timestamp_with_time_zone_format

日付値と時刻値にはさまざまな日付部分があり、上記のオプションに基づいてそれぞれ異なる方法でフォーマットされるため、DATE、TIME、または TIMESTAMP の値を検索するときに LIKE を使用する場合、検索に成功するには LIKE パターンを慎重に書き込む必要があります。

たとえば、TIMESTAMP カラム arrival_time に値 9:20 と現在の日付を入力した場合、timestamp_format オプションによってコロンを使用して時間と分を区切るように値の時刻部分がフォーマットされると、次の句は TRUE と評価されます。

```
WHERE arrival_time LIKE '%09:20%'
```

LIKE とは異なり、文字列リテラルと DATE、TIME、TIMESTAMP、または TIMESTAMP WITH TIME ZONE の各値の単純な比較を含む検索条件では、比較ドメインとして日付/時刻データ型が使用されます。この場合、データベースサーバでは、まず文字列リテラルを TIMESTAMP 値に変換してから、その値の必要な部分を使用して比較を実行します。TIME、DATE、TIMESTAMP 値の変換について、SQL Anywhere は ISO 8601 標準に従っており、追加の拡張が行われています。

たとえば、定数文字列値 9:20 は 9:20 を時刻部分および日付部分の現在の日付として使用して TIMESTAMP に変換されるため、次の句は TRUE と評価されます。

```
WHERE arrival_time = '9:20'
```

NOT LIKE の使用

LIKE とともに使用できるのと同じワイルドカード文字を、NOT LIKE にも使用できます。次のクエリのいずれかを使用すると、Contacts テーブル内で市外局番が 415 ではない電話番号をすべて検索します。

```
SELECT Phone
   FROM Contacts
   WHERE Phone NOT LIKE '415%';
```

```
SELECT Phone
   FROM Contacts
   WHERE NOT Phone LIKE '415%';
```

アンダースコアの使用

LIKE とともに使用できるもう 1 つの特殊文字は _ (アンダースコア) 文字です。この特殊文字は、厳密に 1 つの文字と対応します。たとえば、パターン 'BR_U%' は、BR で始まり 4 番めの文字が U であるすべての名前と対応します。Braun では、_ は文字 A に対応し、% は N に対応します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.6.5 文字列と引用符

文字データや日付データを入力したり、検索したりするときは、一重引用符で囲む必要があります。

次に例を示します。

```
SELECT GivenName, Surname
   FROM Contacts
   WHERE GivenName = 'John';
```

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は (デフォルトでは On)、次の例のように二重引用符を使用して文字や日付のデータを囲むこともできます。

```
SET OPTION quoted_identifier = 'Off';
```

quoted_identifier オプションは、Adaptive Server Enterprise との互換性のために提供されています。デフォルトでは、Adaptive Server Enterprise オプションは quoted_identifier が Off で、SQL Anywhere オプションは quoted_identifier が On です。

文字列での引用符

文字エントリ内でリテラル引用符を指定する方法は 2 つあります。1 つめの方法は、引用符を 2 回連続して使用する方法です。たとえば、一重引用符で文字列を開始し、そのエントリの一部として一重引用符を 1 つ入力する場合は、一重引用符を 2 つ使用します。

```
'I don''t understand.'
```

二重引用符の場合は次のように指定します (quoted_identifier が Off の場合)。

```
"He said, ""It is not really confusing."""
```

2 つめの方法は、quoted_identifier が Off の場合にしか使用できませんが、引用符を別の種類の引用符で囲む方法です。つまり、二重引用符が入っているエントリは一重引用符で囲み、逆に一重引用符が入っているエントリは二重引用符で囲みます。次にその例を示します。

```
'George said, "There must be a better way."'
"Isn't there a better way?"
'George asked, "Isn''t there a better way?"'
```

1.3.1.6.6 不定の値: NULL

カラムに NULL 値がある場合は、ユーザまたはアプリケーションがそのカラムに何も入力していないことを意味します。

つまり、このカラムのデータ値は、不定か、使用不可です。

NULL は 0 (数値) やブランク (文字値) と同じではありません。むしろ、NULL 値によって、数値カラムの場合の 0 や文字カラムの場合のブランクなどの意図的な入力と、入力がない場合とを区別できます。入力が行われていない場合は、数値カラムと文字カラムは両方とも NULL です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

NULL の入力

NULL は、NULL 値が許可されたカラムにのみ入力できます。カラムに NULL 値が許可されるかどうかは、テーブルの作成時に決まります。カラムに NULL 値が許可されることを前提として、NULL を挿入します。

デフォルト

データが入力されず、カラムに他のデフォルト設定がない場合。

明示的な入力

引用符なしで NULL と明示的に挿入できます。NULL という単語を引用符をつけて文字カラムに入力すると、NULL は NULL 値としてではなく、1 つのデータとして処理されます。

たとえば、Departments テーブルの DepartmentHeadID カラムは NULL 値を許可します。次のように、マネージャのいない部署のローを 2 種類入力できます。

```
INSERT INTO Departments (DepartmentID, DepartmentName)
   VALUES (201, 'Eastern Sales')
INSERT INTO Departments
   VALUES (202, 'Western Sales', NULL);
```

NULL 値を返す

NULL 値は他の値とまったく同じように、クライアントアプリケーションに返され、表示されます。たとえば、次の例では、Interactive SQL で NULL 値がどのように表示されるかを示しています。

```
SELECT *
FROM Departments;
```

DepartmentID	DepartmentName	DepartmentHeadID
100	R & D	501
200	Sales	904
300	Finance	1293
400	Marketing	1576
500	Shipping	703
201	Eastern Sales	(NULL)
202	Western Sales	(NULL)

1.3.1.6.7 カラム値を NULL と比較する

IS NULL 検索条件を使用すると、カラム値を NULL と比較したり、その比較の結果に基づいてカラム値を選択したりするなど、特定の動作を実行できます。

TRUE の値を返すカラムだけが、選択されたり、結果として指定の動作を行ったりします。FALSE や UNKNOWN を返すカラムの場合、そういうことはありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の例では、UnitPrice が $15 未満または NULL のローだけを選択します。

```
SELECT Quantity, UnitPrice
   FROM Products
   WHERE UnitPrice < 15
   OR UnitPrice IS NULL;
```

NULL が指定の値や別の NULL に等しいか (または等しくないか) わからないので、NULL 比較の結果はすべて UNKNOWN になります。

true を決して返さない条件があり、そうした条件を使用するクエリは結果セットを返しません。たとえば、NULL は不定の値があるという意味であるため、次の比較は決して true とは見なされません。

```
WHERE column1 > NULL
```

WHERE 句でカラム名を 2 つ使用する場合、つまり 2 つのテーブルをジョインする場合にも、この論理は適用されます。条件 `WHERE column1 = column2` を含む句は、カラムに NULL が含まれるローを返しません。

次のパターンで NULL や NOT NULL も検索できます。

```
WHERE column_name IS NULL
```

```
WHERE column_name IS NOT NULL
```

次に例を示します。

```
WHERE advance < $5000
OR advance IS NULL
```

1.3.1.6.8 NULL のプロパティ

NULL 値のプロパティは、いくつかの方法で拡張することができます。

NULL 値のプロパティの拡張について、次の一覧で説明します。

FALSE と UNKNOWN の相違

FALSE と UNKNOWN はどちらも値を返しませんが、FALSE と UNKNOWN の間には大きな論理的相違があります。false の反対 ("not false") は true ですが、UNKNOWN の反対は何かを知っていることを意味しません。たとえば、`1 = 2` は false に評価され、`1 != 2` (1 は 2 に等しくない) は true に評価されます。

ただし、比較の中に NULL がある場合は、式を否定しても、反対のローセットや反対の真の値は得られません。UNKNOWN 値は UNKNOWN のままです。

複数の NULL 値を 1 つの値で置き換える

ISNULL 組み込み関数を使用して、複数の NULL 値を 1 つの特定の値と置き換えることができます。置換は表示目的のためだけに実行されます。実際のカラム値には影響しません。構文は次のとおりです。

```
ISNULL( expression, value  )
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば、次の文を使用して Departments のすべてのローを選択し、すべての NULL 値を -1 という値でカラム DepartmentHeadID に表示します。

```
SELECT DepartmentID,
       DepartmentName,
       ISNULL( DepartmentHeadID, -1 ) AS DepartmentHead
   FROM Departments;
```

NULL と評価される式

オペランドのいずれかが NULL 値の場合、算術演算子またはビット処理演算子のある式は NULL と評価されます。たとえば `1 + column1` は、column1 が NULL であれば NULL と評価されます。

文字列と NULL の連結

文字列と NULL を連結すると、式は文字列と評価されます。たとえば、次の文は文字列 abcdef を返します。

```
SELECT 'abc' || NULL || 'def';
```

1.3.1.6.9 条件を接続する論理演算子

論理演算子 AND、OR、NOT を使用して、WHERE 句で複数の検索条件を接続します。

1 つの文で複数の論理演算子が使用されている場合、通常は、AND 演算子が OR 演算子の前に評価されます。実行順序はカッコを使用して変更できます。

AND の使用

AND 演算子は 2 つ以上の条件を結合し、それらの条件のすべてが true である場合にだけ、結果を返します。たとえば次のクエリは、連絡先の姓が Purcell で、名前が Beth のローだけを検出します。

```
SELECT *
   FROM Contacts
   WHERE GivenName = 'Beth'
      AND Surname = 'Purcell';
```

OR の使用

OR 演算子は 2 つ以上の条件を結合し、それらの条件のうち、いずれかが true の場合に、結果を返します。次のクエリは、GivenName カラムの中で Elizabeth の別名を含むローを検索します。

```
SELECT *
   FROM Contacts
   WHERE GivenName = 'Beth'
      OR GivenName = 'Liz';
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

NOT の使用

NOT 演算子は、その後に来る式を否定します。次のクエリは、カリフォルニア州以外の連絡先をすべてリストします。

```
SELECT *
   FROM Contacts
   WHERE NOT State = 'CA';
```

1.3.1.6.10 日付を比較する探索条件

演算子 =、<、および > を使用して、日付を比較することができます。

例

Interactive SQL で次のクエリを実行して、1964 年 3 月 13 日より前に生まれたすべての従業員をリストします。

```
SELECT Surname, BirthDate
   FROM Employees
   WHERE BirthDate < 'March 13, 1964'
   ORDER BY BirthDate DESC;
```

Surname	BirthDate
Ahmed	1963-12-12
Dill	1963-07-19
Rebeiro	1963-04-12
Garcia	1963-01-23
Pastor	1962-07-14
..	..

注記

日付への自動変換

データベースサーバは BirthDate カラムに日付が入っていることを認識して、'March 13, 1964' の文字列を自動的に日付に変換します。

日付の指定方法

日付を指定するには、さまざまな方法があります。次に例を示します。

```
'March 13, 1964'
'1964/03/13'
'1964-03-13'
```

date_order オプションデータベースオプションを設定して、クエリに含まれる日付の解釈を設定できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

yyyy/mm/dd または yyyy-mm-dd フォーマットの日付は、date_order 設定に関係なく、常に日付として明確に認識されます。

その他の比較演算子

複数の比較演算子がサポートされています。

1.3.1.6.11 音によるローの一致

SOUNDEX 関数を使用すると、ローを発音で対応させることができます。

たとえば、電話メッセージが残されていて、その宛先が Ms. Brown のように発音されていたとします。次のクエリを実行して、Brown のように聞こえる名前を持つ従業員を検索することができます。

注記

SOUNDEX によって使用されるアルゴリズムは、主に英語版データベースを対象としています。

例

Interactive SQL で次のクエリを実行して、Brown のように聞こえる姓の従業員をリストします。

```
SELECT Surname, GivenName
   FROM Employees
   WHERE SOUNDEX( Surname ) = SOUNDEX( 'Brown' );
```

Surname	GivenName
Braun	Jane

1.3.1.7 ORDER BY 句: 結果の順序付け

特に指定しないかぎり、データベースサーバは、テーブルのローを意味のない順序で返します。

テーブルのローは、多くの場合、意味のある順序にした方が便利です。たとえば、製品をアルファベット順に見たいとします。

構文を使用して SELECT 文の末尾に ORDER BY 句を追加し、結果セットのローの順序を指定します。

```
SELECT column-name-1, column-name-2,..
FROM table-name
ORDER BY order-by-column-name
```

column-name-1、column-name-2、table-name を、問い合わせるカラムとテーブルの名前に置き換えてください。order-by-column-name はテーブルのカラムです。テーブルのすべてのカラムを表す省略形としてアスタリスクを使用できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

注記

句の順序が重要

ORDER BY 句は、FROM 句と SELECT 句の後に指定してください。

昇順または降順を指定できる

デフォルトの順序は昇順です。次のクエリのように句の末尾にキーワード DESC を追加すると、降順を指定できます。

```
SELECT ID, Quantity
   FROM Products
   ORDER BY Quantity DESC;
```

ID	Quantity
400	112
700	80
302	75
301	54
600	39
..	..

複数のカラム順に順序を設定できる

次のクエリは、最初にサイズ順 (アルファベット順)、次に名前順にソートします。

```
SELECT ID, Name, Size
   FROM Products
   ORDER BY Size, Name;
```

ID	Name	Size
600	Sweatshirt	Large
601	Sweatshirt	Large
700	Shorts	Medium
301	Tee Shirt	Medium
..	..	..

ORDER BY カラムが **SELECT** リストになくてもよい

次のクエリは、価格が結果セットになくても、製品を単価順にソートします。

```
SELECT ID, Name, Size
   FROM Products
   ORDER BY UnitPrice;
```

ID	Name	Size
500	Visor	One size fits all
501	Visor	One size fits all

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Name	Size
300	Tee Shirt	Small
400	Baseball Cap	One size fits all
..	..	..

ORDER BY 句を使用せず、クエリを複数回実行すると、異なる結果が表示される可能性がある

この理由は、データベースサーバが同じ結果セットを異なる順序で返す可能性があるためです。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。これは、結果セットの提示が、最後にアクセスしたローとその他の要因によって変化する可能性があることを意味します。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。

例

Interactive SQL で次の文を実行して、製品をアルファベット順にリストします。

```
SELECT ID, Name, Description
   FROM Products
   ORDER BY Name;
```

ID	Name	Description
400	Baseball Cap	Cotton Cap
401	Baseball Cap	Wool cap
700	Shorts	Cotton Shorts
600	Sweatshirt	Hooded Sweatshirt
..	..	..

1.3.1.8 ORDER BY のパフォーマンスを改善するインデックス

インデックスを使用すると、データベースサーバがより効率的にテーブルを検索できるようになります。

WHERE 句と ORDER BY 句を使ったクエリ

複数の実行方法があるクエリの一例は、WHERE 句と ORDER BY 句の両方を含むクエリです。

```
SELECT *
   FROM Customers
   WHERE ID > 300
   ORDER BY CompanyName;
```

この例で、データベースサーバは次の 2 つの方法のどちらを採用するか決定する必要があります。

1. Customers テーブル全体を、会社名の順序で検索し、各ローの顧客の ID の値が 300 以上かどうかをチェックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. ID カラムのキーを使用して、300 を超える ID を持つ会社だけを読み込みます。結果は会社名順にソートされます。

ID 値が 300 を超える会社がほとんどない場合は、2 番目の方法の方が優れています。スキャンするローの数が少なく、ソートにも時間がかからないからです。大半の会社の ID 値が 300 を超える場合は、ソートの必要がない最初の方法の方がはるかに優れています。

問題の解決

ID と CompanyName の 2 つのカラムでインデックスを作成すれば、上の例を解決できます。データベースサーバは、このインデックスを使用してテーブルからローを適切な順序で選択できます。ただし、インデックスはデータベースファイルの領域を消費し、更新にオーバヘッドがかかることは覚えておく必要があります。インデックスの作成は慎重に行ってください。

1.3.1.9 クエリの集合関数

集合関数および GROUP BY 句を使用すると、テーブル内の、個別のローではなくローグループのプロパティを反映するデータの内容を検査するのに役立ちます。

たとえば、顧客が注文した総額の平均値や、各部門で何人の従業員が仕事をしているかなどです。この種のタスクには、集合関数と GROUP BY 句を使用します。

関数 COUNT、MIN、MAX は集合関数です。この 3 つの関数は、それぞれ情報を要約します。その他の集合関数として、AVG、STDDEV、VARIANCE などの統計関数があります。COUNT 以外のすべての集合関数では、パラメータが必要です。

集合関数は、一連のローについて単一の値を返します。GROUP BY 句がない場合、集合関数はスカラ集合関数と呼ばれ、クエリの他の条件を満たすすべてのローについて、単一の値を返します。GROUP BY 句がある場合、集合関数はベクトル集合関数と呼ばれ、グループごとに 1 つの値を返します。

OLAP 関数と呼ばれることもある分析用の追加の集合関数がサポートされています。それらの関数のいくつかは Window 関数として使用されます。これには、RANK、PERCENT_RANK、CUME_DIST、ROW_NUMBER、線形回帰分析をサポートするための関数があります。

例

社内の従業員数をリストするには、Interactive SQL で次のクエリを実行します。

```
SELECT COUNT( * )
   FROM Employees;
```

COUNT()
75

この結果セットは、タイトル COUNT(*) を持つ 1 つのカラムと、従業員の合計数が入った 1 つのローで構成されています。

社内の従業員数と、最年長の従業員および最年少の従業員の生年月日をリストするには、Interactive SQL で次のクエリを実行します。

```
SELECT COUNT( * ), MIN( BirthDate ), MAX( BirthDate )
   FROM Employees;
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

COUNT()	MIN(Employees.BirthDate)	MAX(Employees.BirthDate)
75	1936/01/02	1973/01/18

このセクションの内容:

グループ分けされたデータに対する集合関数の使用方法 [196 ページ]
集合関数を使用して、ローグループで計算を実行することができます。

HAVING 句: グループ内のローの制限 [198 ページ]
HAVING 句を使用して、グループ内のローを制限することができます。

WHERE 句と HAVING 句の組合せ [199 ページ]
WHERE 句または HAVING 句のいずれかを使用して、同じローセットを指定できます。

関連情報

OLAP のサポート [435 ページ]

1.3.1.9.1 グループ分けされたデータに対する集合関数の使用方法

集合関数を使用して、ローグループで計算を実行することができます。

GROUP BY 句は、ローをグループ単位で配置し、集合関数はローグループごとに 1 つの値を返します。

空のセットの意味の違い

SQL 言語では、集合関数を使用するとき、空のセットを扱う方法が異なります。GROUP BY 句がない場合、0 より大きい入力ローの集合関数があるクエリでは、結果として単一のローが返されます。COUNT の場合、結果は値 0 となり、その他のすべての集合関数では、結果は NULL となります。ただし、クエリに GROUP BY 句があり、クエリへの入力が空の場合、クエリの結果は空となり、ローは返されません。

たとえば、次のクエリでは値 0 の単一のローが返されます。部署 103 に従業員はいません。

```
SELECT COUNT() FROM Employees WHERE DepartmentID = 103;
```

ただし、この変更したクエリでは、GROUP BY 句が存在するためにローは返されません。

```
SELECT COUNT() FROM Employees WHERE DepartmentID = 103 GROUP BY State;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

GROUP BY 句を使用する場合の一般的なエラー

GROUP BY 句を使用する場合の一般的なエラーは、1 つのグループにまとめることができない情報を得ようとすることです。たとえば、次のクエリではエラーが発生します。

```
SELECT SalesRepresentative, Surname, COUNT( * )
   FROM SalesOrders KEY JOIN Employees
   GROUP BY SalesRepresentative;
```

エラーメッセージには、GROUP BY 句にも Surname カラムへの参照が指定される必要があることが示されています。このエラーは、指定された ID を持つ従業員に対するそれぞれの結果ローで、姓が同じであることをデータベースサーバが検証できないことによって発生します。

このエラーを解決するには、GROUP BY 句にカラムを追加します。

```
SELECT SalesRepresentative, Surname, COUNT( * )
   FROM SalesOrders KEY JOIN Employees
   GROUP BY SalesRepresentative, Surname
   ORDER BY SalesRepresentative;
```

この方法が適切でない場合は、代わりに集合関数を使用して、値を 1 つだけ選択できます。

```
SELECT SalesRepresentative, MAX( Surname ), COUNT( * )
   FROM SalesOrders KEY JOIN Employees
   GROUP BY SalesRepresentative
   ORDER BY SalesRepresentative;
```

MAX 関数は、各グループのディテールローから最大 (アルファベット順の最後) の姓 (Surname) を選択します。最大値は 1 つしかないので、この文は有効です。この場合は、グループ内のすべてのディテールローに同じ姓が表示されます。

例

Interactive SQL で次のクエリを実行して、営業担当と各担当が受けた注文数をリストします。

```
SELECT SalesRepresentative, COUNT( * )
   FROM SalesOrders
   GROUP BY SalesRepresentative
   ORDER BY SalesRepresentative;
```

SalesRepresentative	COUNT()
129	57
195	50
299	114
467	56
..	..

GROUPBY 句はデータベースサーバに対して、もしこれがなければ返されるすべてのローのセットを分割するように指示します。各分割、つまりグループに含まれるすべてのローは、指定のカラムに同じ値を持ちます。ユニークな値ごと、または値セットごとに、1 つだけグループがあります。この場合、各グループに含まれるすべてのローの SalesRepresentative 値が同じになります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

COUNT などの集合関数は、各グループのローに適用されます。したがって、この結果セットには各グループのローの合計数が表示されます。クエリの結果は、各営業担当者の ID 番号が入った 1 つのローで構成されています。各ローには、営業担当者 ID と、その担当者の受注の合計数が入ります。

GROUP BY を使用した場合には、結果テーブルには GROUP BY 句に指定したカラムまたはカラムセットごとにローが 1 つずつあります。

関連情報

GROUP BY 句: クエリ結果のグループへの編成 [360 ページ]
集合関数を伴う GROUP BY [363 ページ]

1.3.1.9.2 HAVING 句: グループ内のローの制限

HAVING 句を使用して、グループ内のローを制限することができます。

例

Interactive SQL 句で次の文を実行して、55 件を超える注文を持っているすべての営業担当をリストします。

```
SELECT SalesRepresentative, COUNT( * ) AS orders
FROM SalesOrders KEY JOIN Employees
GROUP BY SalesRepresentative
HAVING count( * ) > 55
ORDER BY orders DESC;
```

SalesRepresentative	orders
299	114
129	57
1142	57
467	56

関連情報

HAVING 句: データグループの選択 [365 ページ]
HAVING 句でのサブクエリ [482 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.9.3 WHERE 句と HAVING 句の組合せ

WHERE 句または HAVING 句のいずれかを使用して、同じローセットを指定できます。

このような場合に、効率的な方法とそうでない方法があります。オプティマイザは、入力された各文を常に自動的に分析し、効率的な実行方法を選択します。意図する結果を最も明確に記述する構文を使用するのが最善です。通常は、前にある句の不要なローが削除されます。

例

受注数が 55 を超え、かつ ID が 1000 よりも大きいすべての営業担当者をリストするには、次のクエリを入力します。

```
SELECT SalesRepresentative, COUNT( * )
FROM SalesOrders
WHERE SalesRepresentative > 1000
GROUP BY SalesRepresentative
HAVING count( * ) > 55
ORDER BY SalesRepresentative;
```

次のクエリも同じ結果になります。

```
SELECT SalesRepresentative, COUNT( * )
FROM SalesOrders
GROUP BY SalesRepresentative
HAVING count( * ) > 55 AND SalesRepresentative > 1000
ORDER BY SalesRepresentative;
```

データベースサーバは、両方の文で同じ結果セットが記述されていることを検出するため、それぞれの文が効率的に実行されます。

1.3.1.10 高度: クエリ処理のフェーズ

注釈フェーズから実行完了まで、文にはいくつかのフェーズがあります

結果セットのない文 (UPDATE 文や DELETE 文など) も、クエリ処理のフェーズを経由します。

注釈フェーズ

データベースサーバは、クエリを受け取ると、パーサを使用して文を解析し、クエリの代数表現 (解析ツリー) に変換します。このフェーズでは、解析ツリーはセマンティックと構文のチェック (クエリで参照されるオブジェクトがカタログ内に存在することの検証など)、権限のチェック、定義済み参照整合性制約を使用した KEY JOIN と NATURAL JOIN 変換、非マテリアライズドビュー展開などに使用されます。このフェーズの出力は、解析ツリーの形式で書き換えられたクエリで、元のクエリで参照されるすべてのオブジェクトに対する注釈が含まれます。

セマンティック変形フェーズ

このフェーズでは、クエリに対して反復的なセマンティック変形を実行します。クエリが注釈付きの解析ツリーとして表されることに変わりはありませんが、リライト最適化 (ジョインの削除、DISTINCT の削除、述部の正規化など) がこのフェーズで適用されます。このフェーズのセマンティック変形は、解析ツリー表現にヒューリスティックに適用されるセマンティック変形規則に従って実行されます。

データベースサーバによってプランがキャッシュ済みのクエリは、このクエリ処理のフェーズをスキップします。また、単純な文もこのフェーズをスキップする場合があります。たとえば、オプティマイザのバイパスでヒューリスティックプラン選択を

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

使用する文の多くは、セマンティック変形フェーズで処理されません。このフェーズが文に適用されるかどうかは SQL 文の複雑さによって決まります。

最適化フェーズ

最適化フェーズでは、クエリの別の内部表現であるクエリ最適化構造体を使用します。クエリ最適化構造体は、解析ツリーから構築されます。

データベースサーバによってプランがキャッシュ済みのクエリでは、このクエリ処理フェーズはスキップされます。また、単純な文もこのフェーズをスキップする場合があります。

このフェーズは、次の 2 つのサブフェーズに分かれています。

最適化前フェーズ

最適化前フェーズは、後から列挙フェーズで必要になる情報を最適化構造体に設定します。このフェーズでは、クエリを分析し、クエリアクセスプランで使用できる関連するインデックスとマテリアライズドビューをすべて検出します。たとえばこのフェーズでは、ビューマッチングアルゴリズムにより、クエリのすべてまたは一部を満たすために使用可能なすべてのマテリアライズドビューが特定されます。また、オプティマイザは、クエリの述部分析を基にして、クエリのテーブルをジョインするために列挙フェーズで使用可能な代替のジョイン方式を構築します。このフェーズでは、最適なクエリアクセスプランに関する決定は行われません。このフェーズの目的は、列挙フェーズの準備です。

列挙フェーズ

このフェーズでオプティマイザは、最適化前フェーズで生成した構成要素を使用して、クエリの可能なアクセスプランを列挙します。検索領域は非常に大きいため、オプティマイザは生成に独自の列挙アルゴリズムを使用し、生成されたアクセスプランを削除します。プランごとにコスト推定が計算されます。コスト推定は、それまでの最適なプランと現在のプランとを比較するために使用されます。この比較時に、コストの高いプランは廃棄されます。コスト推定では、リソースの使用 (ディスクや CPU の操作など)、中間結果のロー数の予測値、最適化目標、キャッシュサイズなどが考慮されます。列挙フェーズの出力は、クエリの最適なアクセスプランです。

プラン構築フェーズ

プラン構築フェーズでは、最適なアクセスプランを利用して、クエリの実行に使用するクエリ実行プランの対応する最終表現を構築します。Interactive SQL のプランビューアで、プランのグラフィカルバージョンを表示できます。グラフィカルなプランにはツリー構造があり、各ノードは特定の関係代数演算を実装する物理演算子です。たとえば [ハッシュジョイン] や [順序付けされた Group By] は、それぞれジョイン操作や GROUP BY 操作を実装する物理演算子です。

データベースサーバによってプランがキャッシュ済みのクエリでは、このクエリ処理フェーズはスキップされます。

実行フェーズ

クエリの結果は、プラン構築フェーズで構築されたクエリ実行プランを使用して計算されます。

このセクションの内容:

クエリ処理のフェーズをスキップするための条件 [201 ページ]
ほぼすべての文が、すべてのクエリ処理のフェーズを経由します。

関連情報

クエリ処理中に実行される最適化 [210 ページ]
プランのキャッシュ [207 ページ]
グラフィカルプラン [216 ページ]
オプティマイザの仕組み [203 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Query Processing Based on SQL Anywhere 12.0.1 Architecture

1.3.1.10.1 クエリ処理のフェーズをスキップするための条件

ほぼすべての文が、すべてのクエリ処理のフェーズを経由します。

ただし、例外が主に 2 つあります。プランのキャッシュの恩恵を受けるクエリ (データベースサーバによってプランがキャッシュ済のクエリ) とバイパスクエリです。

プランのキャッシュ

ストアドプロシージャまたはユーザ定義関数内のクエリの場合、データベースサーバは再利用できるように実行プランをキャッシュします。このクラスのクエリの場合、クエリ実行プランは実行後にキャッシュされます。次回クエリが実行されると、プランが取得され、実行フェーズまでのすべてのフェーズがスキップされます。

バイパスクエリ

バイパスクエリは、データベースサーバがオプティマイザをバイパスできると認識する特定の特性を持った、単純なクエリのサブクラスです。最適化をバイパスすることで、実行プランの構築時間を削減できます。

クエリがバイパスクエリとして認識されると、コストベースの最適化ではなくヒューリスティックが使用されます。つまり、セマンティック変形と最適化のフェーズはスキップされ、クエリの実行プランはクエリの解析ツリー表現から直接構築されます。

単純なクエリ

単純なクエリとは、次の特性を持った単一クエリブロックの SELECT 文、INSERT 文、DELETE 文、UPDATE 文です。

- クエリブロックに、サブクエリ、UNION、INTERSECT、EXCEPT などの追加クエリブロック、および共通テーブル式が含まれていません。
- クエリブロックが 1 つのベーステーブルまたはマテリアライズドビューを参照します。
- クエリブロックには、TOP N 句、FIRST 句、ORDER BY 句、DISTINCT 句を含めることができます。
- クエリブロックには、GROUP BY 句や HAVING 句を含まない集合関数を含めることができます。
- クエリブロックに、Window 関数が含まれません。
- クエリブロック式に、NUMBER、IDENTITY、サブクエリが含まれません。
- ベーステーブルに定義された制約が単純な式です。

セマンティック変形フェーズ後、複雑な文は単純な文に変形される場合があります。単純な文に変形されると、クエリをオプティマイザバイパスで処理したり、SQL Anywhere サーバでクエリプランをキャッシュしたりすることができます。

最適化の実行と最適化の非実行の強制

プランのキャッシュまたはオプティマイザのバイパスの条件を満たすクエリは、SQL Anywhere オプティマイザでの処理を強制できます。これには、任意の SQL 文で FORCE OPTIMIZATION 句を使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

また、文がオプティマイザをバイパスするように強制することもできます。これには、文の FORCE NO OPTIMIZATION 句を使用します。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性などが原因で、文が複雑すぎてオプティマイザをバイパスできない場合、クエリは失敗し、エラーが返されます。

FORCE OPTIMIZATION 句と FORCE NO OPTIMIZATION 句は、次の文の OPTION 句で使用できます。

- SELECT 文
- UPDATE 文
- INSERT 文
- DELETE 文

関連情報

プランのキャッシュ [207 ページ]

1.3.1.11 高度: クエリ最適化

最適化は、クエリの適切なアクセスプランを生成するのに重要な処理です。

クエリが解析されると、クエリオプティマイザ (または簡略して、オプティマイザ) はクエリを分析し、できるだけ少ないリソースを使用して結果を計算するアクセスプランを決定します。最適化が実行直前に開始されます。アプリケーションでカーソルを使用している場合は、カーソルを開いたときに最適化が開始されます。

他の多くの商用データベースシステムとは異なり、SQL Anywhere では通常、各文を実行する直前に最適化を行います。データベースサーバは各文の最適化をその都度実行するため、オプティマイザはホスト変数とストアドプロシージャ変数の値にアクセスできます。これにより、より良い選択性推定分析を実行できます。また、最適化をその都度実行するため、オプティマイザは前のクエリ実行後に保存された統計を基に、選択を調整できます。

データベースサーバは操作を効率化するために、セマンティック上は同等で、構文上は異なる形式にユーザのクエリを書き換えます。データベースサーバは、多数の異なる書き換えオペレーションを実行します。アクセスプランを読めば、それが元の文のリテラルな解釈と一致していないことがよくあります。たとえば、SQL 文の効率を高めるために、オプティマイザはジョインを使ってサブクエリを可能なかぎり書き換えようとします。

このセクションの内容:

オプティマイザの仕組み [203 ページ]
オプティマイザの役割は、SQL 文を実行する効率的な方法を考案することです。

クエリ処理中に実行される最適化 [210 ページ]
クエリ書き換えフェーズでは、より効率性に優れたクエリの表現方法を検討するために、データベースサーバはセマンティック変形を実行します。

関連情報

Query Optimization Based on SQL Anywhere 12.0.1 Architecture

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.11.1 オプティマイザの仕組み

オプティマイザの役割は、SQL 文を実行する効率的な方法を考案することです。

これを行うには、オプティマイザは、クエリの実行プランを判断する必要があります。これには、クエリで参照されるテーブルのアクセス順序、各テーブルに使用されるジョイン演算子とアクセス方式、クエリの各部分の計算でクエリで参照されないマテリアライズドビューを使用できるかどうかなどの判断が含まれます。オプティマイザは、クエリで可能なアクセスプランを生成してコストを計算するときに、ジョイン列挙フェーズ中にクエリを実行するための最適なアクセスプランを選択します。最適なアクセスプランでは、オプティマイザの推測値が最短時間と最低コストで望ましい結果セットを返します。オプティマイザは、ディスクへの必要な読み書き回数を推定して、列挙された各方式のコストを決定します。

Interactive SQL で、ツール > プランビューア をクリックして、クエリの実行に使用される最適なアクセスプランを表示できます。

最初のローを返すコストの最小化

オプティマイザは、汎用的なディスクアクセスコストモデルを使用して、データベースファイルに対するランダム検索と逐次検索の相対的なパフォーマンスの差異を認識します。ALTER DATABASE 文を使用すると、データベースを特定のハードウェア構成に対応させることができます。

デフォルトでは、クエリ処理はすべての結果セットを返すように最適化されています。optimization_goal オプションを使用してデフォルトの動作を変更すると、最初のローを迅速に返すコストを最小限に抑えることができます。このオプションを First-row に設定すると、オプティマイザは、クエリの結果の最初のローをフェッチするまでの時間を短縮するアクセスプランを選択します。この場合、検索にかかる合計時間は長くなることがあります。

セマンティック上等しい構文の使用

ほとんどの文は、SQL 言語を使用してさまざまな方法で表すことができます。これらの表現は、同じタスクを実行するという点でセマンティック上は同等ですが、構文はまったく異なる場合があります。ごくわずかな例外はありますが、オプティマイザは、各文のセマンティックだけに基づいて適切なアクセスプランを考案します。

構文の違いは重要に見えるかもしれませんが、通常はまったく影響ありません。たとえば、クエリ構文で述部、テーブル、属性の順序が違ってもアクセスプランの選択には影響しません。クエリに非マテリアライズドビューが含まれているかどうかによってオプティマイザが影響を受けることもありません。

クエリを最適化するコストの削減

オプティマイザは実現可能な最も効率の良いアクセスプランを見つけようとしますが、通常、これは現実的ではありません。複雑なクエリの場合、さまざまな可能性が存在します。

オプティマイザは効率的ですが、各オプションの分析には、時間とリソースを必要とします。オプティマイザは、これから行う最適化のコストと、これまでに見つけた最高のプランの実行にかかるコストを比較します。相対的にコストの低いプランが考案されると、オプティマイザは停止し、そのプランの実行を進めます。さらに最適化を行うと、すでに見つかっているアクセスプラン

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

を実行する場合よりも多くのリソースを消費する可能性があります。optimization_level オプションの値を高く設定することで、オプティマイザが費やす作業量を指定できます。

高いコストがかかる複雑なクエリの場合、または、最適化レベルが高く設定されている場合は、オプティマイザの動作時間が長くなります。コストが非常に高いクエリの場合は、クエリの実行時間が長くなり、認識できるほどの遅延が発生することがあります。

このセクションの内容:

オプティマイザの推定と統計 [204 ページ]
オプティマイザは、データベースに格納されているカラム統計とヒューリスティックに基づいて、文の処理方式を選択します。

選択性推定ソース [206 ページ]
任意の述部で、オプティマイザは選択性推定のいくつかのソースを使用できます。選択されたソースは、クエリの長いグラフィカルプランに示されます。

プランのキャッシュ [207 ページ]
オプティマイザでは、クエリ実行時にキャッシュされたプランを使用することができます。

サブクエリと関数のキャッシュ [209 ページ]
データベースサーバは、サブクエリを処理すると、結果をキャッシュします。

関連情報

グラフィカルプラン [216 ページ]

高度: クエリ実行プラン [210 ページ]

1.3.1.11.1.1 オプティマイザの推定と統計

オプティマイザは、データベースに格納されているカラム統計とヒューリスティックに基づいて、文の処理方式を選択します。

オプティマイザが検討するアクセスプランごとに、推定された結果サイズ (ローの数) を計算する必要があります。たとえば、クエリで使用される述部の選択性推定に基づいたジョイン方式やインデックスアクセスごとに、推定された結果サイズが計算されます。推定された結果サイズは、プランで使用される演算子ごと (ジョイン方式、GROUP BY 方式、逐次スキャンなど) にディスクアクセスや CPU の推定コストの計算に使用されます。カラム統計は、述部の選択性推定を計算するためにオプティマイザが使用するプライマリデータです。そのため、アクセスプランのコストを適切に推定するためにカラム統計は重要です。

カラム統計が古くなったりなくなったりすると、不正確な統計により非効率な実行プランとなる可能性があり、パフォーマンスが低下することがあります。パフォーマンスの低下の原因が不正確なカラム統計にあると考えられる場合は、カラム統計を再作成してください。

オプティマイザによる統計の使用方法

オプティマイザが使用するカラム統計の最も重要なコンポーネントは、ヒストグラムです。ヒストグラムは、単一カラムの値の分散に関する情報を格納します。ヒストグラムはカラムのデータ分散を表します。これは、カラムのドメインを連続する値の範囲

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

集合 (バケットとも呼ばれる) に分け、値の範囲 (バケット) それぞれに収まるカラム値のあるテーブルのロー数を記憶することで表されます。

データベースサーバでは、テーブルの多数のローにある単一のカラム値が特に注目されます。重要な単一値の選択性は、シングルトンのヒストグラムバケット (たとえば、カラムドメインの単一の値を包含するバケット) で管理されます。データベースサーバは、各ヒストグラムのシングルトンバケットの数を最小限に抑えようとします。その数は、テーブルのサイズによって決まりますが、通常 10 から 100 の間です。また、選択性が 1% より大きい単一値はすべてシングルトンバケットとして管理されます。その結果、あるカラムのヒストグラムは、そのカラムの単一値の選択性のうち上位 N 個を記憶します。ここで N の値は、テーブルのサイズと 1% より大きい単一値の選択性の数によって決まります。

いったん値の範囲の数が最小数に達すると、頻度の高い選択性が出現するごとに頻度の低い選択性を置き換えます。ヒストグラムは、選択性が 1% より大きい値が十分あると判断した場合のみ、最小数を超える単集合の値の範囲を持ちます。

ベーステーブルとは異なり、FROM 句で実行されるプロシージャコールにはカラム統計がありません。したがって、オプティマイザでは、プロシージャコールからのデータの選択性推定にデフォルトまたは推測が使用されます。プロシージャコールの実行時間と、結果セット内のローの合計数は、以前の呼び出しから収集された統計を使用して推定されます。これらの統計は、ISYSPROCEDURE システムテーブルの stats カラムに格納されます。

オプティマイザによるヒューリスティックの使用方法

採用できそうな実行プラン内の各テーブルについて、オプティマイザは結果の一部となるローの数を推定します。ローの数は、テーブルのサイズとクエリの WHERE 句または ON 句に指定された制限によって異なります。

データベースサーバは、カラムに対する特定のクエリ述部を満たすローの数を、そのカラムに対するヒストグラムによって推定します。そのためには、指定の述部を満たす値を含むすべての範囲にあるローの数を合算します。クエリの結果セットに部分的に含まれるヒストグラムの値の範囲には、内挿法が使用されます。

多くの場合、オプティマイザはより高度なヒューリスティックを使用します。たとえば、オプティマイザは適切な統計がない場合に限ってデフォルトの推定値を使用します。また、オプティマイザは、インデックスとキーを使用してロー数の推定精度を上げます。単一のカラムで推定する例を次に示します。

- カラム内である値を持つロー数: そのカラムがユニークインデックスを持つかプライマリキーである場合、ロー数は 1 つだと推定します。
- インデックス付きカラムで定数と比較したときのロー数: インデックスを調査し、比較条件を満たすローのパーセンテージを推定します。
- 外部キーからプライマリキーへのロー数 (キージョイン): テーブルの相対的サイズを使って推定します。たとえば、5000 ローのテーブルが 1000 ローのテーブルに対して外部キーを持つとき、オプティマイザは 1 つのプライマリキーローに対して 5 個の外部キーローがあると推定します。

関連情報

選択性推定ソース [206 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.11.1.2 選択性推定ソース

任意の述部で、オプティマイザは選択性推定のいくつかのソースを使用できます。選択されたソースは、クエリの長いグラフィカルプランに示されます。

以下は、選択性推定のソースとして考えられるものです。

統計情報

オプティマイザは、格納されたカラム統計を使用して選択性推定を計算できます。述部で定数が使用されている場合、格納された統計情報を使用できるのは、定数の選択性が統計情報に格納されるだけ重大な数値であるときだけです。

たとえば、述部 `EmployeeID > 100` では、EmployeeID カラムの統計情報が存在する場合、カラム統計を選択性推定ソースとして使用できます。

Join

オプティマイザは、参照整合性制約、一意性制約、またはジョインヒストグラムを使用して選択性推定を計算できます。ジョインヒストグラムは、T.X カラムおよび R.X カラムの使用可能な統計情報から `T.X=R.X` の形式の述部に対して計算されます。

Column-column

選択性ソースとして使用できる参照整合性制約、一意性制約、またはジョインヒストグラムがないジョインの場合、オプティマイザは選択性ソースとして、ジョインの結果セットの推定ロー数を 2 つのテーブルの直積のロー数で割った値を使用できます。

Column

オプティマイザは、カラム統計に格納されたすべての値の平均値を使用できます。

たとえば、述部 `DepartmentName = expression` の選択性は、`expression` が定数でない場合、平均値を使用して計算できます。

Index

オプティマイザは、インデックスを調査して選択性推定を計算できます。通常、インデックスは、カラム統計などの選択性推定の他のソースが使用できない場合に選択性推定に使用されます。

たとえば、述部 `DepartmentName = 'Sales'` では、オプティマイザはカラム DepartmentName で定義されたインデックスを使用して Sales の値を持つローの数を推定できます。

User

user_estimates データベースオプションが Disabled に設定されていない場合、オプティマイザはユーザが提供する選択性推定を使用できます。

Guess

使用できる適切なインデックスがない場合、参照先カラムの統計情報が収集されていない場合、または述部が複雑である場合、オプティマイザは最も妥当な推測を行って選択性推定を計算できます。この場合は、各タイプの述部に組み込み規則が定義されます。

Computed

たとえば、選択性を乗算するか、加算して選択性推定が計算された場合、非常に複雑な述部では、たとえば、選択性推定を 100% に設定し、選択性ソースを Computed に設定できます。

Always

述部が常に true の場合、選択性ソースは 'Always' になります。たとえば、述部 `1=1` は常に true です。

Combined

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

上記の複数のソースを組み合わせて選択性推定が計算される場合、選択性ソースは 'Combined' になります。

Bounded

データベースサーバが選択性推定に上限か下限またはその両方を設定した場合、選択性ソースは 'Bounded' になります。たとえば、限界は、推定値が 100% を超えたり、選択性が 0% より小さくなったりしないことを保証するために設定されます。

関連情報

グラフィカルプランの選択性情報 [221 ページ]

1.3.1.11.1.3 プランのキャッシュ

オプティマイザでは、クエリ実行時にキャッシュされたプランを使用することができます。

プランキャッシュとは、アクセスプランの実行に使用されるデータ構造の接続ごとのキャッシュのことです。効率的と判断される場合にプランを再利用することが目的です。キャッシュされたプランの再利用には、キャッシュ内のプランを調べることも含まれますが、通常、これはすべてのクエリ処理フェーズを経由させて文を再処理するよりも、はるかに高速です。

クエリ実行時に最適化すると、オプティマイザは、現在のシステム状態、現在の選択性推定値、ホスト変数の値に基づく推定値に従ってプランを選択できます。頻繁に実行されるクエリの場合は、実行時に最適化することによる利点よりもクエリ最適化のコストの方が重要である場合があります。データベースサーバは、これらの文を繰り返し最適化するコストを削減するために、後述の文に関する実行プランのキャッシュを検討します。

クライアント文の場合、キャッシュされた実行プランの存続期間は、対応する文の存続期間に制限されます。また、クライアント文が削除されると、対応する文がプランキャッシュから削除されます。クライアント文 (および対応するすべての実行プラン) の存続期間は、準備されたクライアント文の個別のキャッシュによって延長することができ、max_client_statements_cached オプションで制御します。システムの設定によって、クライアント文はパラメータ化されてキャッシュされ、対応する実行プランが再利用される機会が増えることがあります。

キャッシュできるプランの最大数は、max_plans_cached オプションで指定します。

sp_plancache_contents システムプロシージャを使用して、プランキャッシュの現在の内容を調べます。

QueryCachedPlans 統計を使用して、現在キャッシュされているクエリ実行プラン数を表示できます。このプロパティを検索するために CONNECTION_PROPERTY 関数を使用すると、特定の接続についてキャッシュされているクエリ実行プラン数を表示できます。DB_PROPERTY 関数を使用すると、接続全体でキャッシュされている実行プランを数えることができます。このプロパティを QueryCachePages、QueryOptimized、QueryBypassed、QueryReused と組み合わせて使用すると、max_plans_cached オプションの最適な設定を決定するのに役立ちます。

データベースプロパティまたは接続プロパティである QueryCachePages を使用すると、実行プランをキャッシュするために使用するページ数を決定できます。これらのページはテンポラリファイル内の領域を占めますが、メモリに常駐するとはかぎりません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

プランがキャッシュされるタイミング

データベースサーバは、キャッシュするプランとしないプランを判断します。プランキャッシュのポリシーでは、文やそのプランの評価時に採用する基準やアクションを定義します。このポリシーはバックグラウンドで作用し、プランキャッシュの動作を管理します。たとえば、ポリシーによって、文に対して行われる実行数 (トレーニング期間) や、得られるプラン内で検索する結果数を決定して、キャッシュと再利用のプランの条件を定めます。

接続によって、条件に合う文が複数回実行されると、データベースサーバは再利用プランの構築を決定します。再利用可能なプランの構造が、文が以前に実行されたときのプランと同じである場合、データベースサーバは、そのプランキャッシュに再利用可能なプランを追加します。最適化を避けることによるコストの削減よりも、実行のたびに最適化しないことによるリスクが勝る場合、実行プランはキャッシュされません。

クエリ実行プランは、実行時間の長いクエリについてはキャッシュされません。これは、クエリの最適化を避けることのメリットが、クエリの総コストに比べて小さいためです。さらに、データベースサーバでは、ホスト変数の値に大きく左右されるクエリのプランはキャッシュしない場合があります。

文で参照されないマテリアライズドビューを実行プランで使用し、materialized_view_optimization オプションが Stale 以外に設定されている場合、実行プランはキャッシュされず、次回実行時に文は再度最適化されます。

キャッシュの使用量を最小化するため、キャッシュされたプランの使用頻度が低い場合は、ディスクに格納されることがあります。また、オプティマイザはクエリを定期的に最適化し直して、キャッシュされたプランが依然として効率的であるかどうかを確認します。

プランキャッシュと文のパラメータ化

データベースサーバでは、条件を満たすクライアント文をパラメータ化し、プランキャッシュの機会を広げることができます。パラメータ化された文には、実行時に評価される変数のように動作するプレースホルダを使用します。パラメータ化がもたらすパフォーマンスのオーバヘッドは、一文の文に対してはごく少量ですが、パラメータ化された文のテキストはより汎用的なため、より多くの SQL 文に一致します。結果として、パラメータ化された文に一致するすべての SQL クエリでキャッシュされた同じプランを共有できるため、文のパラメータ化によってプランキャッシュの効率性を改善することができます。

文のパラメータ化は、parameterization_level オプションで制御します。このオプションによって、データベースサーバがパラメータ化のタイミングを決定 (シンプル) したり、できるだけ早急にすべての文をパラメータ化 (強制) したり、またはいずれの文もパラメータ化しない (オフ) という設定をすることができます。デフォルトでは、データベースサーバが文のパラメータ化のタイミングを決定できるように設定されています (シンプル)。

parameterization_level 接続プロパティを問い合わせて、有効なパラメータ化動作の情報を取得します。パラメータ化が有効な場合、ParameterizationPrepareCount 接続プロパティを問い合わせて、現在の接続がデータベースサーバに発行したパラメータ化された文の準備要求を取得します。

関連情報

クエリ処理のフェーズをスキップするための条件 [201 ページ]
マテリアライズドビュー [64 ページ]
高度: クエリ処理のフェーズ [199 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.11.1.4 サブクエリと関数のキャッシュ

データベースサーバは、サブクエリを処理すると、結果をキャッシュします。

このキャッシュは要求ごとに行われるので、キャッシュされた結果が同時に実行された要求や接続の間で共有されることはありません。同じ相関値のセットについてサブクエリの再評価が必要な場合、データベースサーバではキャッシュから結果を取り出すだけで済みます。このようにして、データベースサーバは、何度も繰り返される冗長な計算を避けます。要求が完了すると (クエリのカーソルが閉じられると)、データベースサーバはキャッシュされた値を解放します。

クエリの処理が進むに従って、データベースサーバは、キャッシュされたサブクエリの値が再使用された頻度をモニタします。相関変数の値がめったに繰り返されない場合、データベースサーバは、ほとんどの値を 1 回しか計算する必要がありません。このような場合、データベースサーバは、一度しか発生しない数多くのエントリをキャッシュするよりも、時折重複する値を再計算する方が効率的であると判断します。そのため、データベースサーバは文の残りの部分についてこのサブクエリのキャッシュを中断し、外部クエリブロック内のすべてのローに関するサブクエリの再評価を開始します。

従属カラムのサイズが 255 バイトを超える場合も、データベースサーバはキャッシュを行いません。その場合、クエリを書き換えるか、または別のカラムをテーブルに追加して、操作をより効率的にすることを検討します。

このセクションの内容:

関数のキャッシュ [209 ページ]
一部の組み込み関数とユーザ定義関数は、サブクエリの結果と同じ方法でキャッシュされます。

1.3.1.11.1.4.1 関数のキャッシュ

一部の組み込み関数とユーザ定義関数は、サブクエリの結果と同じ方法でキャッシュされます。

このため、同じパラメータを使用したクエリの処理中に呼び出される高コストの関数のパフォーマンスが大幅に向上します。ただし、これは関数の呼び出し回数が予想より少なくなることを意味しています。

関数がキャッシュされるためには、2 つの条件を満たす必要があります。

- 特定のパラメータセットに対して常に同じ結果を戻す
- 基本となるデータに対し副次的な影響を与えない

これらの条件を満たす関数は、決定性関数、またはべき等関数と呼ばれます。データベースサーバでは、すべてのユーザ定義関数は、作成時に NOT DETERMINISTIC と宣言されない限り、決定性として扱われます。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は、どちらの呼び出しでも同じ結果が返され、クエリの意味に不要な弊害は生じないものとみなします。

組み込み関数は、決定性関数として扱われますが、いくつか例外があります。RAND、NEWID、GET_IDENTITY 関数は非決定性関数として扱われ、その結果はキャッシュされません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.11.2 クエリ処理中に実行される最適化

クエリ書き換えフェーズでは、より効率性に優れたクエリの表現方法を検討するために、データベースサーバはセマンティック変形を実行します。

クエリはセマンティック上等しいクエリに書き換えられる場合があるため、このプランは、元のクエリとはかなり異なる場合があります。一般的な操作は次のとおりです。

- 不要な DISTINCT 条件の排除
- サブクエリのネスト解除
- UNION または GROUP 化されたビューや派生テーブルでの述部のプッシュダウンの実行
- OR 述部と IN リスト述部の最適化
- LIKE 述部の最適化
- 外部ジョインの内部ジョインへの変換
- 外部ジョインと内部ジョインの削除
- 述部の推定による利用可能な条件の発見
- 大文字と小文字の不要な変換の排除
- サブクエリを EXISTS 述部として書き換え
- 検索引数可能な IN 述部の推定 (AND 述部に変換できない OR 述部からの部分インデックススキャンに使用可能)

注記

カーソルが更新可能な場合、メインクエリブロックではクエリ書き換え最適化を実行できないことがあります。カーソルを読み込み専用として宣言すると、最適化を利用できます。

クエリ書き換えフェーズで実行される書き換えの最適化のいくつかは、REWRITE 関数で返される結果で観察できます。

1.3.1.12 高度: クエリ実行プラン

実行プランは、データベースサーバがデータベース内の文に関連する情報にアクセスするために使用する一連のステップです。

最適化されたばかりかどうか、オプティマイザをバイパスしたかどうか、プランが以前の実行からキャッシュされたかどうかなどに関係なく、文の実行プランの保存と確認が可能です。クエリの実行プランは、元の文で使用される構文と正確に対応するとはかぎりません。クエリで明示的に指定したベーステーブルの代わりにマテリアライズドビューを使用する場合があります。ただし、実行プランに記述された操作は、セマンティック上は元のクエリと同等です。

Interactive SQL 内で、または SQL 関数を使用することによって、実行プランを表示できます。実行プランを取り出すときに、次のようなフォーマットを選択できます。

- 短いテキストプラン
- 長いテキストプラン
- グラフィカルなプラン
- グラフィカルなプラン (ルート統計あり)
- グラフィカルなプラン (全統計あり)
- Ultra Light (短い、長い、またはグラフィカル)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

クエリ実行プランのテキスト表示には、短いプランと長いプランの 2 種類があります。SQL 関数を使用して、テキストプランにアクセスできます。プランには、グラフィカルなバージョンもあります。GRAPHICAL_PLAN と EXPLANATION の各関数を使用し、特定のカーソルタイプに基づいて SQL クエリのプランを取得することもできます。

このセクションの内容:

短いテキストプラン [211 ページ]
　短いテキストプランは、プランを短時間で比較する場合に便利です。

長いテキストプラン [212 ページ]
　長いテキストプランでは、短いテキストプランより多くの情報が提供されるので、スクロールしなくても簡単に印刷したり表示したりすることができます。

短いテキストプランの表示 [214 ページ]
　SQL を使用して、短いテキストプランを表示します。

長いテキストプランの表示 (SQL の場合) [215 ページ]
　SQL を使用して、長いテキストプランを表示します。

グラフィカルプラン [216 ページ]
　Interactive SQL および プロファイラ のグラフィカルプラン機能を使用すると、クエリの実行プランを表示できます。

チュートリアル: Interactive SQL におけるプランの比較 [224 ページ]
　Interactive SQL の**プラン比較**ウィンドウを使用して、クエリの実行プランを比較できます。

実行プランのコンポーネント [230 ページ]
　実行プランには多数の省略形があります。

関連情報

高度: クエリ処理のフェーズ [199 ページ]

高度: クエリ最適化 [202 ページ]

Query Processing Based on SQL Anywhere 12.0.1 Architecture

グラフィカルプランの表示 [223 ページ]

1.3.1.12.1 短いテキストプラン

短いテキストプランは、プランを短時間で比較する場合に便利です。

短いプランでは、すべてのプランフォーマットの最低限の情報が 1 行で表示されます。

次の例では、ORDER BY 句によって結果セット全体がソートされるため、プランは `Work[Sort` で始まります。Customers テーブルは、プライマリキーインデックス CustomersKey によってアクセスされます。カラム Customers.ID がプライマリキーのため、インデックススキャンを使用して探索条件が満たされます。省略形 JNL は、Customers と SalesOrders の間のジョインを処理するためにオプティマイザでマージジョインが選択されたことを示します。最後に、外部キーインデックス

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

FK_CustomerID_ID を使用して SalesOrders テーブルがアクセスされ、Customers テーブル内で CustomerID が 100 未満であるローが検索されます。

```
SELECT EXPLANATION ('SELECT GivenName, Surname, OrderDate
FROM Customers JOIN SalesOrders
WHERE CustomerID < 100
ORDER BY OrderDate');
```

```
Work[ Sort[ Customers<CustomersKey> JNL SalesOrders<FK_CustomerID_ID> ] ]
```

ジョイン方式の区切りにコロンを使用

次に示す文には、クエリブロックが 2 つ含まれています。1 つは SalesOrders テーブルと SalesOrderItems テーブルを参照する外部 SELECT ブロック、もう 1 つは Products テーブルから選択するサブクエリです。

```
SELECT EXPLANATION ('SELECT *
FROM SalesOrders AS o
   KEY JOIN SalesOrderItems AS I
WHERE EXISTS
    (  SELECT *
       FROM Products p
       WHERE p.ID = 300 )');
```

```
o<seq> JNL i<FK_ID_ID> : p<ProductsKey>
```

各クエリブロックのジョイン方式はコロンで区切られます。短いプランでは常に、メインブロックのジョイン方式が先にリストされます。その後に、他のクエリブロックのジョイン方式がリストされます。このような他のクエリブロックのジョイン方式の順序は、文におけるクエリブロックの順序やその実行順序とは一致しない場合があります。

関連情報

実行プランのコンポーネント [230 ページ]
チュートリアル: Interactive SQL におけるプランの比較 [224 ページ]

1.3.1.12.2 長いテキストプラン

長いテキストプランでは、短いテキストプランより多くの情報が提供されるので、スクロールしなくても簡単に印刷したり表示したりすることができます。

長いプランには、文のキャッシュされたプランなどの情報が含まれています。

例

例 1

この例では、長いテキストプランは次の文に基づいています。

```
SELECT PLAN ('SELECT GivenName, Surname, OrderDate, Region, Country
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM Customers JOIN SalesOrders ON ( SalesOrders.CustomerID = Customers.ID )
WHERE CustomerID < 120 AND ( Region LIKE ''Eastern''
      OR Country LIKE ''Canada'' )
ORDER BY OrderDate', 'keyset-driven', 'read-only');
```

長いテキストプランは次のように表示されます。

```
( Plan [ Total Cost Estimate: 6.46e-005, Costed Best Plans: 1, Costed Plans:
10, Optimization Time: 0.0011462,
Estimated Cache Pages: 348 ]
  ( WorkTable
    ( Sort
      ( NestedLoopsJoin
        ( IndexScan Customers CustomersKey[ Customers.ID < 100 : 0.0001% Index
| Bounded ] )
        ( IndexScan SalesOrders FK_CustomerID_ID[ Customers.ID =
SalesOrders.CustomerID : 0.79365% Statistics ]
          [  ( SalesOrders.CustomerID < 100 : 0.0001% Index | Bounded )
          AND  ( ( ((Customers.Country LIKE 'Canada' : 100% Computed)
          AND (Customers.Country = 'Canada' : 5% Guess))
          OR ((SalesOrders.Region LIKE 'Eastern' : 100% Computed)
          AND (SalesOrders.Region = 'Eastern' : 5% Guess)) ) : 100%
Guess )  ] )
      )
    )
  )
)
```

Plan という語はクエリブロックの開始を示します。Total Cost Estimate は、オプティマイザでプランの実行に要すると推測された時間 (ミリ秒単位) です。Costed Best Plans、Costed Plans、Optimization Time は最適化処理の統計であり、Estimated Cache Pages は文の処理に使用できる現在の推定キャッシュサイズです。

このプランは、結果がソートされ、ネストループジョインが使用されることを示します。ジョイン演算子と同じ行に、ジョイン条件とその選択性推定 (ジョイン演算子によって作成されるすべてのローについて推定) があります。IndexScan の行は、Customers と SalesOrders の各テーブルが、それぞれ CustomersKey と FK_CustomerID_ID の各インデックスを使用してアクセスされることを示します。

例 2

次の文がプロシージャ、トリガ、または関数内で使用され、文のプランがキャッシュされて 5 回再利用された場合、長いテキストプランには、文が再利用可能であり、キャッシュ後に 5 回使用されたことを示す文字列 `[R: 5]` が含まれます。文で使用されるパラメータ parm1 は、このプランでは不定の値を持ちます。

```
UPDATE Account SET Account.A = 10 WHERE Account.B =parm1
```

```
( Update [ Total Cost Estimate: 1e-006, Costed Best Plans: 1, Costed Plans: 2,
Carver pages: 0,
Estimated Cache Pages: 46768 ] [ R: 5 ]
  ( Keyset
    ( TableScan ( Account ) ) [ Account.B = parm1 : 0.39216% Column ]
  )
 )
)
```

同じ文のプランがまだキャッシュされていない場合、長いテキストプランにはパラメータ parm1 の値 (10 など) が含まれており、プランはこのパラメータの値を使用して最適化されたことを示しています。

```
( Update [ Total Cost Estimate: 1e-006, Costed Best Plans: 1, Costed Plans: 2,
Carver pages: 0,
Estimated Cache Pages: 46768 ]
  ( Keyset
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
      ( TableScan ( Account ) ) [ Account.B = parm1 [ 10 ] : 0.001% Statistics ]
    )
   )
  )
```

関連情報

実行プランのコンポーネント [230 ページ]

1.3.1.12.3 短いテキストプランの表示

SQL を使用して、短いテキストプランを表示します。

前提条件

その関数が実行されるオブジェクトの所有者であるか、または、そのオブジェクトに対する SELECT、UPDATE、DELETE、INSERT のうち、適切な権限を持っている必要があります。

手順

1. データベースに接続します。
2. EXPLANATION 関数を実行します。

結果

短いテキストプランは、Interactive SQL の *[結果]* ウィンドウ枠に表示されます。

例

この例では、短いテキストプランは次の文に基づいています。

```
SELECT EXPLANATION ('SELECT GivenName, Surname, OrderDate
FROM GROUPO.Customers JOIN GROUPO.SalesOrders
WHERE CustomerID < 100
ORDER BY OrderDate');
```

短いテキストプランは次のように表示されます。

```
Work[ Sort[ Customers<CustomersKey> JNL SalesOrders<FK_CustomerID_ID> ] ]
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ORDER BY 句によって結果セット全体がソートされるため、短いテキストプランは `Work[Sort` で始まります。Customers テーブルは、プライマリキーインデックス CustomersKey によってアクセスされます。カラム Customers.ID がプライマリキーのため、インデックススキャンを使用して探索条件が満たされます。省略形 JNL は、Customers と SalesOrders の間のジョインを処理するためにオプティマイザでマージジョインが選択されたことを示します。最後に、外部キーインデックス FK_CustomerID_ID を使用して SalesOrders テーブルがアクセスされ、Customers テーブル内で CustomerID が 100 未満であるローが検索されます。

関連情報

実行プランのコンポーネント [230 ページ]
短いテキストプラン [211 ページ]
長いテキストプランの表示 (SQL の場合) [215 ページ]

1.3.1.12.4 長いテキストプランの表示 (SQL の場合)

SQL を使用して、長いテキストプランを表示します。

前提条件

その関数が実行されるオブジェクトの所有者であるか、または、そのオブジェクトに対する SELECT、UPDATE、DELETE、INSERT のうち、適切な権限を持っている必要があります。

手順

1. データベースに接続します。
2. PLAN 関数を実行します。

結果

長いテキストプランは、Interactive SQL の**結果**ウィンドウ枠に表示されます。

例

この例では、長いテキストプランは次の文に基づいています。

```
SELECT PLAN ('SELECT GivenName, Surname, OrderDate, Region, Country
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM GROUPO.Customers JOIN GROUPO.SalesOrders ON ( SalesOrders.CustomerID =
Customers.ID )
WHERE CustomerID < 100 AND ( Region LIKE ''Eastern''
      OR Country LIKE ''Canada'' )
ORDER BY OrderDate');
```

長いテキストプランは次のように表示されます。

```
( Plan [ Total Cost Estimate: 6.46e-005, Costed Best Plans: 1, Costed Plans: 10,
Optimization Time: 0.0011462,
Estimated Cache Pages: 348 ]
   ( WorkTable
     ( Sort
       ( NestedLoopsJoin
         ( IndexScan Customers CustomersKey[ Customers.ID < 100 : 0.0001% Index |
Bounded ] )
         ( IndexScan SalesOrders FK_CustomerID_ID[ Customers.ID =
SalesOrders.CustomerID : 0.79365% Statistics ]
           [  ( SalesOrders.CustomerID < 100 : 0.0001% Index | Bounded )
           AND  ( ( ((Customers.Country LIKE 'Canada' : 100% Computed)
           AND (Customers.Country = 'Canada' : 5% Guess))
           OR ((SalesOrders.Region LIKE 'Eastern' : 100% Computed)
           AND (SalesOrders.Region = 'Eastern' : 5% Guess)) ) : 100% Guess )  ] )
       )
     )
   )
)
```

Plan という語はクエリブロックの開始を示します。Total Cost Estimate は、オプティマイザでプランの実行に要すると推測された時間 (ミリ秒単位) です。Costed Best Plans、Costed Plans、Optimization Time は最適化処理の統計であり、Estimated Cache Pages は文の処理に使用できる現在の推定キャッシュサイズです。

このプランは、結果がソートされ、ネストループジョインが使用されることを示します。ジョイン演算子と同じ行に、ジョイン条件とその選択性推定 (ジョイン演算子によって作成されるすべてのローについて推定) があります。IndexScan の行は、Customers と SalesOrders の各テーブルが、それぞれ CustomersKey と FK_CustomerID_ID の各インデックスを使用してアクセスされることを示します。

関連情報

実行プランのコンポーネント [230 ページ]
長いテキストプラン [212 ページ]
短いテキストプランの表示 [214 ページ]

1.3.1.12.5 グラフィカルプラン

Interactive SQL および プロファイラ のグラフィカルプラン機能を使用すると、クエリの実行プランを表示できます。

実行プランは、関係代数演算子のツリーで構成されます。ツリーのルートで最終的な結果が得られるように、ツリーの最下位から開始して、クエリへの基本入力 (通常はテーブルのロー) を消費し、下から上にローを処理します。このツリーのノードは特定の代数演算子に対応しています。ただし、サーバで実行されるすべてのクエリ検査がノードで表示されるとはかぎりません。たとえば、サブクエリと関数のキャッシュの影響は、グラフィカルなプランには直接表示されません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

グラフィカルなプランでは、次のようにノードをさまざまな形で表示して、実行される操作の種類を示します。

- データを実体化する操作は六角形。
- インデックススキャンは台形。
- テーブルスキャンは長方形。
- その他の操作は角丸四角形。

グラフィカルなプランを使用して、特定のクエリのパフォーマンス上の問題を診断できます。たとえば、プラン内の情報に基づいて、この特定のクエリのパフォーマンスを向上させるためにテーブルにインデックスが必要かを判断できます。

Interactive SQL では、プランビューアウィンドウで名前を付けて保存ボタンをクリックすると、クエリのグラフィカルプランを保存して後で参照できます。プロファイラ では、実行文のプロパティウィンドウのプランタブに移動し、グラフィカルプラン、プランの取得、名前を付けて保存をクリックすると、実行文のグラフィカルプランを取得して保存できます。コストの高い文のグラフィカルプランを保存する場合は、コストの高い文のプロパティウィンドウのプランタブに移動し、名前を付けて保存をクリックします。SQL Anywhere のグラフィカルプランは拡張子 `.saplan` で保存されます。

パフォーマンスに問題がある可能性がある場合は、グラフィカルプラン内で太い線と赤い枠線で示されます。例:

- 処理対象のロー数が増加するに従って、プラン内のノード間の線が太くなります。テーブルスキャンに太い線が表示されている場合は、インデックスの作成が必要である可能性を示していることがあります。
- ノードの枠線が赤い場合は、実行プラン内の他の演算子に比べて、その演算子のコストが高かったことを示しています。

ノードの形など、プランのグラフィカルなコンポーネントは、Interactive SQL および プロファイラ 内でカスタマイズできます。

グラフィカルなプラン、概要付きのグラフィカルなプラン、または詳細な統計情報付きのグラフィカルなプランのいずれかを表示できます。3 つのプランはすべて、プランの中で最も高コストとして推定された部分を表示できます。統計情報付きのグラフィカルなプランの生成は、クエリの実行時にデータベースサーバがモニタしている実際のクエリ実行統計が表示されるため高コストです。統計情報付きのグラフィカルなプランでは、クエリオプティマイザがアクセスプランの作成時に使用する推定を、実行中にモニタされた実際の統計と直接比較できます。ただし、オプティマイザはクエリのコストを正確に推定できないことが多いので、推定値と実際の統計値には違いがあると想定してください。

このセクションの内容:

統計情報付きのグラフィカルプラン [218 ページ]
グラフィカルなプランは、短いテキストプランや長いテキストプランよりも多くの情報を提供します。

統計情報付きのグラフィカルプランによるパフォーマンス分析 [218 ページ]
統計情報付きのグラフィカルプランを使用して、データベースのパフォーマンス上の問題を特定できます。

グラフィカルプランのノードの詳細情報 [220 ページ]
グラフィカルプランでノードに関する詳細情報を表示することができます。

グラフィカルプランの選択性情報 [221 ページ]
グラフィカルプランで選択制情報を表示することができます。

グラフィカルプランの表示 [223 ページ]
Interactive SQL でグラフィカルプランを表示します。

関連情報

実行プランのコンポーネント [230 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.12.5.1 統計情報付きのグラフィカルプラン

グラフィカルなプランは、短いテキストプランや長いテキストプランよりも多くの情報を提供します。

統計情報付きのグラフィカルなプランを生成する場合は負荷が高くなりますが、クエリの実行時にデータベースサーバがモニタしている実際のクエリ実行統計が表示されます。このため、オプティマイザがアクセスプランの作成時に使用する推定を、実行中にモニタされた実際の統計と直接比較できます。推定と実際の統計が大きく異なる場合は、情報不足のためにオプティマイザが正確にクエリのコストを推定できず、その結果非効率な実行プランになっていることがあります。

統計情報付きのグラフィカルなプランを生成するには、データベースサーバで文を実行する必要があります。実行時間が長い文のグラフィカルプランを生成すると、かなり時間がかかる場合があります。UPDATE、INSERT、または DELETE 文の場合は、文の読み込み専用の部分だけが実行されます。テーブルの操作は実行されません。ただし、文にユーザ定義関数が含まれる場合は、クエリの一部としてユーザ定義関数が実行されます。ユーザ定義関数に副次的影響 (たとえば、ローの変更、テーブルの作成、コンソールへのメッセージ送信など) がある場合、統計情報付きのグラフィカルなプランを取得するときにこれらの変更が実行されます。統計情報付きのグラフィカルなプランの取得後に ROLLBACK 文を実行して、これらの副次的影響を元に戻すことができる場合があります。

1.3.1.12.5.2 統計情報付きのグラフィカルプランによるパフォーマンス分析

統計情報付きのグラフィカルプランを使用して、データベースのパフォーマンス上の問題を特定できます。

クエリの実行上の問題の特定

クエリの実行に影響するデータベースオプションやその他のグローバルな設定をルート演算子ノードについて表示できます。

選択性のパフォーマンスの確認

述部の選択性 (条件式) とは、条件を満たすローの割合のことです。推定される述部の選択性が提供する情報に基づいて、オプティマイザはコストの推定を行います。オプティマイザの正常な処理には、正確な選択性推定が不可欠です。たとえば、オプティマイザが述部の選択性が高い (5% の選択性など) と誤って推定しても、実際には述部の選択性がかなり低い (50% など) 場合は、パフォーマンスが低下することがあります。選択性推定は正確ではない場合がありますが、極端に大きな誤差は問題がある可能性を示しています。

クエリの重要な部分の選択性情報が不正確であると判断した場合、CREATE STATISTICS を使用してカラムの新しい統計値セットを生成できます。まれに、明示的な選択性推定を指定したい場合があります。ただし、この方法では、後で統計を更新するときに問題が発生する可能性があります。

クエリがバイパスクエリだと判断された場合、選択性統計値は表示されません。

次のような場合に不適切な選択性の兆候が見られます。

RowsReturned の実際値と推定値

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

RowsReturned は結果セット内のロー数です。*RowsReturned* 統計値は、ツリーの先頭にあるルートノードのテーブル内に表示されます。推定ローカウントが実際のローカウントと大きく異なる場合は、このノードまたはサブツリーに接続された述部の選択性が正しくない可能性があります。

述部選択性の実際値と推定値

「述部」というサブヘッダを検索して、述部の選択性を確認します。

ヒストグラムが存在しないベースカラムに対する述部がある場合は、ヒストグラムを作成するために CREATE STATISTICS 文を実行すると問題を修正できる場合があります。

選択性の誤差に問題が残る場合は、クエリテキストに述部とともにユーザ推定選択性を指定することを検討してください。

推定ソース

選択性推定のソースも、統計情報ウィンドウ枠の述部サブヘッダの下にリストされています。

述部選択性推定のソースが *Guess* の場合、オプティマイザには述部のフィルタリングの特性の判断に使用できる情報がないため、ヒストグラムがないなどの問題を示す可能性があります。推定ソースが *Index* で、選択性推定が正しくない場合は、インデックスが偏っていることが問題である可能性があります。REORGANIZE TABLE 文を使用してインデックスの断片化を解除すると改善できる場合があります。

キャッシュのパフォーマンスの確認

キャッシュの読み込み数 (*CacheRead* フィールド) とヒット数 (*CacheHits* フィールド) が同じである場合は、この SQL 文のために処理されるすべてのオブジェクトがキャッシュに保持されています。キャッシュの読み込み数がヒット数よりも多い場合は、テーブルまたはインデックスページがサーバのキャッシュに存在しないため、データベースサーバがこれらをディスクから読み込んでいることを示します。これは、ハッシュジョインなどの場合に予想されます。ネストループジョインなどの場合、キャッシュヒット率が低いときは、クエリを効率的に実行するためのキャッシュ (バッファプール) の不足を示している可能性があります。この場合、サーバのキャッシュサイズを増やすと改善できることがあります。

無効なインデックスの識別

クエリ実行プランからは、インデックスがパフォーマンスの向上に役立っているかどうかが不明な場合がよくあります。スキャンベースのクエリ操作の中には、インデックスを使用しないで多くのクエリに対して最高のパフォーマンスを提供するものもあります。

データの断片化の問題の特定

Runtime および *FirstRowRunTime* の実際値と推定値は、ルートノード統計値で提供されます。ノードに *RunTime* が存在する場合、サブツリー統計セクションにはこの値だけが表示されます。

RunTime の解釈は、表示される統計セクションによって異なります。ノード統計の場合、*RunTime* はこのノードだけを実行している間に、対応する演算子が費やした累積時間です。サブツリー統計 の場合、*RunTime* はこのノードの直下にある演算子のサブツリー全体に対して費やされた合計実行時間を表します。したがって、ほとんどの演算子では *RunTime* と *FirstRowRunTime* は独立した測定値で、個別に分析する必要があります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

FirstRowRunTime は、このノードの中間結果の最初のローを作成するために要した時間です。

テーブルスキャンまたはインデックススキャンで、ノードの *RunTime* が予想よりも大きい場合、REORGANIZE TABLE 文を実行するとパフォーマンスを向上できることがあります。sa_table_fragmentation() システムプロシージャと sa_index_density() システムプロシージャを使用して、テーブルまたはインデックスが断片化されているかどうかを判断できます。

関連情報

実行プランのコンポーネント [230 ページ]
オプティマイザの仕組み [203 ページ]
実行プランのコンポーネント [230 ページ]
グラフィカルプランの選択性情報 [221 ページ]

1.3.1.12.5.3 グラフィカルプランのノードの詳細情報

グラフィカルプランでノードに関する詳細情報を表示することができます。

右側の詳細ウィンドウ枠と高度な詳細ウィンドウ枠に各ノードに関する詳細情報が表示されます。詳細ウィンドウ枠の次の 3 つのメインセクションにノードの統計値が表示されます。

- ノード統計
- サブツリー統計
- オプティマイザ統計

ノード統計は、特定のノードの実行に関する統計です。プランノードには *[詳細]* ウィンドウ枠があり、演算子に対する推定値、実測値、および平均値の統計が表示されます。どのノードについても操作を複数回実行でできます。たとえば、ネストループジョインノードの右側にリーフノードが表示されると、リーフノード演算子からローを複数回フェッチできます。この場合、リーフノード (逐次、インデックス、または RowID スキャンノード) の *[詳細]* ウィンドウ枠には、実行ごとの (平均の) 統計と実行時の実際の累積統計の両方が含まれます。

ノードがリーフノードではない場合、ノードは他のノードの中間結果を消費し、*[詳細]* ウィンドウ枠の *[サブツリー統計]* セクションには、このノードのサブツリー全体に対する推定と実際の累積統計が表示されます。SQL 要求全体を表すオプティマイザ統計情報は、ルートノードだけに存在します。オプティマイザ統計値は特に文の最適化に関連しており、最適化のゴール設定、最適化レベル設定、検討するプラン数などの値があります。

注文日で顧客を並べ替える次のクエリを考えてみます。

```
SELECT GROUPO.Customers.GivenName, GROUPO.Customers.Surname,
GROUPO.SalesOrders.OrderDate
FROM Customers KEY JOIN SalesOrders
WHERE CustomerID > 100
ORDER BY OrderDate
```

このクエリのグラフィカルプランでは、*Hash Join (JH)* ノードが選択され、このノードに関連する情報だけが詳細ウィンドウ枠に表示されています。述部の説明によれば、Hash Join ノードに適用された述部は `Customers.ID = SalesOrders.CustomerID : 0.79365% Statistics | Join` です。Customers ノードをクリックすると、スキャン

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

述部によれば、Customers ノードに適用された述部は `Customers.ID > 100 : 100% Index;` であることがわかります。

> **注記**
>
> 前述の例のクエリを実行すると、プランビューアに表示されたのとは異なるプランが取得される場合があります。データベースの設定や最近実行されたクエリなど、多くの要因によってオプティマイザによるプランの選択が影響を受ける可能性があります。

高度な詳細ウィンドウ枠に表示される情報は、各演算子によって異なります。ルートノードの場合、高度な詳細ウィンドウ枠に、クエリが最適化されたときに有効になっていた接続オプションの設定が表示されます。他の種類のノードでは、高度な詳細ウィンドウ枠に、特定のノードの処理で検討されたインデックスまたはマテリアライズドビューに関する情報が表示される場合があります。

グラフィカルなプランの各ノードのコンテキスト別ヘルプを表示するには、ノードを右クリックして *[ヘルプ]* をクリックします。

> **注記**
>
> クエリがバイパスクエリとして認識されている場合、一部の最適化ステップがバイパスされ、クエリオプティマイザセクションと述部セクションはどちらもグラフィカルプランに表示されません。

関連情報

オプティマイザの仕組み [203 ページ]
高度: クエリ実行プラン [210 ページ]
実行プランのコンポーネント [230 ページ]
グラフィカルプランの表示 [223 ページ]

1.3.1.12.5.4 グラフィカルプランの選択性情報

グラフィカルプランで選択制情報を表示することができます。

次の例では、選択されたノードが Departments テーブルのスキャンを表し、統計情報ウィンドウ枠には探索条件、選択性推定、実際の選択性として述部が表示されています。

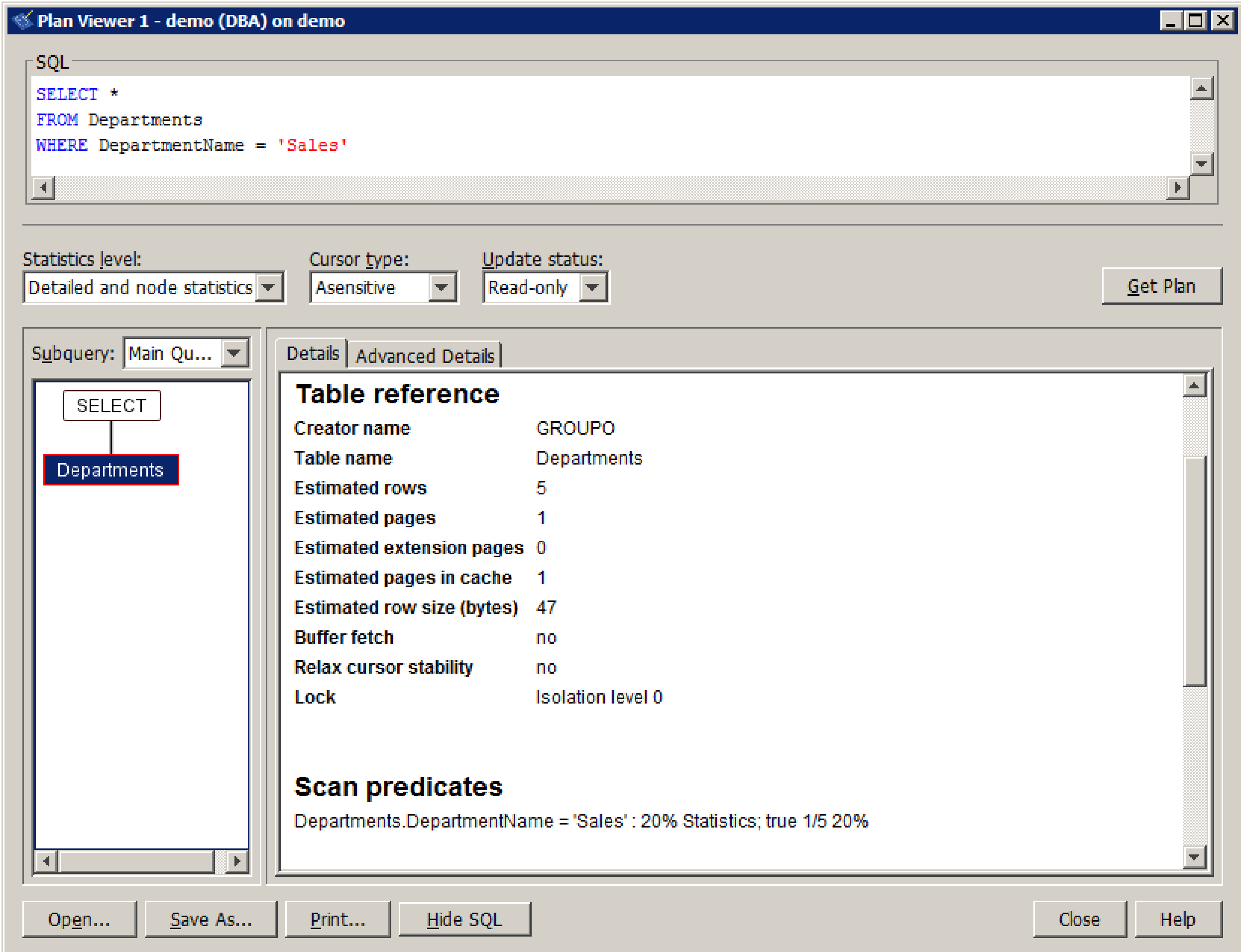
詳細ウィンドウ枠のノード統計、サブツリー統計、オプティマイザ統計の 3 つのセクションに、各ノードに関する統計値が表示されます。

ノード統計は、特定のノードの実行に関連しています。プラン内のノードがリーフノードではなく、他のノードの中間結果を消費する場合、*[詳細]* ウィンドウ枠の *[サブツリー統計]* セクションには、対象ノードのサブツリー全体に対する推定と実際の累積統計が表示されます。オプティマイザ統計情報はルートノードだけに存在し、SQL 要求全体を表します。

バイパスクエリには、選択性の情報が表示されない場合があります。

アクセスプランは、データベース内で使用可能な統計値によって決まります。また、この統計値は、どのクエリが実行済かによります。ここでは、各種の統計値やプランを確認できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.



この述部記述は次のとおりです。

```
Departments.DepartmentName = 'Sales' : 20% Column; true 1/5 20%
```

この述部は次のように解釈できます。

- `Departments.DepartmentName = 'Sales'` は述部です。
- `20%` は、オプティマイザによる選択性推定です。つまり、オプティマイザは 20% のローが述部を満たすという推定に基づいてクエリアクセスを選択しています。
  これは、ESTIMATE 関数で得られる出力と同じです。
- `Column` は推定ソースです。これは、ESTIMATE_SOURCE 関数で得られる出力と同じです。
- `true 1/5 20%` は、実行時における述部の実際の選択性です。述部は 5 回評価され、その内の 1 回が TRUE だったので、実際の選択性は 20% になります。
  実際の選択性が推定値と大幅に異なる場合、および述部の評価回数が非常に多い場合は、不正確な推定によりクエリのパフォーマンスに重大な問題が発生している可能性があります。述部の統計値の収集により、オプティマイザにその選択の基礎となる良質な情報を提供することで、パフォーマンスを改善することができます。

i 注記

統計情報付きのグラフィカルプランではないグラフィカルプランを選択すると、最後の 2 つの統計情報は表示されません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

選択性推定ソース [206 ページ]
オプティマイザの仕組み [203 ページ]

1.3.1.12.5.5 グラフィカルプランの表示

Interactive SQL でグラフィカルプランを表示します。

手順

1. Interactive SQL を起動して、データベースに接続します。
2. ツール > プランビューアを開く をクリックするか、Shift + F5 キーを押します。
3. *SQL* ウィンドウ枠に文を入力します。
4. 統計レベル、カーソルタイプ、更新のステータスを選択します。
5. プランの取得をクリックします。

結果

グラフィカルプランが表示されます。

関連情報

実行プランのコンポーネント [230 ページ]
高度: クエリ実行プラン [210 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.12.6 チュートリアル: Interactive SQL におけるプランの比較

Interactive SQL のプラン比較ウィンドウを使用して、クエリの実行プランを比較できます。

前提条件

Interactive SQL のプラン比較ツールを使用するために必要な権限はありません。

このチュートリアルでは、sa_flush_cache システムプロシージャを実行するため、SERVER OPERATOR システム権限が必要です。また、サンプルデータベース内の以下のテーブルおよびマテリアライズドビューがプラン生成時に問い合わせるオブジェクトとなるため、これらに対する SELECT 権限も必要です。

- SalesOrders テーブル
- Employees テーブル
- SalesOrderItems テーブル
- Products テーブル
- MarketingInformation マテリアライズドビュー

コンテキスト

テーブルの状態、オプティマイザの設定、およびデータベースキャッシュの内容など多くの変数が、同一の 2 つのクエリの実行に影響を与える場合があります。同じように、2 つのデータベースサーバまたは 2 つのバージョンのソフトウェアでクエリを実行すると、著しく異なる結果となる場合があります。

このような環境で、実行プランの保存、比較、分析を行って、差異が発生した箇所を把握することができます。Interactive SQL のプラン比較ツールを使用して、保存した 2 つの実行プランを比較し、その差異を特定することができます。

このチュートリアルでは、プランビューアとプラン比較ツールを使用して、あるクエリに関する 2 つの異なる実行プランを作成し、それらを比較します。通常の操作中は、同じクエリに関する 2 つのプランを数分間隔で保存することはないでしょう。通常は、しばらく前にあるクエリについてプランを保存しており、次に新しいプランを保存しようとするときに、プランを比較し、どのような差異があるのかを把握することができます。

このセクションの内容:

レッスン 1: 2 つの実行プランの作成およびファイルへの保存 [225 ページ]
Interactive SQL を使用して、同じクエリに対して 2 つのプランを作成します。

レッスン 2: プラン比較の分析 [226 ページ]
Interactive SQL のプラン比較ツールで比較された 2 つのプランの結果を分析します。

レッスン 3: 演算子とクエリの手動による一致および一致解除 [228 ページ]
プラン比較ツールを使用して、2 つのプランの比較時に一致している演算子とクエリをすべて検出します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

実行プランのコンポーネント [230 ページ]

1.3.1.12.6.1 レッスン 1: 2 つの実行プランの作成およびファイルへの保存

Interactive SQL を使用して、同じクエリに対して 2 つのプランを作成します。

前提条件

このチュートリアルの冒頭に一覧表示されているロールと権限を持っている必要があります。

コンテキスト

なし。

手順

1. Interactive SQL で、以下の文を実行してデータベースのキャッシュをクリアしてください。

```
CALL sa_flush_cache();
```

2. 1 つ目のプランを生成し、保存します。
   a. ツール > プランビューア を開くをクリックします。
   b. プランビューアで、次のクエリを *SQL* に貼り付け、統計レベルから詳細とノードの統計を選択し、プランの取得をクリックします。

```
SELECT DISTINCT EmployeeID, GivenName, Surname
  FROM GROUPO.SalesOrders WITH (INDEX (SalesOrdersKey)), GROUPO.Employees,
GROUPO.SalesOrderItems
  WHERE SalesRepresentative = EmployeeID and
   SalesOrders.ID = SalesOrderItems.ID and
   SalesOrderItems.ProductID = (SELECT ProductID
    FROM GROUPO.Products, GROUPO.MarketingInformation
    WHERE Name = 'Tee Shirt' AND
    Color = 'White' AND
    Size = 'Small' AND
    MarketingInformation.Description LIKE '%made of recycled water bottles%');
```

   このクエリは、ペットボトルから作った S サイズの白い T シャツなどを担当している販売員の EmployeeID、GivenName、および Surname を返します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

c. 名前を付けて保存をクリックし、FirstPlan というファイルにプランを保存します。
3. プランビューアは閉じないでください。
4. 2 つ目のプランを生成し、保存します。
    a. Interactive SQL で、同じクエリを *SQL* 文に貼り付け、実行します。
    b. プランビューアに変更したら、プラン取得をクリックして再度プランを取得し、SecondPlan というファイルに保存します。

結果

同じクエリに対して 2 つのプランを作成し、それぞれ別のファイルに保存しました。

次のステップ

次のレッスンに進みます。

1.3.1.12.6.2 レッスン 2: プラン比較の分析

Interactive SQL のプラン比較ツールで比較された 2 つのプランの結果を分析します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

ファイルに保存されていれば、2 つのプランを比較するのにデータベースサーバに接続する必要はありません。

手順

1. 前のレッスンで作成した 2 つのプランについて比較を生成します。
    a. Interactive SQL で、ツール > プラン比較 をクリックし、プラン比較ツールを開きます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

b. *プラン 1:* で、作成したファイル `FirstPlan.saplan` を参照し選択します。
c. *プラン 2:* で、作成したファイル `SecondPlan.saplan` を参照し選択します。
d. プラン比較をクリックします。

プラン比較ツールは、2 つのプランの間のサブクエリと演算子を一致させようとします。この照合結果は、比較概要領域に一覧表示されます。

比較概要にある項目の左側の数字は、プラン比較ツールで参照した演算子またはクエリの一致を特定する照合識別子です。項目に照合識別子がない場合は、その項目はプラン比較ツールで一致しなかったことになります。

2. 比較概要の番号の一覧を使用して、2 つのプランの間の演算子とサブクエリの違いを分析します。一致する演算子とクエリは同じ行に配置されています。ただし、プラン比較ツールで一致しているものを検討するには、照合識別子が最も良い指標となります。たとえば、プラン比較ツールを使用して、両方のプランで SELECT 演算子が一致した場合、その一致に対して識別子 7 が与えられます。

エントリ間の記号で、各一致に関する詳細が表されます。
- 等号否定 (≠) は、いずれのプランにも演算子が存在しますが、(プラン図の下の詳細ウィンドウ枠にある) 推定値カラムの値は異なるということを表します。両方のプランに演算子が存在していれば、ほぼすべての場合に等号否定が表示されます。これは、2 つのクエリ実行が全く同じ推定値となる可能性は非常に低いためです。推定値は、10 分の 1 から 1,000 分の 1 秒、あるいはそれ以上の精度で測定されます。
- 等号 (=) は、演算子が両方のプランに存在し、推定値カラムの値が同一であることを表しています。
- 大なり不等号 (>) は、演算子が 1 つ目のプランにのみ存在することを示しています。
- 小なり不等号 (<) は、演算子が 2 つ目のプランにのみ存在することを示しています。
- ダッシュ記号 (-) は、サブクエリのノードに一致していることを表しています。

比較概要ウィンドウ枠か、グラフィカルなプラン図 (*1. FirstPlan* および *2. SecondPlan*) のどちらかでローを選択すると、下部にある詳細タブと高度な詳細タブに演算子のプロパティ値が表示されます。

3. 比較概要ウィンドウ枠かグラフィカルなプラン図のいずれかで演算子をクリックして、2 つのプラン間の差異を分析します。

たとえば、比較概要で、`FirstPlan` に一覧表示されている *3: NestedLoopsJoin* をクリックします。これにより、*3: HashJoin* で `SecondPlan` が選択されます。これらのノードが一致していると識別されるためです。

4. 詳細タブと高度な詳細タブを使用して、2 つのプラン間の統計的な差異を分析します。
- 両方のプランで統計が使用可能で、かつ値が同じ場合、特別な書式は設定されません。
- 黄色の強調表示は、統計が片方のプランのみにあることを示しています。欠落している統計が、2 つのプラン間でクエリの処理方法がどのように異なるのかを把握する手がかりとなります。
- 濃い赤の強調表示は、統計上の大きな相違点を示しています。
- 明るい赤の強調表示は、統計上の小さな相違点を示しています。

このチュートリアルの 2 つのプランでは、プラン間の重要な違いは、ほぼすべてクエリの 2 回目の実行によるものであり、データベースサーバではキャッシュのデータを使用できたため、ディスクからデータを読み込む必要はありませんでした。そのため、*SecondPlan* にはメモリ使用を表す統計がありますが、*FirstPlan* にはその統計はありません。また、一致しているすべてのノードにおいて、*DiskReadTime* と *DiskRead* の値には大きな違いがあります。`SecondPlan` のこれらの演算子の値が著しく低いのは、データがディスクではなくメモリから読み込まれたためです。

結果

Interactive SQL のプラン比較ツールを使用して、保存されているクエリの 2 つの実行プランを比較し、結果を分析しました。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

次のレッスンに進みます。

関連情報

グラフィカルプラン [216 ページ]
実行プランのコンポーネント [230 ページ]

1.3.1.12.6.3 レッスン 3: 演算子とクエリの手動による一致および一致解除

プラン比較ツールを使用して、2 つのプランの比較時に一致している演算子とクエリをすべて検出します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

プラン比較ツールでは、演算子やサブクエリ名だけでなく、演算子の生成結果や、その結果がこの後プラン内でどのように使用されるかという要素によっても、一致を検出します。たとえばこのチュートリアルでは、`FirstPlan` の *NestedLoopJoin* 演算子が `SecondPlan` の *HashJoin* 演算子と一致していますが、これは、異なるアルゴリズムを使用していても同じ結果が生成されているためです。

プラン比較ツールでは、期待される一致が特定されない場合があります。手動で統計を比較して一致を発見することができます。また、演算子やサブクエリでも手動で行うことができます。プラン比較ツールが作成した一致を削除することもできます。

手順

1. 比較概要ウィンドウ枠で、演算子一覧の一番下までスクロールします。`FirstPlan` の *HashFilter* 一覧の最後の 2 項目は、`SecondPlan` のどの演算子とも一致していません。同様に、`SecondPlan` のリストで、*HashFilter* の最後の 2 つの演算子は、`FirstPlan` の演算子に合致していません。
2. `FirstPlan` で 1 つ目の *HashFilter* 演算子をクリックし、詳細ウィンドウ枠でハッシュリストの値を確認します。ここでは、値は *Employees.EmployeeID integer* となっています。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

3. SecondPlan で 1 つ目の *HashFilter* 演算子をクリックし、詳細ウィンドウ枠でハッシュリストの値を確認します。ここでは、値は *Employees.EmployeeID integer* となっています。

   これは、FirstPlan の *HashFilter* 演算子が、SecondPlan の *HashFilter* 演算子の最初のインスタンスと一致していることを意味しています。
4. 次のように演算子の一致を行います。
   a. FirstPlan のグラフィカルプランで、*HF* ノードをクリックして選択します。これが、FirstPlan の *HashFilter* 演算子です。
   b. SecondPlan のグラフィカルプランで、*JH* (ジョインハッシュ) ノードの子ノードである *HF* ノードをクリックして選択します。これが、FirstPlan の *HashFilter* 演算子に一致する *HashFilter* 演算子です。
   c. 演算子の一致をクリックします。

      プラン比較ツールで、手動で一致を生成し、照合識別子 (SubQ 1 など) を割り当てます。比較概要ウィンドウ枠が更新されて新しい一致が反映され、演算子が同じ行に位置付けられます。
   d. 比較概要ウィンドウ枠の FirstPlan と SecondPlan の最下部にある残りの *HashFilter* 演算子に対して、一致の手順を同じように繰り返します。
5. 一致を削除するには、関連する演算子を選択し、演算子の一致解除をクリックします。プラン比較ツールで一致した演算子と同様、一致している演算子から手動で削除することができます。
6. 演算子と同じ手順で、サブクエリの一致の作成や削除を手動で行います。ただし、演算子ではなくクエリの一致およびクエリの一致解除ボタンを使用します。

結果

プラン比較ツールで、演算子やサブクエリの一致および一致解除を行う方法を学習しました。

関連情報

グラフィカルプラン [216 ページ]
実行プランのコンポーネント [230 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.1.12.7 実行プランのコンポーネント

実行プランには多数の省略形があります。

短いテキストプラン	長いテキストプラン	その他の情報
	見積済最良プラン	オプティマイザは、特定のクエリのアクセスプランを生成してコストを計算します。この処理の最中、現在の最良プランが、より低いコストが推定される新しい最良プランで置き換えられる場合があります。最後の最良プランが、文の実行に使用する実行プランになります。見積済最良プランは、現在の最良プランよりも優れたプランをオプティマイザが検出した回数を示しています。数字が小さい場合は、最良プランが列挙プロセスの早い段階で決定されたことを示しています。オプティマイザは特定の文のクエリブロックごとに 1 回以上の列挙プロセスを起動するため、見積済最良プランは累積回数を表します。
	見積済プラン	オプティマイザによって生成される多数のプランが、それまでに検出された最良プランと比較してコストが高いと判断されます。見積済プランは、指定された文の列挙プロセスの最中にオプティマイザが検討した一部のプランまたは完全なプランの数を表します。
**	**	完全なインデックススキャン。インデックススキャンではすべてのローが読み込まれます。
DELETE	*Delete*	削除操作のルートノード。
DistH	*HashDistinct*	*HashDistinct* は 1 つの入力を受け取り、すべての排他ローを返します。
DistO	*OrderedDistinct*	*OrderedDistinct* では、各ローが読み込まれ、前のローと比較されます。両者が同じであれば後の入力ローは無視され、それ以外の場合は出力されます。
DP	*DecodePostings*	*DecodePostings* は、テキストインデックス内の単語の位置情報を復号化します。
DT	*DerivedTable*	*DerivedTable* は、特にクエリに 1 つ以上の外部ジョインが含まれる場合に、クエリ書き換え最適化などのさまざまな理由でプランに表示される場合があります。
EAH	*HashExceptAll*	SQL の集合差異演算子 EXCEPT のハッシュベースの実装が使用されたことを示します。
EAM	*MergeExceptAll*	SQL の集合差異演算子 EXCEPT のソートベースの実装が使用されたことを示します。
EH	*HashExcept*	SQL の集合差異演算子 EXCEPT のハッシュベースの実装が使用されたことを示します。
EM	*MergeExcept*	SQL の集合差異演算子 EXCEPT のソートベースの実装が使用されたことを示します。
Exchange	*Exchange*	SELECT 文の処理時にクエリ内並列処理が使用されたことを示します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

短いテキストプラン	長いテキストプラン	その他の情報
Filter	*Filter*	任意のタイプの述部、subselect を含む比較、EXISTS と NOT EXISTS の各サブクエリ (その他の形式の限定サブクエリ) などの探索条件の適用を示します。
GrByH	*HashGroupBy*	*HashGroupBy* では、グループごとに 1 つのローで構成されるインメモリハッシュテーブルが構築されます。入力ローが読み込まれると、ワークテーブル内で関連グループが検索されます。集合関数が更新され、グループローがワークテーブルに再び書き込まれます。グループレコードが見つからなければ、新しいグループレコードが初期化され、ワークテーブルに挿入されます。
GrByHClust	*HashGroupByClustered*	入力テーブルのグループ化カラム内の値はクラスタ化されているため、似た値が互いに近接して現れることがあります。*ClusteredHashGroupBy* は、このクラスタ化を利用します。
GrByHP	*ParallelHashGroupBy*	*HashGroupBy* の変形。
GrByHSets	*HashGroupBySets*	GROUPING SETS クエリを実行する場合、*HashGroupBy* の変形である *HashGroupBySets* が使用されます。
GrByO	*OrderedGroupBy*	*OrderedGroupBy* では、グループ化カラムによって順序付けされた入力が読み込まれます。各ローは、読み込まれるたびに前のローと比較されます。グループ化カラムが一致すると、現在のグループが更新されます。それ以外の場合は、現在のグループが出力され、新しいグループが開始されます。
GrByOSets	*OrderedGroupBySets*	GROUPING SETS クエリを実行する場合、*OrderedGroupBy* の変形である *OrderedGroupBySets* が使用されます。
GrByS	*SingleRowGroupBy*	GROUP BY を指定しなければ、単一ローの集合の生成に *SingleRowGroupBy* が使用されます。各入力ローに対して単一のグループローがメモリに格納され、更新されます。
GrBySSets	*SortedGroupBySets*	*SortedGroupBySets* は、GROUPING SETS を含む OLAP クエリを処理する場合に使用されます。
HF	*HashFilter*	ハッシュフィルタ (ブルームフィルタ) が使用されたことを示します。
HFP	*ParallelHashFilter*	ハッシュフィルタ (ブルームフィルタ) が使用されたことを示します。
HTS	*HashTableScan*	ハッシュテーブルスキャンが使用されたことを示します。
IAH	*HashIntersectAll*	SQL の集合差異演算子 INTERSECT のハッシュベースの実装が使用されたことを示します。
IAM	*MergeIntersectAll*	SQL の集合差異演算子 INTERSECT のソートベースの実装が使用されたことを示します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

短いテキストプラン	長いテキストプラン	その他の情報
IH	*HashIntersect*	SQL の集合差異演算子 INTERSECT のハッシュベースの実装が使用されたことを示します。
IM	*MergeIntersect*	SQL の集合差異演算子 INTERSECT のソートベースの実装が使用されたことを示します。
IN	*InList*	*InList* は、インデックスを使用して IN リスト述部を満たすことができる場合に使用されます。
`table-name<index-name>`	*IndexScan*、*ParallelIndexScan*	グラフィカルなプランで、インデックススキャンの場合は台形の中にインデックス名が表示されます。
INSENSITIVE	*Insensitive*	
INSERT	*Insert*	*INSERT* 操作のルートノード。
IO	*IndexOnlyScan*、*ParallelIndexOnlyScan*	クエリを満たすために必要なすべてのデータを含んだインデックスをオプティマイザが使用したことを示します。
JH	*HashJoin*	*HashJoin* は、2 つの入力のうち小さい方のインメモリハッシュテーブルを作成してから、大きい方の入力を読み込みます。次に、インメモリハッシュテーブルを調査して一致するローを検索します。見つかったローはワークテーブルに書き込まれます。小さい方の入力がメモリに収まらない場合は、*HashJoin* によって両方の入力が小さなワークテーブルに分割されます。これらの小さくなったワークテーブルは、小さい方の入力がメモリに収まるようになるまで再帰的に処理されます。
JHS	*HashSemijoin*	*HashSemijoin* は、左側と右側のセミジョインを実行します。
JHSP	*ParallelHashSemijoin*	*HashJoin* の変形。
JHFO	*Full Outer HashJoin*	*HashJoin* の変形。
JHA	*HashAntisemijoin*	*HashAntisemijoin* は、左側と右側の非セミジョインを実行します。
JHAP	*ParallelHashAntisemijoin*	*HashJoin* の変形。
JHO	*Left Outer HashJoin*	*HashJoin* の変形。
JHP	*ParallelHashJoin*	*HashJoin* の変形。
JHPO	*ParallelLeftOuterHashJoin*	*HashJoin* の変形。
JHR	*RecursiveHashJoin*	*HashJoin* の変形。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

短いテキストプラン	長いテキストプラン	その他の情報
JHRO	*RecursiveLeftOuterHashJoin*	*HashJoin* の変形。
JM	*MergeJoin*	*MergeJoin* は 2 つの入力を読み込みます。このとき、2 つの入力はどちらもジョイン属性で順序付けされます。左側の入力のローごとに、右側の入力のローにソート順にアクセスすることで、一致する右側のローをすべて読み込みます。
JMFO	*Full Outer MergeJoin*	*MergeJoin* の変形。
JMO	*Left Outer MergeJoin*	*MergeJoin* の変形。
JNL	*NestedLoopsJoin*	*NestedLoopsJoin* では、左側のローごとに右側全体が読み込まれ、左側と右側のジョインが計算されます。
JNLA	*NestedLoopsAntisemijoin*	*NestedLoopsAntisemijoin* は左側のローごとに右側をスキャンして、入力をジョインします。
JNLFO	*Full Outer NestedLoopsJoin*	*NestedLoopsJoin* の変形。
JNLO	*Left Outer NestedLoopsJoin*	*NestedLoopsJoin* の変形。
JNLS	*NestedLoopsSemijoin*	*NestedLoopsSemijoin* は左側のローごとに右側をスキャンして、入力をジョインします。
KEYSET	*Keyset*	キーセット駆動型カーソルを指定します。
LOAD	*Load*	ロード操作のルートノード。
MultiIdx	*MultipleIndexScan*	論理演算子 AND または OR で組み合わされた探索条件のセットを含むクエリを満たすために、複数のインデックスを使用できる、または使用する必要がある場合は、*MultipleIndexScan* を使用します。
OpenString	*OpenString*	*OpenString* は、OPENSTRING 句を含む SELECT 文の FROM 句で使用されます。
	Optimization Time	指定された文のすべての列挙プロセスにオプティマイザが費やした合計時間。
PC	*ProcCall*	プロシージャコール (テーブル関数)。
PreFilter	*PreFilter*	フィルタでは、任意のタイプの述部、subselect を含む比較、EXISTS と NOT EXISTS の各サブクエリ (その他の形式の限定サブクエリ) などの探索条件が適用されます。
R	*R*	リバースインデックススキャン。インデックススキャンでローがインデックスから逆の順序で読み込まれます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

短いテキストプラン	長いテキストプラン	その他の情報
RL	*RowLimit*	*RowLimit* は、入力の最初の n 個のローを返し、残りのローは無視します。ローの制限は、SELECT 文の TOP n または FIRST 句によって設定されます。
ROWID	*RowIdScan*	グラフィカルプランで、ロー ID スキャンの場合は長方形の中にテーブル名が表示されます。
ROWS	*RowConstructor*	*RowConstructor* は、他のアルゴリズムへの入力として使用できる仮想ローを作成する特殊な演算子です。
RR	*RowReplicate*	*RowReplicate* は、EXCEPT ALL や INTERSECT ALL などの集合操作の実行時に使用されます。
RT	*RecursiveTable*	再帰テーブルがクエリ内の WITH 句の結果として使用されたことを示します。WITH 句は、再帰的な UNION クエリに使用されました。
RU	*RecursiveUnion*	*RecursiveUnion* は、再帰的な UNION クエリの実行中に使用されます。
SELECT	*Select*	*SELECT* 操作のルートノード。
seq	*TableScan*、*ParallelTableScan*	グラフィカルなプランで、テーブルスキャンの場合は長方形の中にテーブル名が表示されます。
Sort	*Sort*	インデックスソートまたはマージソート。
SrtN	*SortTopN*	*SortTopN* は、TOP N 句と ORDER BY 句を含むクエリで使用されます。
TermBreak	*TermBreak*	全文検索の *TermBreaker* アルゴリズム。
UA	*UnionAll*	*UnionAll* では、重複に関係なく各入力からローが読み込まれ、出力されます。このアルゴリズムは、UNION 文と UNION ALL 文を実装するために使用されます。
UPDATE	*Update*	*UPDATE* 操作のルートノード。
Window	*Window*	*Window* は、Window 関数を使用する OLAP クエリを評価する場合に使用します。
Work	*Work table*	中間結果を表す内部ノード。

オプティマイザ統計フィールドの説明

グラフィカルプランのオプティマイザ統計、ローカルオプティマイザ統計、グローバルオプティマイザ統計セクションの説明を次に示します。これらの統計は、データベースサーバの状態と、最適化プロセスについての情報を提供します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

フィールド	説明
最適化時間の構築	最適化内部の構築に費やされる時間。
クリーンアッププランタイム	クリーンアップフェーズ中に費やされた時間。
見積済プラン	コストを部分的または完全に推定したこの要求に対して、オプティマイザが検討した異なるアクセスプランの数。見積済最良プランと同様、この値が小さい場合は通常、最適化時間の短縮を示し、この値が大きい場合は SQL クエリがより複雑であることを示します。 見積済最良プラン、見積り済みプラン、最適化時間の値が 0 の場合、文は最適化されていません。データベースサーバは文をバイパスし、文の最適化を行わず実行プランを生成しています。
見積済最良プラン	クエリオプティマイザが異なるクエリ実行方式を列挙するとき、現在の方式の前に検出された最良な方式よりも推定された負荷が低い方式が検出された回数を追跡します。特定のクエリに対してこの発生回数を予測することは困難ですが、この数値が小さい場合、オプティマイザのアルゴリズムによって検索領域が大幅に削減され、通常は、最適化時間が短縮されることを示します。オプティマイザは特定の文のクエリブロックごとに 1 回以上の列挙プロセスを起動するため、見積済最良プランは累積回数を表します。 見積済最良プラン、見積済プラン、最適化時間の値が 0 の場合、SQL Anywhere オプティマイザによって文は最適化されていません。データベースサーバはこの文をバイパスし、文の最適化を行わずに実行プランを生成したか、または文のプランはキャッシュされました。
ランタイムの見積もり	見積フェーズ中に費やされた時間。
CurrentCacheSize	最適化時のデータベースサーバのキャッシュサイズ (キロバイト単位)。
予測キャッシュページ数	文の処理に使用できる現在の推定キャッシュサイズ。 非効率なアクセスプランを削減するため、オプティマイザは現在のキャッシュサイズの半分は選択された文の処理に使用できると想定します。
予測最大コスト	この最適化の予測最大コスト。
予測最大コストランタイム	予測最大コストフェーズ中に費やされた時間。
予測クエリメモリページ	文に使用できる予想クエリメモリページ。クエリメモリは、ソート、ハッシュジョイン、ハッシュ GROUP BY、ハッシュ DISTINCT などのクエリ実行アルゴリズムに使用されます。
予測されるタスク	クエリ内並列処理で使用可能な予測されるタスクの数。
ジョイン列挙によって使用された追加ページ	削除によるジョイン列挙によって使用された追加メモリページ数。
最終プラン構築時間	最終プランの構築に費やされる時間。
初期化ランタイム	初期化フェーズ中に費やされた時間。

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

フィールド	説明
isolation_level	文の独立性レベル。文の独立性レベルは、同じトランザクション内の他の文と異なる場合があります。また、特定のベーステーブルに対して FROM 句のヒントを使用して優先される可能性があります。
ジョイン列挙アルゴリズム	ジョイン列挙で使用するアルゴリズム。使用可能な値は、次のとおりです。 ● ブッシーツリー 1 ● ブッシーツリー 2 ● 最適化前によるブッシーツリー ● 削除によるブッシーツリー ● 並列ブッシーツリー ● 左が深いツリー ● ブッシーツリー 3 ● メモ化による左が深いツリー
ジョイン列挙ランタイム	ジョイン列挙フェーズ中に費やされた時間。
左が深いツリーの生成ランタイム	左が深いツリーの生成フェーズ中に費やされた時間。
ロギングランタイム	ロギングフェーズ中に費やされた時間。
論理プラン生成ランタイム	論理プラン生成フェーズ中に費やされた時間。
max_query_tasks	単一のクエリの並列実行プランで使用される可能性があるタスクの最大数。
タスクの最大数	クエリ内並列処理で使用可能なタスクの最大数。
ジョイン列挙中に使用されたメモリページ	ジョイン列挙フェーズ中に使用されたメモリページ数。
その他のランタイム	その他のフェーズ中に費やされた時間。
非最適化時間	非最適化フェーズに費やされる時間。
算出された最適化前の数	算出された最適化前フェーズで使用されたメモリページ数。
見積もられたジョインの数	見積もられたジョインの数。
列挙された論理ジョインの数	列挙された論理ジョインの数。
最適化前の数	最適化前ジョイン列挙アルゴリズムによるブッシーツリーで有効です。
メモ化テーブルでの操作	メモ化テーブルでの操作 (挿入、置換、検索が行われた)。
optimization_goal	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。
optimization_level	クエリオプティマイザがアクセスプランの検索に費やす作業量を制御します。
optimization_workload	optimization_workload 設定値 (*Mixed* または *OLAP*)。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

フィールド	説明
最適化方法	実行方式の選択に使用されたアルゴリズム。戻り値は次のとおりです。 ● バイパス (クエリ書き換え) ● バイパス (簡易クエリ書き換え) ● バイパス (ヒューリスティック) ● バイパス後、最適化 ● 最適化 ● 再利用
Optimization Time	文の最適化に要した時間。 見積済最良プラン、見積り済みプラン、最適化時間の値が 0 の場合、文は最適化されていません。データベースサーバは文をバイパスし、文の最適化を行わず実行プランを生成しています。
最適化前で使用するページ	最適化前フェーズで使用されたメモリページ数。
並列ランタイム	並列フェーズ中に費やされた時間。
パーティションランタイム	パーティションフェーズ中に費やされた時間。
物理プラン生成ランタイム	物理プラン生成フェーズ中に費やされた時間。
最適化後の時間	最適化後に費やされる時間。
最適化前の時間	最適化前に費やされる時間。
最適化前ランタイム	最適化前フェーズ中に費やされた時間。
削除されたジョイン	ローカルコストとグローバルコストに基づいて削除されたジョインの数。
ランタイムの削除	削除フェーズ中に費やされた時間。
QueryMemActiveEst	安定した状態のデータベースサーバで、クエリメモリをアクティブに使用するタスクの推定平均数。
QueryMemActiveMax	特定の時点でクエリメモリをアクティブに使用できるタスクの最大数。
QueryMemLikelyGrant	この文がすぐに実行される場合、この文に付与されるクエリメモリプールの推定ページ数。この推定値は、プラン内のメモリを大量に消費する演算子の数、データベースサーバのマルチプログラミングレベル、メモリを大量に消費する同時実行の要求数によって異なります。
QueryMemMaxUseful	この要求に使用できるクエリメモリのページ数。この数値が 0 の場合、文の実行プランにはメモリを大量に消費する演算子がなく、サーバのメモリガバナの制御対象ではありません。
QueryMemNeedsGrant	この要求の実行方式に存在し、メモリを大量に消費する 1 つ以上のクエリ実行演算子に対して、メモリガバナがメモリを付与する必要があるかどうかを示します。
QueryMemPages	すべての接続に対して、メモリを大量に消費するクエリ実行アルゴリズムで使用できるクエリメモリプール内のメモリの合計容量を表すページ数。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

フィールド	説明
user_estimates	クエリテキストの各述部で指定されたユーザ推定を尊重するか無視するかを制御します。
ジョイン列挙中に使用されたページ	ジョイン列挙中に使用されたメモリページ数。

ノード統計フィールドの説明

次に、グラフィカルプランのノード統計セクションに表示されるフィールドについて説明します。

フィールド	説明
CacheHits	この演算子によるキャッシュ読み込み要求で、バッファプールで条件が満たされ、ディスク読み込み操作が不要になった要求の合計数。
CacheRead	この演算子がデータベースファイルのページ (通常、テーブルページやインデックスページなど) を読み込もうとした合計回数。
CPUTime	このノードが表す処理アルゴリズムによって発生した CPU 時間。
CPUTimeAllThreads	すべてのスレッドの CPU による所要時間。
DiskRead	このノードの処理の結果として、ディスクから読み込まれた累積ページ数。
DiskReadTime	このノードの処理に必要なデータベースページをディスクから読み込むために要した累積時間。
DiskWrite	このノードの処理の結果として、ディスクに書き込まれた累積ページ数。
DiskWriteTime	このノードのアルゴリズムの処理に必要なデータベースページをディスクに書き込むために要した累積時間。
FirstRowRunTime	*FirstRowRunTime* 値は、このノードの中間結果の最初のローを作成するために実際に要した時間です。
Invocations	ノードが結果を計算するために呼び出された回数。親ノードに結果を返します。ほとんどのノードは 1 回だけ呼び出されます。ただし、スキャンノードの親がネストループジョインで、ノードが複数回実行される可能性がある場合、各呼び出し後に異なるローセットを返すことがあります。
PercentTotalCost	特定のノード内で結果の計算に費やす *RunTime*。文の合計 RunTime に対する割合で表します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

フィールド	説明
QueryMemMaxUseful	この特定の演算子での使用が予想されるクエリメモリの推定容量。*Actual* 統計値で報告されるクエリメモリの実際の使用量と大幅に異なる場合、クエリオプティマイザによる結果セットのサイズの推定に潜在的な問題がある可能性があります。この推定のエラーの原因は、述部選択性推定が不正確であるか存在しないことが考えられます。
RowsReturned	要求の処理の結果として親ノードに返されたローの数。*RowsReturned* は、このノードで表されるオブジェクト (派生オブジェクトの場合がある) 内のローの数と同一であることがよくありますが、必ず同じとはかぎりません。ベーステーブルスキャンを表すリーフノードを考慮してください。*RowsReturned* 値がテーブル内のローの数よりも小さい、または大きい場合があります。最終結果の計算で、親ノードがテーブルのすべてのローを要求できない場合、*RowsReturned* 値は小さくなります。GROUP BY GROUPING SETS クエリなど、親のハッシュ GROUP BY GROUPING SETS ノードが異なるグループを計算するために入力の受け渡しを複数回要求する場合、*RowsReturned* は大きくなります。 返された推定ローと返された実際の数とが大幅に食い違っている場合、オプティマイザが不正確な選択性情報に基づいて操作していることを示している可能性があります。
RunTime	実際の時間の計測値。入出力の待機、ローのロック、テーブルのロック、内部サーバの同時制御メカニズム、実際の処理実行時間が含まれます。*RunTime* の解釈は、表示される統計セクションによって異なります。[ノード統計] の場合、*RunTime* はこのノードだけを実行している間に、ノードの対応する演算子が費やした累積時間です。[ノード統計] セクションには、この統計の推定値と実際値の両方が表示されます。 テーブルスキャンまたはインデックススキャンで、ノードの *RunTime* が予想よりも大きい場合、さらに分析を進めると問題の特定に役立つ場合があります。クエリで共有リソースに対する競合が発生し、その結果ブロックした可能性があります。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をモニタできます。別の例として、ディスクのデータベースページレイアウトが最適化されていない場合や、テーブルが内部ページの断片化による影響を受けている場合があります。REORGANIZE TABLE 文を実行するとパフォーマンスを向上できることがあります。sa_table_fragmentation() システムプロシージャと sa_index_density() システムプロシージャを使用して、テーブルまたはインデックスが断片化されているかどうかを判断できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

プランに使用される一般的な統計

次の統計は実際の測定値です。

統計情報	説明
CacheHits	キャッシュ内の検索において一致したデータベースページの数を返します。
CacheRead	キャッシュ内で検索されたデータベースページの数を返します。
CacheReadTable	キャッシュから読み込まれたテーブルページの数を返します。
CacheReadIndLeaf	キャッシュから読み込まれたインデックスリーフページの数を返します。
CacheReadIndInt	キャッシュから読み込まれたインデックス内部ノードページの数を返します。
DiskRead	ディスクから読み込まれたページの数を返します。
DiskReadTable	ディスクから読み込まれたテーブルページの数を返します。
DiskReadIndLeaf	ディスクから読み込まれたインデックスリーフページの数を返します。
DiskReadIndInt	ディスクから読み込まれたインデックス内部ノードページの数を返します。
DiskWrite	ディスクに書き込まれた修正ページの数を返します。
IndAdd	インデックスに追加されたエントリの数を返します。
IndLookup	インデックス内で検索されたエントリの数を返します。
FullCompare	インデックスのハッシュ値を超えて実行された比較の回数を返します。

プランに使用される一般的な推定

統計情報	説明
EstRowCount	呼び出されるたびにノードが返すローの推定数。
AvgRowCount	各呼び出しで返される平均ロー数。これは推定値ではなく、*RowsReturned / Invocations* として計算されます。この値が *EstRowCount* とまったく異なる場合は、選択性推定が不十分な場合があります。
EstRunTime	推定実行所要時間 (*EstDiskReadTime*、*EstDiskWriteTime*、*EstCpuTime* の合計)。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

統計情報	説明
AvgRunTime	平均実行所要時間 (測定値)。
EstDiskReads	ディスクからの読み込み操作の推定回数。
AvgDiskReads	ディスクからの読み込み操作の平均回数 (測定値)。
EstDiskWrites	ディスクへの書き込み操作の推定回数。
AvgDiskWrites	ディスクへの書き込み操作の平均回数 (測定値)。
EstDiskReadTime	ディスクからローを読み込むときの推定所要時間。
EstDiskWriteTime	ディスクにローを書き込むときの推定所要時間。
EstCpuTime	プロセッサの推定実行所要時間。

SELECT、INSERT、UPDATE、DELETE に関連するプランの項目

項目	説明
最適化目標	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。
最適化負荷	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。
ANSI 更新制約	更新が許される範囲を制御します (オプションは、Off、Cursors、Strict)。
最適化レベル	予約済み。
Select リスト	クエリによって選択される式のリスト。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

項目	説明
Materialized views	オプティマイザによって検討されるマテリアライズドビューのリスト。リスト内の各エントリは `view-name [ view-matching-outcome ] [ table-list ]` という形式の組です。ここで `view-matching-outcome` はマテリアライズドビューの使用法を示します。この値が COSTED の場合、ビューは列挙時に使用されています。`table-list` は、このビューで置き換えられる可能性のあったクエリテーブルのリストです。 `view-matching-outcome` の値は、次のとおりです。 ● ベーステーブルが一致しません ● 権限が一致しません ● 述部が一致しません ● Select リストが一致しません ● 見積り済みです ● 失効が一致しません ● スナップショットの失効が一致しません ● オプティマイザでは使用できません ● オプティマイザでは内部で使用できません ● 定義を構築できません ● アクセスできません ● 無効になっています ● オプションが一致しません ● ビューマッチングがしきい値に到達しました ● ビューは使用されています

ロックに関連するプランの項目

項目	説明
ロックテーブル	すべてのロックテーブルとその独立性レベルのリスト。

スキャンに関連するプランの項目

項目	説明
テーブル名	テーブルの実際の名前。
相関名	テーブルのエイリアス。
予測ロー数	テーブルの推定ロー数。
予測ページ数	テーブルの推定ページ数。
予測ローサイズ	テーブルの推定ローサイズ。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

項目	説明
ページマップ	複数ページの読み込みにページマップが使用される場合は YES。

インデックススキャンに関連するプランの項目

項目	説明
選択性	範囲バウンドと一致する推定ロー数。
インデックス名	インデックスの名前。
キータイプ	PRIMARY KEY、FOREIGN KEY、CONSTRAINT (一意性制約)、UNIQUE (ユニークインデックス) のいずれか。インデックスがユニークでないセカンダリインデックスの場合、キータイプは表示されません。
深さ	インデックスの高さ。
予測リーフページ数	リーフページの推定数。
連続変換	インデックスがどれだけクラスタ化されているかを示す、各物理インデックスの統計情報。
ランダム変換	インデックスがどれだけクラスタ化されているかを示す、各物理インデックスの統計情報。
キー値	インデックス内のユニークなエントリの数。
カーディナリティ	推定ロー数と異なる場合の、インデックスのカーディナリティ。バージョン 6.0.0 以前の SQL Anywhere データベースにのみ適用されます。
方向	FORWARD または BACKWARD。
範囲バウンド	範囲バウンドは、リスト (col_name=value) または col_name IN [low, high] として表示されます。
プライマリキーテーブル	外部キーインデックススキャンのプライマリキーテーブル名。
プライマリキーテーブルの予測ロー数	外部キーインデックススキャンのプライマリキーテーブル内のロー数。
プライマリキーカラム	外部キーインデックススキャンのプライマリキーカラム名。

ジョイン、フィルタ、事前フィルタに関連するプランの項目

項目	説明
ハッシュテーブルバケット数	ハッシュテーブルに使用されるバケット数。
キー値	重複しないハッシュキー値の推定数。
述部	このノードで評価される探索条件、選択性推定、測定値。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ハッシュフィルタに関連するプランの項目

項目	説明
構築値	入力内の重複しない値の推定数。
測定値	述部をチェックする場合の、入力内の重複しない値の推定数。
ビット数	ハッシュマップを構築するために選択されたビット数。
ページ数	ハッシュマップを格納するために必要なページ数。

UNION に関連するプランの項目

項目	説明
Union リスト	UNION 文が対象とするカラム。

GROUP BY に関連するプランの項目

項目	説明
集合関数	すべての集合関数。
Group-by リスト	GROUP BY 句に指定されているすべてのカラム。
ハッシュテーブルバケット数	ハッシュテーブルに使用されるバケット数。
キー値	重複しないハッシュキー値の推定数。

DISTINCT に関連するプランの項目

項目	説明
Distinct リスト	DISTINCT 句に指定されているすべてのカラム。
ハッシュテーブルバケット数	ハッシュテーブルに使用されるバケット数。
キー値	重複しないハッシュキー値の推定数。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

IN リストに関連するプランの項目

項目	説明
In リスト	指定したセットのすべての式。
式 *SQL*	リストと比較される式。

SORT に関連するプランの項目

項目	説明
Order-by	ソート基準となるすべての式のリスト。

ロー制限に関連するプランの項目

項目	説明
ロー制限数	FIRST または TOP `n` で指定された、返されるローの最大数。

WINDOW 演算子に関連するプランの項目

項目	説明
ウィンドウフレーム	OVER 句の処理方法に関する情報。
ローの削除方式	フレームが UNBOUNDED PRECEDING として定義されていない場合に、フレームからローを削除する際に使用する方法。反転集合関数の 1 つとして、SUM や COUNT などの反転可能な関数に使用される効率的な方法。または、再スキャンバッファとして、MIN または MAX など、すべての入力を再検討する必要がある関数に使用されるコストの高い方法。
Partition By	OVER 句の PARTITION BY に使用される式のリスト。この項目が空の場合は、除外されます。
Order-by	OVER 句の ORDER BY に使用される式のリスト。この項目が空の場合は、除外されます。
Window 関数	WINDOW 演算子によって計算される Window 関数のリスト。

関連情報

Query Processing Based on SQL Anywhere 12.0.1 Architecture

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプティマイザの仕組み [203 ページ]
グラフィカルプランの選択性情報 [221 ページ]
マテリアライズドビューの制限 [69 ページ]

1.3.1.13 高度: クエリ実行時の並列処理

クエリ実行に対して、クエリ間とクエリ内の 2 種類の並列処理があります。

クエリ間並列処理とは、異なる要求を別個の CPU 上で同時に実行することです。各要求 (タスク) は単一のスレッド、単一のプロセッサで実行されます。

クエリ内並列処理とは、単一の要求を複数の CPU で同時処理することです。クエリが分割されて各部がマルチプロセッサのハードウェアで並列処理されます。これらの各部は交換アルゴリズムで処理されます。

クエリ内並列処理では、同時に実行されるクエリ数が使用可能なプロセッサ数より少ないことが多い場合に、負荷が分散されます。並列処理の程度は、max_query_tasks オプションを設定して制御します。

オプティマイザは、並列処理の追加コスト (ローの追加コピー、作業量の調整のための追加コスト) を推定し、パフォーマンスの向上が期待できる場合のみ並列プランを選択します。

クエリ内並列処理は、priority オプションが background に設定されている接続には使用されません。

現在要求を処理しているサーバスレッド数 (ActiveReq サーバプロパティ) が、データベースサーバが使用ライセンスを持つコンピュータの CPU コア数を最近超えた場合は、クエリ内並列処理は使用されません。正確な時間はサーバによって決められ、通常は数秒です。

並列実行

クエリで並列実行が利用されるかどうかは、次の要因によって決まります。

- 最適化時にシステムで使用可能なリソース (メモリ、キャッシュ内のデータなど)
- コンピュータの論理プロセッサ数
- データベースの格納に使用されるディスクデバイス数、プロセッサとコンピュータの I/O アーキテクチャの速度に対する相対速度
- 要求に必要な特定の代数演算子。SQL Anywhere では、次の 5 つの代数演算子がサポートされています。これらの演算子は並列実行できます。
    - 並列逐次スキャン (テーブルスキャン)
    - 並列インデックススキャン
    - 並列ハッシュジョイン、ハッシュセミジョインと非セミジョインの並列バージョン
    - 並列ネストループジョイン、ネストループセミジョインとネストループ非セミジョインの並列バージョン
    - 並列ハッシュフィルタ
    - 並列ハッシュ Group By

サポートされていない演算子を使用しているクエリでも並列実行できますが、プランで、サポートされている演算子がサポートされていない演算子の下にある必要があります (Interactive SQL での表示)。サポートされていない演算子のほとんどが上の方にあるクエリは、並列処理される確率が高くなります。たとえば、ソート演算子は並列処理できませんが、最も外側のブロ

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ックで ORDER BY を使用するクエリは、ソートをプランの一番上に置き、すべての並列演算子をその下に置くことで並列処理できます。これに対して、派生テーブルで TOP `n` と ORDER BY を使用するクエリは、ソートがプランの一番上にないので、並列処理が使用される確率が低くなります。

データベースサーバでは、DB 領域がデフォルトで単一プラッターのディスクサブシステムにあると想定されます。このような環境では、クエリの並列実行を行う効果があっても、テーブルのデータが完全にキャッシュ内に収まっていない限り、オプティマイザの単一デバイス用の I/O コストモデルによって、オプティマイザで並列テーブルまたはインデックススキャンを選択するのが困難になります。ただし、ALTER DATABASE CALIBRATE PARALLEL READ 文を使用してディスクのサブシステムを調整することによって、オプティマイザで並列実行の効果をより正確に計算できます。ディスクのサブシステムに複数のプラッターが搭載されているときには、オプティマイザでは並行実行を伴う実行プランが選択される傾向があります。

クエリ内並列処理がアクセスプランに使用される場合、各サブツリーの並列処理の結果をマージ (UNION) する交換演算子がプランに含まれます。交換演算子の下位のサブツリー数が、並列処理の程度です。各サブツリー、またはアクセスプランのコンポーネントは、データベースサーバのタスクです。データベースサーバのカーネルでは、実行スレッド (ファイバー) が使用可能かどうかに応じて、個々の SQL 要求と同じようにこれらのタスクがスケジュールされます。このアーキテクチャでは、すべてのアクセスプランの並列処理の大部分が自動的に調整されます。並列実行タスクの処理はサーバカーネルの許可に従ってスレッド (ファイバー) にスケジュールされ、プランのコンポーネントが均等に実行されます。

このセクションの内容:

[クエリの並列処理 [247 ページ]]
クエリは、クエリが処理するロー数が、返されるロー数よりも多い場合に並列処理が使用される確率が高くなります。

関連情報

高度: クエリ実行プラン [210 ページ]
Query Processing Based on SQL Anywhere 12.0.1 Architecture

1.3.1.13.1 クエリの並列処理

クエリは、クエリが処理するロー数が、返されるロー数よりも多い場合に並列処理が使用される確率が高くなります。

この場合、処理されるロー数には、スキャンされるすべてのローのサイズとすべての中間結果のサイズが含まれます。インデックスを使用してテーブルのほとんどがスキップされるので、スキャンされないローは含まれません。理想的なケースは、大きなテーブルに対する単一ローの GROUP BY であり、この場合、多数のローがスキャンされ、1 つのローだけが返されます。グループのサイズが大きい場合は、複数グループのクエリも並列処理の確率が高くなります。多数のローを削除する述部またはジョイン条件も高い確率で並列処理されます。

最適化または実行のときにクエリに並列処理が使用されない場合のリストを次に示します。

- サーバのコンピュータに複数のプロセッサがありません。
- サーバのコンピュータに、複数のプロセッサを使用するためのライセンスがありません。これは、NumLogicalProcessorsUsed サーバプロパティで確認できます。ただし、ハイパースレッドのプロセッサはクエリ内並列処理対象として数えられないので、コンピュータでハイパースレッドを使用している場合は、NumLogicalProcessorsUsed の値を 2 で割ります。
- max_query_tasks オプションが 1 に設定されています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- priority オプションが background に設定されています。
- クエリを含む文が SELECT 文ではありません。
- 最近、ActiveReq の値が NumLogicalProcessorsUsed の値以上になったことがあります (コンピュータでハイパースレッドを使用している場合はプロセッサ数を 2 で割ります)。
- 使用可能なタスク数が不十分です。

1.3.2 全文検索

テーブルに対して全文検索を実行できます。

このセクションの内容:

全文検索とは [249 ページ]
　全文検索は、データベースを検索するためのより高度な方法です。

全文検索のタイプ [260 ページ]
　全文検索を使用して、単語、フレーズ (単語のシーケンス)、またはプレフィクスを検索できます。

全文検索結果のスコア [276 ページ]
　クエリの FROM 句に CONTAINS 句を含めると、それぞれの一致にスコアが関連付けられます。

テキスト設定オブジェクトの概念と参照 [279 ページ]
　テキスト設定オブジェクトは、テキストインデックスの作成時または再表示時に格納される単語を制御し、全文クエリの解釈方法を制御します。

テキストインデックスの概念と参照 [292 ページ]
　テキストインデックスには、インデックス付けされたカラム内の単語の位置情報が格納されます。

チュートリアル: GENERIC テキストインデックスに対する全文検索の実行 [295 ページ]
　GENERIC 単語区切りを使用するテキストインデックスに対して、全文検索を実行します。

チュートリアル: あいまい全文検索の実行 [306 ページ]
　NGRAM 単語区切りを使用するテキストインデックスに対して、あいまい全文検索を実行します。

チュートリアル: NGRAM テキストインデックスに対する非あいまい全文検索の実行 [311 ページ]
　NGRAM 単語区切りを使用するテキストインデックスに対して、非あいまい全文検索を実行します。中国語、日本語、韓国語データの全文検索を作成する際もこの手順で実行できます。

高度: 全文検索での単語の削除 [325 ページ]
　テキストインデックスは、テキストインデックスの作成に使用されるテキスト設定オブジェクトに定義された設定に従って、構築されます。

高度: 外部単語区切りライブラリとプレフィルタライブラリ [326 ページ]
　独自の外部単語区切りライブラリとプレフィルタライブラリを作成して使用できます。

高度: 外部全文ライブラリの API [335 ページ]
　テキストインデックスで事前フィルタまたは単語区切りの外部ライブラリを作成し使用するには、次の手順を実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.1 全文検索とは

全文検索は、データベースを検索するためのより高度な方法です。

全文検索を行うと、ローをスキャンする必要や単語が格納されているカラムを知る必要がなく、テーブル内の特定の単語のすべてのインスタンスを簡単に検索できます。全文検索は、テキストインデックスを使用することで機能します。テキストインデックスには、テキストインデックスを作成した、カラム内のすべての単語の位置情報が格納されます。特定の値を含むローを検索する場合、テキストインデックスを使用した方が、通常のインデックスを使用するよりも処理が速い可能性があります。

SQL Anywhere の全文検索機能は単語単位であり、パターン単位でないという点で、LIKE、REGEXP、SIMILAR TO などの述部を使用した検索とは異なります。

全文検索の文字列の比較では、データベースのすべての通常の照合設定が使われます。たとえば、データベースが大文字と小文字を区別しないように設定されている場合、全文検索でも大文字と小文字が区別されません。

別途指定されないかぎり、全文検索では、SQL Anywhere でサポートされているすべての国際化機能が使われます。

全文検索を実行する 2 つの方法

全文クエリを実行するには、SELECT 文の FROM 句で CONTAINS 句を使用するか、または WHERE 句で CONTAINS 検索条件 (述部) を使用します。どちらも同じローが返されますが、FROM 句で CONTAINS 句を使用した場合は、一致するローのスコアも返されます。

次の例は、クエリでの CONTAINS 句と CONTAINS 探索条件の使用方法を示します。これらの例では、サンプルデータベースで提供されている MarketingInformation.Description テキストインデックスが使用されています。

```
SELECT *
   FROM MarketingInformation CONTAINS ( Description, 'cotton' );
```

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( Description, 'cotton' );
```

全文検索を使用する前の考慮事項

通常のインデックスではなく全文インデックスを使用するかどうかを決定するときには、次の考慮事項があります。

- CONTAINS 句または CONTAINS 探索条件ではエイリアスを使用できません。
- クエリで重複する相関名を使用する場合、CONTAINS (FROM CONTAINS()) は、相関名の最初のインスタンスでのみサポートされます。たとえば、2 番目の CONTAINS 述部に A が含まれているため、次の構文ではエラーが返されます。

```
SELECT *
FROM CONTAINS(A contains-query-string) JOIN B ON A.x = B.x,
     CONTAINS(A contains-query-string) JOIN C ON A.y = C.y;
```

外部の単語区切りライブラリと事前フィルタライブラリを使用する場合は、次の追加の考慮事項があります。

クエリと更新

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ライブラリを使用して作成されたテキストインデックスの更新、クエリ、または変更を必要とするすべての操作に対して、外部ライブラリが引き続き使用可能になっている必要があります。

アンロードと再ロード

全文インデックスに関連付けられているデータのアンロード時と再ロード時に、外部ライブラリが使用可能になっている必要があります。

データベースのリカバリ

データベースをリカバリするには、外部ライブラリが使用可能である必要があります。これは、トランザクションログ内に、最終チェックポイント以降に外部ライブラリが関与した操作が存在する場合は、データベースをリカバリできないためです。

このセクションの内容:

テキスト設定オブジェクトの作成 (SQL Central の場合) [251 ページ]

テキスト設定オブジェクト作成ウィザードを使用して、SQL Central でテキスト設定オブジェクトを作成します。

テキスト設定オブジェクトの変更 [252 ページ]

単語区切りタイプ、ストップリスト、オプション設定などのテキスト設定オブジェクトのプロパティを変更します。

データベースのテキスト設定オブジェクトの表示 [253 ページ]

SQL Central でテキスト設定オブジェクトの設定や他のプロパティを表示します。

テキストインデックスの作成 [253 ページ]

カラムのテキストインデックスを作成します。

テキストインデックスの再表示 [255 ページ]

テキストインデックスを再表示し、テキストインデックスのデータを更新します。テキストインデックスを再表示すると、基本となるテーブルで実施されたデータの変更を反映します。

テキストインデックスの変更 [256 ページ]

テキストインデックスの再表示タイプ、名前、Content 特性を変更します。

テキストインデックスの単語と設定の表示 (SQL Central の場合) [258 ページ]

SQL Central でテキストインデックスの単語と設定を表示します。

テキストインデックスの単語と設定の表示 (SQL の場合) [259 ページ]

Interactive SQL でテキストインデックスの単語と設定を表示します。

関連情報

高度: 外部単語区切りライブラリとプレフィルタライブラリ [326 ページ]

テキストインデックスの概念と参照 [292 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.1.1 テキスト設定オブジェクトの作成 (SQL Central の場合)

テキスト設定オブジェクト作成ウィザードを使用して、SQL Central でテキスト設定オブジェクトを作成します。

前提条件

所有するオブジェクトのテキスト設定を作成するには、CREATE TEXT CONFIGURATION システム権限が必要です。

他のユーザが所有するオブジェクトのテキスト設定を作成するには、CREATE ANY TEXT CONFIGURATION または CREATE ANY OBJECT システム権限が必要です。

コンテキスト

テキスト設定オブジェクトは、テキストインデックスを構築したり更新するときに使用します。

手順

1. SQL Central で、*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. テキスト設定オブジェクトを右クリックし、新規 > テキスト設定オブジェクト をクリックします。
3. テキスト設定オブジェクト作成ウィザードの指示に従います。
4. テキスト設定オブジェクトウィンドウ枠をクリックします。

結果

テキスト設定オブジェクトウィンドウ枠に、新しいテキスト設定オブジェクトが表示されます。

関連情報

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]
テキスト設定オブジェクトの例 [286 ページ]
データベースのテキスト設定オブジェクトの表示 [253 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.1.2 テキスト設定オブジェクトの変更

単語区切りタイプ、ストップリスト、オプション設定などのテキスト設定オブジェクトのプロパティを変更します。

前提条件

CREATE EXTERNAL REFERENCE システム権限が必要です。

コンテキスト

テキストインデックスは、作成時に使用したテキスト設定オブジェクトに依存しています。このため、依存するテキストインデックスをトランケートまたは削除してください。また、テキスト設定オブジェクトの保存された日付または時間形式のオプションを変更する場合は、希望する設定値を設定したオプションを指定してデータベースに接続してください。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、**テキスト設定オブジェクト**をクリックします。
3. テキスト設定オブジェクトを右クリックし、**プロパティ**をクリックします。
4. テキスト設定オブジェクトのプロパティを編集し、*[OK]*をクリックします。

結果

テキスト設定オブジェクトが変更されます。

関連情報

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]
テキスト設定オブジェクトの例 [286 ページ]
データベースのテキスト設定オブジェクトの表示 [253 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.1.3 データベースのテキスト設定オブジェクトの表示

SQL Central でテキスト設定オブジェクトの設定や他のプロパティを表示します。

前提条件

テキスト設定オブジェクトの所有者であるか、ALTER ANY TEXT CONFIGURATION または ALTER ANY OBJECT システム権限を持っている必要があります。

- ALTER ANY TEXT CONFIGURATION
- ALTER ANY OBJECT

手順

1. SQL Central で、*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. 左ウィンドウ枠で、**テキスト設定オブジェクト**をクリックします。
3. テキスト設定オブジェクトを右クリックし、**プロパティ**をクリックします。

結果

テキスト設定オブジェクトの設定が、表示されます。

関連情報

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]

1.3.2.1.4 テキストインデックスの作成

カラムのテキストインデックスを作成します。

前提条件

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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限
- そのテーブルに対する REFERENCES 権限に加え、COMMENT ANY OBJECT システム権限、ALTER ANY INDEX システム権限、または 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

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 句で開かれているカーソルがある場合、テキストインデックスは作成できません。

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。無効になっているマテリアライズドビューのテキストインデックスは作成できません。

コンテキスト

テキストインデックスはディスク領域を消費し、再表示を必要とします。作成する場合は、クエリをサポートするために必要となるカラムのみを対象にしてください。

VARCHAR または NVARCHAR タイプ以外のカラムは、インデックスの作成時に文字列に変換されます。

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ます。

手順

1. SQL Central で、*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. テキストインデックスタブをクリックします。
3. ファイル > 新規 > テキストインデックス をクリックします。
4. インデックス作成ウィザードの指示に従います。

   テキストインデックスタブに新しいテキストインデックスが表示されます。また、テキストインデックスフォルダにも表示されます。

結果

テキストインデックスが作成されます。即時再表示のテキストインデックスを作成した場合は、データが自動的に設定されます。その他の再表示タイプの場合は、テキストインデックスを手動で再表示する必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

テキストインデックスの再表示タイプ [293 ページ]
テキストインデックスの再表示 [255 ページ]

1.3.2.1.5 テキストインデックスの再表示

テキストインデックスを再表示し、テキストインデックスのデータを更新します。テキストインデックスを再表示すると、基本となるテーブルで実施されたデータの変更を反映します。

前提条件

テキストインデックスを再表示するには、基本となる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。

- CREATE ANY INDEX システム権限
- CREATE ANY OBJECT システム権限
- ALTER ANY INDEX システム権限
- ALTER ANY OBJECT システム権限
- そのテーブルに対する REFERENCES 権限

再表示できるテキストインデックスは、AUTO REFRESH および MANUAL REFRESH として定義されているテキストインデックスだけです。IMMEDIATE として定義されているテキストインデックスは再表示できません。

コンテキスト

マテリアライズドビューのテキストインデックスは、ビューが更新されたり再表示されると必ず再表示されます。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、**テキストインデックス**をクリックします。
3. テキストインデックスを右クリックし、*[データの再表示]* をクリックします。
4. 再表示の独立性レベルを選択し、*[OK]*をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

テキストインデックスが再表示されます。

関連情報

テキストインデックスの再表示タイプ [293 ページ]

1.3.2.1.6 テキストインデックスの変更

テキストインデックスの再表示タイプ、名前、Content 特性を変更します。

再表示タイプ

の再表示タイプを AUTO REFRESH から MANUAL REFRESH に変更したり、その逆に変更することができます。再表示タイプを変更するには、ALTER TEXT INDEX 文の REFRESH 句を使用します。

テキストインデックスを IMMEDIATE REFRESH に変更したり、IMMEDIATE REFRESH から変更することはできません。この変更を実行するには、テキストインデックスを削除して、再作成する必要があります。

名前

テキストインデックスの名前を変更するには、ALTER TEXT INDEX 文の RENAME 句を使用します。

内容

カラムリストを除き、インデックス付けの対象を制御する設定は、テキスト設定オブジェクトに格納されます。インデックス付けの対象を変更する場合は、テキストインデックスから参照されるテキスト設定オブジェクトを変更します。依存するテキストインデックスをトランケートしてからテキスト設定オブジェクトを変更し、その後テキストインデックスを再表示する必要があります。即時再表示のテキストインデックスの場合、テキストインデックスを削除し、テキスト設定オブジェクトを変更した後にテキストインデックスを再作成する必要があります。

別のテキスト設定オブジェクトを参照するように、テキストインデックスを変更することはできません。テキストインデックスで別のテキスト設定オブジェクトを参照させるには、テキストインデックスを削除して、新しいテキスト設定オブジェクトを指定して再作成します。

このセクションの内容:

テキストインデックスの変更 [257 ページ]
テキストインデックスの名前を変更したり、その再表示タイプを変更できます。

関連情報

テキストインデックスの再表示タイプ [293 ページ]
テキスト設定オブジェクトの概念と参照 [279 ページ]
テキストインデックスの概念と参照 [292 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.1.6.1 テキストインデックスの変更

テキストインデックスの名前を変更したり、その再表示タイプを変更できます。

前提条件

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。

- ALTER ANY INDEX システム権限
- ALTER ANY OBJECT システム権限
- そのテーブルに対する REFERENCES 権限に加え、COMMENT ANY OBJECT システム権限、CREATE ANY INDEX システム権限、または 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

別のテキスト設定オブジェクトを参照するように、テキストインデックスを変更することはできません。テキストインデックスで別のテキスト設定オブジェクトを参照させるには、テキストインデックスを削除して、新しいテキスト設定オブジェクトを指定して再作成します。

テキストインデックスを IMMEDIATE REFRESH に変更したり、IMMEDIATE REFRESH から変更することはできません。この変更を実行するには、テキストインデックスを削除して、再作成する必要があります。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テキストインデックスをクリックします。
3. テキストインデックスを右クリックし、プロパティをクリックします。
4. テキストインデックスのプロパティを編集します。

   テキストインデックスの名前は *[一般]* タブで変更できます。
5. *OK* をクリックします。

結果

テキストインデックスが変更されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

テキストインデックスの再表示タイプ [293 ページ]
テキスト設定オブジェクトの概念と参照 [279 ページ]
テキストインデックスの概念と参照 [292 ページ]

1.3.2.1.7 テキストインデックスの単語と設定の表示 (SQL Central の場合)

SQL Central でテキストインデックスの単語と設定を表示します。

前提条件

テキストインデックスの完全な情報を表示するには、テーブルまたはマテリアライズドビューの所有者であるか、次のいずれかのシステム権限を持っている必要があります。

- CREATE ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY INDEX
- ALTER ANY OBJECT
- DROP ANY INDEX
- DROP ANY OBJECT
- MANAGE ANY STATISTICS

ボキャブラリタブの情報を表示するには、次のいずれかの権限も必要です。

- テキストインデックスが作成されたテーブルまたはマテリアライズドビューに対する SELECT 権限
- SELECT ANY TABLE システム権限

テキストインデックスを初期化する必要があります。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テキストインデックスをクリックします。
3. テキストインデックス内の単語を表示するには、左ウィンドウ枠でテキストインデックスをダブルクリックし、右ウィンドウ枠でボキャブラリタブをクリックします。
4. インデックスが参照するテキスト設定オブジェクトや再表示タイプなど、テキストインデックスの設定を表示するには、テキストインデックスを右クリックし、*[プロパティ]* をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

テキストインデックスの単語と設定が表示されます。

1.3.2.1.8 テキストインデックスの単語と設定の表示 (SQL の場合)

Interactive SQL でテキストインデックスの単語と設定を表示します。

前提条件

テキストインデックスに関する設定や統計の情報を表示するには、次のいずれかのシステム権限が必要です。

- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY OBJECT
- DROP ANY OBJECT

テキストインデックスの単語を表示するには、次のいずれかの権限も必要です。

- テーブルまたはマテリアライズドビューに対する SELECT 権限
- SELECT ANY TABLE システム権限

テキストインデックスを初期化する必要があります。

手順

1. データベースに接続します。
2. sa_text_index_stats システムプロシージャを実行して、テキストインデックスに関する統計情報を表示します。

```
CALL sa_text_index_stats( );
```

3. sa_text_index_vocab システムプロシージャを実行して、テキストインデックスの単語を表示します。

```
CALL sa_text_index_vocab( );
```

結果

テキストインデックスの統計情報や単語が表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

テキストインデックスが作成されると、現在のデータベースオプションはテキストインデックスとともに格納されます。テキストインデックスの作成中に使用されたオプション設定を取得するには、次の文を実行します。

```
SELECT b.object_id, b.table_name, a.option_id, c.option_name, a.option_value
FROM SYSMVOPTION a, SYSTAB b, SYSMVOPTIONNAME c
WHERE a.view_object_id=b.object_id
AND b.table_type=5;
```

SYSTAB ビュー内で table_type が 5 のものは、テキストインデックスです。

1.3.2.2 全文検索のタイプ

全文検索を使用して、単語、フレーズ (単語のシーケンス)、またはプレフィクスを検索できます。

複数の単語、フレーズ、またはプレフィクスをブール式に組み合わせることもできます。または、その式が近接検索で互いに近くに出現する必要があります。

全文検索を実行するには、SELECT 文の WHERE 句または FROM 句で CONTAINS 句を使用します。さらに、IF 探索条件の一部として全文検索を実行することもできます (`SELECT IF CONTAINS...` など)。

このセクションの内容:

全文検索単語とフレーズの検索 [260 ページ]
: 単語のリストの全文検索を実行する場合は、フレーズ内の単語でないかぎり、単語の順序は重要ではありません。

全文プレフィクス検索 [266 ページ]
: 全文検索では、単語の先頭部分を検索できる機能があります。これはプレフィクス検索と呼ばれます。

全文近接検索 [269 ページ]
: 全文検索では、単一のカラム内で互いに近接する単語を検索できる機能があります。これは近接検索と呼ばれます。

全文検索ブール検索 [272 ページ]
: 全文検索を実行するときに、複数の単語をブール演算子 (AND、OR、AND NOT など) で区切って指定できます。

全文あいまい検索 [275 ページ]
: あいまい検索を使用して、単語の綴り間違いや変形を検索できます。

ビューに対する全文検索 [275 ページ]
: ビューまたは派生テーブルで全文検索を使用するには、全文検索を実行したいベーステーブルのカラムにテキストインデックスを構築する必要があります。

1.3.2.2.1 全文検索単語とフレーズの検索

単語のリストの全文検索を実行する場合は、フレーズ内の単語でないかぎり、単語の順序は重要ではありません。

単語がフレーズ内にある場合、データベースサーバは、出現する順序と相対位置が指定されたものと厳密に一致する単語を検索します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

単語検索やフレーズ検索を実行する際、単語長の設定を超過するため、またはストップリストに含まれているために単語がクエリから削除された場合、予期しない数のローが返される可能性があります。これは、単語をクエリから削除することは探索条件を変更することと同じであるためです。たとえば、`'"grown cotton"'` というフレーズを検索し、grown がストップリストに入っている場合、cotton を含むインデックス付けされたローがすべて返されます。

CONTAINS 句の文法でキーワードと見なされる単語は、フレーズに含まれている場合のみ検索できます。

単語検索

サンプルデータベースでは、MarketingInformation テーブルの Description カラムに MarketingTextIndex というテキストインデックスが作成されています。次の文は、MarketingInformation テーブルの Description カラムを問い合わせて、Description カラムの値に *cotton* という単語が含まれるローを返します。

```
SELECT ID, Description
   FROM MarketingInformation
   WHERE CONTAINS ( Description, 'cotton' );
```

ID	Description
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation.cotton Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>

次の例は、MarketingInformation テーブルを問い合わせて、各ローについて、Description カラムの値に `cotton` という単語が含まれるかどうかを示す 1 つの値を返します。

```
SELECT ID, IF CONTAINS ( Description, 'cotton' )
      THEN 1
      ELSE 0
      ENDIF AS Results
   FROM MarketingInformation;
```

ID	Results
901	0
902	0
903	0
904	0
905	0
906	1
907	0
908	1
909	1
910	1

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の例は、MarketingInformation テーブルを問い合わせて、Description カラムに `cotton` という単語が含まれる項目を検索し、一致ごとのスコアを表示します。

```
SELECT ID, ct.score, Description
   FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'cotton' ) as ct
   ORDER BY ct.score DESC;
```

ID	score	Description
908	0.9461597363521859	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>
910	0.9244136988525732	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	score	Description
906	0.9134171046194403	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>
909	0.8856420222728282	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>

フレーズ検索

フレーズの全文検索を実行する場合、フレーズを二重引用符で囲みます。指定の順序と相対位置の単語が含まれる場合、カラムは一致します。

単語がフレーズ内に含まれている場合を除き、AND や FUZZY などの CONTAINS キーワードを、検索する単語として指定することはできません (単一単語のフレーズは検索可能)。たとえば、NOT は CONTAINS キーワードですが、次の文は実行できます。

```
SELECT * FROM table-name CONTAINS ( Remarks, '"NOT"' );
```

フレーズの中で使用される特殊文字は、特殊文字として解釈されません (アスタリスクを除く)。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

フレーズは、近接検索の引数としては使用できません。

次の文は、MarketingInformation テーブルの Description カラムを問い合わせて、`"grown cotton"` というフレーズを検索し、一致ごとのスコアを表示します。

```
SELECT ID, ct.score, Description
   FROM MarketingInformation CONTAINS ( MarketingInformation.Description, '"grown
cotton"' ) as ct
   ORDER BY ct.score DESC;
```

ID	score	Description
908	1.6619019465461564	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>
906	1.6043904700786786	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>

関連情報

全文検索結果のスコア [276 ページ]
全文プレフィクス検索 [266 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.2.2 全文プレフィクス検索

全文検索では、単語の先頭部分を検索できる機能があります。これはプレフィクス検索と呼ばれます。

プレフィクス検索を実行するには、検索するプレフィクスを指定した後に、アスタリスクを付けます。これはプレフィクス単語と呼ばれます。

CONTAINS 句のキーワードは、フレーズに含まれていないかぎり、プレフィクス検索では使用できません。

クエリ文字列に複数のプレフィクス単語 (フレーズ内のプレフィクス単語も含む) を指定することもできます (たとえば、`'"shi* fab"'` など)。

次の例は、MarketingInformation テーブルを問い合わせて、プレフィクス shi で始まる項目を検索します。

```
SELECT ID, ct.score, Description
   FROM MarketingInformation CONTAINS ( MarketingInformation.Description, 'shi*' )
AS ct
   ORDER BY ct.score DESC;
```

ID	score	Description
906	2.295363835537917	`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>`

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	score	Description
901	1.6883275743936228	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee **Shi**rt</title></head><body lang=EN-US> We've improved the design of this perennial favorite. A sleek and technical **shi**rt built for the trail, track, or sidewalk. UPF rating of 50+. </body></html>
903	1.6336529491832605	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee **Shi**rt</title></head><body lang=EN-US> A sporty, casual **shi**rt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin. </body></html>

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	score	Description
902	1.6181703448678983	`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee **Shi**rt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical **shi**rt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin. </body></html>`

テキストインデックスで shield という単語が出現する頻度は shirt よりも低いため、ID 906 のスコアが最も高くなります。

GENERIC テキストインデックスに対するプレフィクス検索

GENERIC テキストインデックスでは、プレフィクス検索は次のように動作します。

- プレフィクス単語が MAXIMUM TERM LENGTH よりも長い場合、MAXIMUM TERM LENGTH よりも長い単語はテキストインデックスに存在できないため、そのプレフィクス単語はクエリ文字列から削除されます。したがって、MAXIMUM TERM LENGTH が 3 のテキストインデックスで `'red appl*'` を検索することは、`'red'` を検索することと同義になります。
- プレフィクス単語が MINIMUM TERM LENGTH よりも短く、フレーズ検索の一部ではない場合、プレフィクス検索は通常通り処理されます。したがって、MINIMUM TERM LENGTH が 5 の GENERIC テキストインデックスで、`'macintosh a*'` を検索すると、macintosh、および a で始まる 5 文字以上の任意の単語を含むインデックス付けされたローが返されます。
- プレフィクス単語が MINIMUM TERM LENGTH よりも短いものの、フレーズ検索の一部になっている場合、プレフィクス単語はクエリから削除されます。したがって、MINIMUM TERM LENGTH が 5 の GENERIC テキストインデックスで、`'"macintosh appl* turnover"'` を検索することは、「macintosh 任意の単語 turnover」を検索することと同義になります。`"macintosh turnover"` を含むローは検索されません。macintosh と turnover の間に単語が必要になります。

NGRAM テキストインデックスに対するプレフィクス検索

NGRAM テキストインデックスの場合、NGRAM テキストインデックスには N-gram しか含まれておらず、単語の先頭に関する情報は含まれていないため、プレフィクス検索を実行すると予期しない結果が返される可能性があります。また、クエリ単語

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

は N-gram に分割され、検索はクエリ単語ではなく N-gram を使用して実行されます。そのため、次のような動作に注意する必要があります。

- プレフィクス単語が N-gram の長さ (MAXIMUM TERM LENGTH) よりも短い場合、クエリはそのプレフィクス単語で始まる N-gram を含むすべてのインデックス付けされたローを返します。たとえば、3-gram のテキストインデックスの場合、'ea*' を検索すると、ea で始まる N-gram を含むすべてのインデックス付けされたローが返されます。したがって、weather と fear という単語がインデックス付けされている場合、これらの N-gram にはそれぞれ eat と ear が含まれているため、ローは一致すると見なされます。
- プレフィクス単語が N-gram の長さよりも長く、フレーズの一部ではなく、近接検索の引数ではない場合、プレフィクス単語は N-gram フレーズに変換され、アスタリスクは削除されます。たとえば、3-gram のテキストインデックスの場合、'purple blac*' を検索することは、'"pur urp rpl ple" AND "bla lac"' を検索することと同義になります。
- フレーズの場合は、次のような動作も発生します。
    - フレーズ内にプレフィクス単語以外の単語がない場合、そのプレフィクス単語は N-gram フレーズに変換され、アスタリスクは削除されます。たとえば、3-gram のテキストインデックスの場合、'"purpl*"' を検索することは、'"pur urp rpl"' を検索することと同義になります。
    - プレフィクス単語がフレーズの末尾に位置する場合は、アスタリスクは削除され、すべての単語は N-gram のフレーズに変換されます。たとえば、3-gram のテキストインデックスの場合、'"purple blac*"' を検索することは、'"pur urp rpl ple bla lac"' を検索することと同義になります。
    - プレフィクス単語がフレーズの末尾に位置していない場合、フレーズは複数のフレーズに分割され、AND で結合されます。たとえば、3-gram のテキストインデックスの場合、'"purp* blac*"' を検索することは、'"pur urp" AND "bla lac"' を検索することと同義になります。
- プレフィクス単語が近接検索の引数の場合、近接検索は AND に変換されます。たとえば、3-gram のテキストインデックスの場合、'red NEAR[1] appl*' を検索することは、'red AND "app ppl"' を検索することと同義になります。

関連情報

テキストインデックスの概念と参照 [292 ページ]

1.3.2.2.3 全文近接検索

全文検索では、単一のカラム内で互いに近接する単語を検索できる機能があります。これは近接検索と呼ばれます。

近接検索を実行するには、2 つの単語を指定して、間に NEAR キーワードまたはチルダ (~) を挿入します。

NEAR キーワードを使用して、整数引数を指定し、最大距離を使用できます。たとえば、term1NEAR[5]term2 は term2 から 5 語以内に出現する term1 のインスタンスを検索します。単語の順序は重要ではなく、'term1 NEAR term2' は 'term2 NEAR term1' と同義です。

距離を指定しない場合、データベースサーバはデフォルトの距離として 10 を使用します。

NEAR キーワードの代わりに、チルダ (~) を指定することもできます。たとえば、'term1 ~ term2' のように記述します。ただし、チルダ形式を使用する場合は距離を指定できず、デフォルトの 10 語が適用されます。

近接検索の引数としてフレーズを指定することはできません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

NGRAM テキストインデックスを使用した近接検索でプレフィクス単語を引数として指定すると、近接検索は AND 式に変換されます。たとえば、3-gram のテキストインデックスの場合、'red NEAR[1] appl*' を検索することは、'red AND "app ppl"' を検索することと同義になります。これは近接検索ではなくなったため、CONTAINS 句で複数のカラムが指定されている場合に、検索が 1 つのカラムに制限されることはありません。

例

MarketingInformation テーブルの Description カラムで、単語 skin から 10 語以内の距離にある単語 fabric を検索するとします。この場合、次の文を実行できます。

```
SELECT ID, "contains".score, Description
   FROM MarketingInformation CONTAINS ( Description, 'fabric ~ skin' );
```

ID	score	Description
902	1.5572371866083279	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester **fabric** is gentle on the earth and soft against your **skin**. </body></html>

デフォルトの距離が 10 語なので、距離を指定する必要はありませんでした。しかし、距離を 1 語拡張することで、次のような別のローが返されます。

```
SELECT ID, "contains".score, Description
   FROM MarketingInformation CONTAINS ( Description, 'fabric NEAR[11] skin' );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	score	Description
903	1.5787803210404958	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The **fabric** has a wicking finish to pull perspiration away from your **skin**. </body></html>
902	2.163125855043747	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester **fabric** is gentle on the earth and soft against your **skin**. </body></html>

単語間の距離が近いため、ID 903 のスコアの方が高くなります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.2.4 全文検索ブール検索

全文検索を実行するときに、複数の単語をブール演算子 (AND、OR、AND NOT など) で区切って指定できます。

全文検索での AND 演算子の使用

AND 演算子では、AND の両側に指定された単語を両方とも含むローが一致となります。AND 演算子としてアンパサンド (&) を使用することもできます。複数の単語を、間に演算子を入れないで指定した場合、AND を暗黙で指定したことになります。

単語がリストされる順番は重要ではありません。

たとえば、次の文はすべて、MarketingInformation テーブルの Description カラムで **fabric** という単語と、**ski** で始まる単語を含むローを検索します。

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'ski* AND fabric' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric & ski*' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'ski* fabric' );
```

全文検索での OR 演算子の使用

OR 演算子では、OR の両側に指定された検索語のうちの少なくとも 1 つを含むローが一致となります。OR 演算子としてパイプ記号 (|) を使用することもできます。この 2 つは同等です。

単語がリストされる順番は重要ではありません。

たとえば、次の文はいずれも、MarketingInformation テーブルの Description カラムで **fabric** という単語、または **ski** で始まる単語を含むローを返します。

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'ski* OR fabric' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric | ski*' );
```

全文検索での AND NOT 演算子の使用

AND NOT 演算子は、左の引数に一致し、右の引数に一致しない結果を検索します。AND NOT 演算子としてハイフン (-) を使用することもできます。この 2 つは同等です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば、次の文は同義であり、いずれも **fabric** という単語を含み、**ski** で始まる単語は含まないローを返します。

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric AND NOT ski*' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric -ski*' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric & -ski*' );
```

異なるブール演算子の組み合わせ

ブール演算子は、クエリ文字列で組み合わせて使用できます。たとえば、次の文は同義であり、いずれも MarketingInformation テーブルの Description カラムで **fabric** と **skin** を含み、**cotton** は含まない項目を検索します。

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'skin fabric -cotton' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric -cotton AND skin' );
```

次の文は同義であり、いずれも MarketingInformation テーブルの Description カラムで **fabric** を含むか、または **cotton** と **skin** の両方を含む項目を検索します。

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric | cotton AND skin' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'cotton skin OR fabric' );
```

単語とフレーズのグループ化

単語と式はカッコを使用してグループ化できます。たとえば、次の文は MarketingInformation テーブルの Description カラムで **cotton** または **fabric** を含み、**ski** で始まる単語を含む項目を検索します。

```
SELECT ID, Description FROM MarketingInformation
   WHERE CONTAINS( MarketingInformation.Description, '( cotton OR fabric ) AND
shi*' );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical **shi**rt is designed for high-intensity workouts in hot and humid weather. The recycled polyester **fabric** is gentle on the earth and soft against your skin. </body></html>
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual **shi**rt made of recycled water bottles. It will serve you equally well on trails or around town. The **fabric** has a wicking finish to pull perspiration away from your skin. </body></html>
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. **Shi**elds against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>

複数カラムに渡る検索

すべてのカラムが同じテキストインデックスの一部である場合は、1 つのクエリで複数のカラムに渡る全文検索を実行できます。

```
SELECT *
   FROM t
      WHERE CONTAINS ( t.c1, t.c2, 'term1|term2' );
```

```
SELECT *
   FROM t
   WHERE CONTAINS( t.c1, 'term1' )
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
        OR CONTAINS( t.c2, 'term2' );
```

t1.c1 に term1 が含まれているか、または t1.c2 に term2 が含まれている場合、最初のクエリは一致します。

t1.c1 または t1.c2 に term1 または term2 が含まれている場合、2 番目のクエリは一致します。この方法で contains を使用すると、一致のスコアも返されます。

関連情報

全文検索結果のスコア [276 ページ]

1.3.2.2.5 全文あいまい検索

あいまい検索を使用して、単語の綴り間違いや変形を検索できます。

ファジー検索を実行するには、FUZZY 演算子の後に二重引用符で囲まれた文字列を付けて、文字列の近似一致を検索します。たとえば、`CONTAINS ( Products.Description, 'FUZZY "cotton"' )` では、`cotton` が返されるほか、`coton` や `cotten` などの綴り間違いが返されます。

> **注記**
>
> あいまい検索は、NGRAM 単語区切りを使用して作成されたテキストインデックスにしか実行できません。

FUZZY 演算子を使用することは、文字列を長さ `n` の部分文字列に手動で分割し、それらを OR 演算子で区切ることと同じです。たとえば、NGRAM 単語区切りと MAXIMUM TERM LENGTH 3 を指定して設定されたテキストインデックスがあるとします。`'FUZZY "500 main street"'` を指定することは、`'500 OR mai OR ain OR str OR tre OR ree OR eet'` を指定することと同義になります。

FUZZY 演算子は、スコアを返す全文検索で便利です。これは、多くの近似一致が返されても、通常は最高スコアを持つ一致だけが意味のある一致であるためです。

関連情報

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]

1.3.2.2.6 ビューに対する全文検索

ビューまたは派生テーブルで全文検索を使用するには、全文検索を実行したいベーステーブルのカラムにテキストインデックスを構築する必要があります。

次の文では、サンプルデータベースの MarketingInformation テーブルにビューを作成します。ビューにはすでにテキストインデックス名があるため、そのビューで全文検索を実行できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

MarketingInformation ベーステーブルにビューを作成するには、次の文を実行します。

```
CREATE VIEW MarketingInfoView AS
SELECT MI.ProductID AS ProdID,
     MI."Description" AS "Desc"
FROM GROUPO.MarketingInformation AS MI
WHERE MI."ID" > 3
```

次の文を使用すると、基本となるテーブルのテキストインデックスを使用してビューを問い合わせることができます。

```
SELECT *
FROM MarketingInfoView
WHERE CONTAINS ( "Desc", 'Cap OR Tee*' )
```

また、次の文を実行すると、基本テーブルのテキストインデックスを使用して派生テーブルにクエリを実行できます。

```
SELECT *
FROM (
     SELECT MI.ProductID, MI."Description"
     FROM MarketingInformation AS MI
     WHERE MI."ID" > 4 ) AS dt ( P_ID, "Desc" )
WHERE CONTAINS ( "Desc", 'Base*' )
```

注記

全文検索を実行するカラムが、ビューまたは派生テーブルの SELECT リストに含まれている必要があります。

基本となるベーステーブルのテキストインデックスを使用してビューを検索する場合、次のような制限事項があります。

- ビューには、TOP 句、FIRST 句、DISTINCT 句、GROUP BY 句、ORDER BY 句、UNION 句、INTERSECT 句、EXCEPT 句、または Window 関数を含めることはできません。
- ビューに集合関数を含めることはできません。
- CONTAINS クエリはビュー内のベーステーブルは参照できるが、別のビュー内にあるビュー内のベーステーブルは参照できません。

1.3.2.3 全文検索結果のスコア

クエリの FROM 句に CONTAINS 句を含めると、それぞれの一致にスコアが関連付けられます。

スコアは一致の近さを示し、スコア情報を使用してデータをソートできます。

スコアは、主に次の 2 つの基準に基づいて付けられます。

インデックス付けされたローにその単語が出現する回数

その単語が、インデックス付けされたローに多く出現するほどスコアは高くなります。

テキストインデックスにその単語が出現する回数

その単語が、テキストインデックスに多く出現するほどスコアは低くなります。SQL Central では、テキストインデックスのボキャブラリタブを表示することで、テキストインデックスに出現する各単語の回数を確認できます。単語をアルファベット順にソートするには、term カラムをクリックします。freq カラムに、その単語がテキストインデックスに出現する回数が表示されます。

また、全文検索のタイプによっては、スコアに影響する基準が他にもあります。たとえば近接検索では、検索語にどれだけ近接しているかがスコアに影響します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

スコアの使用方法

デフォルトでは、CONTAINS 句の結果セットには *contains* という相関名があり、*score* というカラムが 1 つ含まれています。`"contains".score` は、SELECT リスト、ORDER BY 句、またはクエリの他の部分で参照できます。ただし、contains は SQL の予約語であるため、二重引用符で囲む必要があります。また、別の相関名を指定する方法もあります (たとえば、`CONTAINS ( expression ) AS ct`)。このマニュアルで使用している全文検索の例では、スコアカラムは `ct.score` と記載しています。

次の文は、MarketingInformation テーブルの Description カラムで **stretch** または **comfort** で始まる単語を検索します。

```
SELECT ID, ct.score, Description
    FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'stretch* | comfort*' ) AS ct
    ORDER BY ct.score DESC;
```

ID	score	Description
910	5.570408968026068	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying cotton shorts provide all day **comfort** on or off the trails. Now with a more **comfort**able and **stretch**y fabric and an adjustable drawstring waist. </body></html>

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	score	Description
907	3.658418186470189	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of **stretch** to give you a snug yet **comfort**able fit every time you wear it. </body></html>
905	1.6750365447462499	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> A lightweight wool cap with mesh side vents for breathable **comfort** during aerobic activities. Moisture-absorbing headband liner. </body></html>

プレフィクス単語 `comfort` のインスタンスが、他の項目には 1 つしかないのに対して、項目 910 には 2 つ含まれているため、910 のスコアが最も高くなります。また、項目 910 にはプレフィクス単語 `stretch` のインスタンスも含まれています。

例

次の例では、複数のカラムに渡る全文検索を実行し、結果のスコアを表示する方法を示しています。

1. 次のようにして、Products テーブルで即時テキストインデックスを作成します。

```
CREATE TEXT INDEX scoringExampleMult
   ON Products ( Description, Name );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. 次にようにして、Description カラムと Name カラムで、cap または visor という単語の全文検索を実行します。CONTAINS 句の結果には ct という相関名が割り当てられ、結果に含まれるように SELECT リストで参照されます。また、ct.score カラムは ORDER BY 句で参照され、結果はスコアの降順でソートされます。

```
SELECT Products.Description, Products.Name, ct.score
    FROM Products CONTAINS ( Products.Description, Products.Name, 'cap OR 
visor' ) ct
    ORDER BY ct.score DESC;
```

Description	Name	score
Cloth Visor	Visor	3.5635154905713042
Plastic Visor	Visor	3.4507856451176244
Wool cap	Baseball Cap	3.2340501745357333
Cotton Cap	Baseball Cap	3.090467108972918

複数カラムの検索のスコアは、カラム値が連結され、1 つの値としてインデックス付けされているかのように計算されます。ただし、そのフレーズと NEAR 演算子は、カラムの境界を超えて一致することはなく、複数のカラムに出現する検索語は、単一の連結された値よりもスコアが大きくなります。

3. このマニュアルの他の例を正常に動作させるためには、Products テーブルで作成したテキストインデックスを削除する必要があります。削除するには、次の文を実行します。

```
DROP TEXT INDEX scoringExampleMult ON Products;
```

1.3.2.4 テキスト設定オブジェクトの概念と参照

テキスト設定オブジェクトは、テキストインデックスの作成時または再表示時に格納される単語を制御し、全文クエリの解釈方法を制御します。

各テキスト設定オブジェクトの設定は、ISYSTEXTCONFIG システムテーブルにローとして格納されます。

データベースサーバがテキストインデックスを作成または再表示する場合、テキストインデックスの作成時に指定されたテキスト設定オブジェクトの設定が使用されます。テキストインデックスの作成時にテキスト設定オブジェクトを指定しなかった場合、データベースサーバは、インデックス付けされるカラムのデータ型に基づいてデフォルトのテキスト設定オブジェクトを 1 つ選択します。デフォルトのテキスト設定オブジェクトが 2 つ用意されています。

既存のテキスト設定オブジェクトの設定を表示するには、SYSTEXTCONFIG システムビューにクエリを発行します。

このセクションの内容:

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]
テキスト設定オブジェクトの作成時または変更時には数多くの指定項目があります。

テキスト設定オブジェクトに影響するデータベースオプション [285 ページ]
テキスト設定オブジェクトが作成されると、date_format、time_format、timestamp_format の各データベースオプションの現在の設定はテキスト設定オブジェクトに格納されます。

テキスト設定オブジェクトの例 [286 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テキスト設定オブジェクトで文字列がどのように単語に分割されるのかは、システムプロシージャ sa_char_terms および sa_nchar_terms を使用してテストできます。

1.3.2.4.1 テキスト設定オブジェクトの作成時または変更時の指定項目

テキスト設定オブジェクトの作成時または変更時には数多くの指定項目があります。

CHAR データに使用する default_char、および NCHAR データと CHAR データに使用する default_nchar という、デフォルトのテキスト設定オブジェクトが 2 つ用意されています。

default_nchar はすべてのデータに使用できますが、文字セット変換が実行されます。

テキスト設定オブジェクトがどのように単語の分割に影響するかは、sa_char_terms および sa_nchar_terms システムプロシージャを使用してテストできます。

このセクションの内容:

TERM BREAKER 句 ー 単語区切りアルゴリズムの指定 [281 ページ]
TERM BREAKER 設定は、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を指定します。

MINIMUM TERM LENGTH 句 ー 単語の最小長の設定 [282 ページ]
MINIMUM TERM LENGTH 設定で、インデックスに挿入される単語、または全文クエリで検索される単語の最小長 (文字数) を指定します。

MAXIMUM TERM LENGTH 句 ー 単語の最大長の設定 [283 ページ]
MAXIMUM TERM LENGTH 設定がどのように使用されるかは、単語区切りアルゴリズムによって異なります。

STOPLIST 句 ー ストップリストの設定 [283 ページ]
STOPLIST 句は、テキストインデックスの作成時に無視する単語を指定します。

PREFILTER 句 ー 外部プレフィルタアルゴリズムの指定 [284 ページ]
PREFILTER 句は、Word、PDF、HTML、XML などのファイルタイプからテキストデータを抽出するために使用する外部プレフィルタアルゴリズムを指定します。

日付、時刻、およびタイムスタンプ形式の設定 [285 ページ]
テキスト設定オブジェクトが作成されると、現在の接続の date_format、time_format、timestamp_format、timestamp_with_time_zone_format の各オプションの値はテキスト設定オブジェクトに格納されます。

関連情報

テキスト設定オブジェクトの例 [286 ページ]
全文あいまい検索 [275 ページ]
テキストインデックスの概念と参照 [292 ページ]
テキスト設定オブジェクトの作成 (SQL Central の場合) [251 ページ]
テキスト設定オブジェクトの変更 [252 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.4.1.1 TERM BREAKER 句 ー 単語区切りアルゴリズムの指定

TERM BREAKER 設定は、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を指定します。

単語を格納する GENERIC、または N-gram を格納する NGRAM から選択できます。GENERIC では、組み込みの単語区切りアルゴリズムまたは外部単語区切りを使用できます。

次の表は、TERM BREAKER の値がテキストインデックス処理およびクエリ文字列の処理方法に与える影響を示します。

テキストインデックス	クエリ文字列
GENERIC テキストインデックス GENERIC テキストインデックスは、NGRAM テキストインデックスをパフォーマンスで上回ります。ただし、GENERIC テキストインデックスではあいまい検索は実行できません。 組み込みのアルゴリズムを使用して GENERIC テキストインデックスを作成すると、英数字以外の文字に挟まれた英数字の文字グループはデータベースサーバによって単語として処理され、位置が割り当てられます。 単語区切り外部ライブラリを使用して GENERIC テキストインデックスを作成すると、単語と単語の位置は外部ライブラリによって定義されます。 単語が単語区切りによって識別されると、単語長の制限値を超える単語やストップリストに含まれる単語は、カウントはされますがテキストインデックスには挿入されません。 **NGRAM** テキストインデックス **N-gram** とは、長さ `n` の文字のグループです。`n` は MAXIMUM TERM LENGTH の値です。 NGRAM テキストインデックスを作成すると、データベースサーバは、英数字以外の文字に挟まれた英数字の文字グループをすべて単語として処理します。単語が定義されると、データベースサーバは単語を N-gram に分割します。これにより、`n` よりも短い単語と、ストップリストに含まれている N-gram は破棄されます。 たとえば、MAXIMUM TERM LENGTH が 3 の NGRAM テキストインデックスでは、'my red table' という文字列はテキストインデックス内で red tab abl ble という N-gram として表されます。 N-gram の場合、元の単語の位置情報ではなく、N-gram の位置情報が格納されます。	CONTAINS クエリを解析する場合、データベースサーバはクエリ文字列からキーワードと特殊文字を抽出し、残りの単語に単語区切りアルゴリズムを適用します。たとえば、クエリ文字列が `'ab_cd* AND b*'` の場合、* とキーワード AND は抽出され、文字列 ab_cd と b は単語区切りアルゴリズムに渡されて別に解析されます。 **GENERIC** テキストインデックス GENERIC テキストインデックスを問い合わせると、クエリ文字列内の単語は、インデックス付けされる場合と同じ方法で処理されます。クエリ単語をテキストインデックス内の単語と比較することで、照合が実行されます。 **NGRAM** テキストインデックス NGRAM テキストインデックスを問い合わせると、クエリ文字列内の単語は、インデックス付けされる場合と同じ方法で処理されます。クエリ単語の N-gram をインデックス付けされた単語の N-gram と比較することで、マッチングが実行されます。

定義されていない場合、TERM BREAKER のデフォルトは、デフォルトのテキスト設定オブジェクトで設定された値になります。デフォルトのテキスト設定オブジェクトで単語区切りが定義されていない場合、内部単語区切りが使用されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

全文プレフィクス検索 [266 ページ]
テキスト設定オブジェクトの例 [286 ページ]

1.3.2.4.1.2 MINIMUM TERM LENGTH 句 ─ 単語の最小長の設定

MINIMUM TERM LENGTH 設定で、インデックスに挿入される単語、または全文クエリで検索される単語の最小長 (文字数) を指定します。

MINIMUM TERM LENGTH は、NGRAM テキストインデックスでは関係ありません。

MINIMUM TERM LENGTH は、プレフィクス検索に対して特別な影響があります。

MINIMUM TERM LENGTH の値は、0 より大きくする必要があります。MAXIMUM TERM LENGTH よりも大きい値を設定した場合、MAXIMUM TERM LENGTH は、MINIMUM TERM LENGTH と等しい値に自動的に調整されます。

定義されていない場合、MINIMUM TERM LENGTH のデフォルト値は、デフォルトのテキスト設定オブジェクトで設定された値 (通常は 1) になります。

次の表に、MINIMUM TERM LENGTH の値がテキストインデックス処理およびクエリ文字列の処理方法に与える影響を示します。

テキストインデックス	クエリ文字列
GENERIC テキストインデックス GENERIC テキストインデックスでは、MINIMUM TERM LENGTH よりも短い単語はテキストインデックスに含まれません。 **NGRAM テキストインデックス** NGRAM テキストインデックスでは、この設定は無視されます。	**GENERIC テキストインデックス** GENERIC テキストインデックスを問い合わせる場合、MINIMUM TERM LENGTH よりも短いクエリ単語はテキストインデックスに存在できないため、無視されます。 **NGRAM テキストインデックス** MINIMUM TERM LENGTH 設定は、NGRAM テキストインデックスの全文クエリには影響ありません。

関連情報

全文プレフィクス検索 [266 ページ]
テキスト設定オブジェクトの例 [286 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.4.1.3 MAXIMUM TERM LENGTH 句 — 単語の最大長の設定

MAXIMUM TERM LENGTH 設定がどのように使用されるかは、単語区切りアルゴリズムによって異なります。

MAXIMUM TERM LENGTH の値は、60 以下にする必要があります。MINIMUM TERM LENGTH よりも小さい値を設定した場合、MINIMUM TERM LENGTH は、MAXIMUM TERM LENGTH と等しい値に自動的に調整されます。

定義されていない場合、MAXIMUM TERM LENGTH のデフォルト値は、デフォルトのテキスト設定オブジェクトで設定された値 (通常は 20) になります。

次の表に、MAXIMUM TERM LENGTH の値がテキストインデックス処理およびクエリ文字列の処理方法に与える影響を示します。

テキストインデックス	クエリ文字列
GENERIC テキストインデックス GENERIC テキストインデックスでは、MAXIMUM TERM LENGTH はテキストインデックスに挿入される単語の最大長を文字数で指定します。 **NGRAM テキストインデックス** NGRAM テキストインデックスでは、MAXIMUM TERM LENGTH は単語が分割される N-gram の長さを決定します。N-gram の適切な長さの選択は、言語によって異なります。英語の場合の一般的な値は 4 文字または 5 文字で、中国語の場合は 2 文字または 3 文字です。	**GENERIC テキストインデックス** GENERIC テキストインデックスでは、MAXIMUM TERM LENGTH よりも長いクエリ単語はテキストインデックスに存在できないため、無視されます。 **NGRAM テキストインデックス** NGRAM テキストインデックスでは、クエリ単語は長さ `n` の N-gram に分割されます。`n` は MAXIMUM TERM LENGTH と同じ値です。データベースサーバは N-gram を使用してテキストインデックスを検索します。MAXIMUM TERM LENGTH よりも短い単語はテキストインデックスの N-gram と一致しないため、無視されます。このため、引数が長さ `n` のプレフィクスでないかぎり、近接検索は機能しません。

関連情報

テキスト設定オブジェクトの例 [286 ページ]

1.3.2.4.1.4 STOPLIST 句 — ストップリストの設定

STOPLIST 句は、テキストインデックスの作成時に無視する単語を指定します。

ストップリストが定義されていない場合は、デフォルトのテキスト設定オブジェクトの設定 (通常は空のストップリスト) が採用されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

STOPLIST のテキストインデックスへの影響	STOPLIST のクエリ単語への影響
GENERIC テキストインデックス GENERIC テキストインデックスでは、ストップリストに含まれる単語はテキストインデックスに挿入されません。 **NGRAM テキストインデックス** NGRAM テキストインデックスでは、ストップリスト内の単語から構成される N-gram はテキストインデックスには含まれません。	**GENERIC テキストインデックス** GENERIC テキストインデックスでは、ストップリストに含まれるクエリ単語はテキストインデックスに存在できないため、無視されます。 **NGRAM テキストインデックス** ストップリストに含まれる単語は N-gram に分割され、N-gram は単語のフィルタリングで使用されます。同様に、クエリ単語も N-gram に分割され、ストップリスト内の N-gram と一致するものはテキストインデックスには存在できないため、削除されます。

テキスト設定オブジェクトの設定は、解析時にストップリストに適用されます。つまり、指定された単語区切りと最小長設定/最大長設定が適用されます。

NGRAM テキストインデックスのストップリストは、指定したストップリストの単語ではなく、N-gram の形式で格納されるため、予期しない結果になる場合があります。たとえば、MAXIMUM TERM LENGTH が 3 の NGRAM テキストインデックスの場合、`STOPLIST 'there'` を指定すると、the her ere の N-gram がストップリストとして格納されます。これは、the、her、ere の N-gram を含むすべての単語に対する問い合わせに影響します。

> **注記**
>
> 文字列リテラルの指定に関連する制約事項は、ストップリストにも該当します。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。

Samples ディレクトリには、複数の言語用のストップリストをロードするサンプルコードが含まれています。これらのサンプルストップリストは、GENERIC テキストインデックスでのみ使用することをお奨めします。

関連情報

テキスト設定オブジェクトの例 [286 ページ]

1.3.2.4.1.5 PREFILTER 句 – 外部プレフィルタアルゴリズムの指定

PREFILTER 句は、Word、PDF、HTML、XML などのファイルタイプからテキストデータを抽出するために使用する外部プレフィルタアルゴリズムを指定します。

テキストインデックス処理のコンテキストでは、プレフィルタリングを使用して、インデックス処理対象のデータのみを抽出し、HTML タグなどの不要なコンテンツがインデックス処理されないようにできます。特定のドキュメントタイプ (Microsoft Word ドキュメントなど) では、全文インデックスを有用なものにするために、プレフィルタリングが必要です。

組み込みのプレフィルタアルゴリズムはありません。ただし、要件に応じてプレフィルタリングを実行する外部プレフィルタライブラリを作成し、その外部プレフィルタライブラリを指すようにテキスト設定オブジェクトを変更できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の表は、PREFILTER EXTERNAL NAME の値がテキストインデックス処理およびクエリ文字列の処理方法に与える影響を示します。

テキストインデックス	クエリ文字列
GENERIC テキストインデックスと NGRAM テキストインデックス 外部プレフィルタは入力値 (ドキュメント) を受け入れ、プレフィルタライブラリで指定されたルールに従って入力値をフィルタリングします。その結果生成されたテキストは単語区切りに渡され、その後テキストインデックスが作成または更新されます。	**GENERIC テキストインデックスと NGRAM テキストインデックス** クエリ文字列はプレフィルタを介して渡されないため、PREFILTER EXTERNAL NAME 句の設定はクエリ文字列に影響を与えません。

SQL Anywhere インストールの `ExternalLibrariesFullText` ディレクトリには、参照用に、プレフィルタと単語区切りのサンプルコードが含まれています。このディレクトリは、`Samples` ディレクトリ内にあります。

関連情報

外部プレフィルタライブラリ [328 ページ]

1.3.2.4.1.6 日付、時刻、およびタイムスタンプ形式の設定

テキスト設定オブジェクトが作成されると、現在の接続の date_format、time_format、timestamp_format、timestamp_with_time_zone_format の各オプションの値はテキスト設定オブジェクトに格納されます。

これらのオプション値は、テキスト設定オブジェクトを使用して作成されたテキストインデックスの DATE、TIME、TIMESTAMP の各カラムがフォーマットされる方法を制御します。テキスト設定オブジェクトのこれらのオプション値を明示的に設定することはできません。設定には、テキスト設定オブジェクトを作成した接続に対して有効になっているオプション値が反映されます。ただし、これらのオプション値の変更は可能です。

関連情報

テキスト設定オブジェクトの変更 [252 ページ]

1.3.2.4.2 テキスト設定オブジェクトに影響するデータベースオプション

テキスト設定オブジェクトが作成されると、date_format、time_format、timestamp_format の各データベースオプションの現在の設定はテキスト設定オブジェクトに格納されます。

このようになるのは、テキスト設定オブジェクトに依存するテキストインデックスの作成時および再表示時に、これらの設定が文字列の変換に影響するためです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

設定をテキスト設定オブジェクトに格納することで、依存するテキストインデックスに格納されるデータのフォーマットを変えることなく、データベースオプションの設定を変更できます。

テキストインデックスにある日付や時刻を表す文字列の形式を変更するには、次の手順に従います。

1. テキストインデックス、テキスト設定オブジェクト、およびすべての依存するテキストインデックスを削除します。
2. テキスト設定オブジェクトの作成に使用したデフォルトのテキスト設定オブジェクトとすべての依存するテキストインデックスを削除します。
3. 日付、時刻、タイムスタンプのフォーマットオプションを目的の形式に変更します。
4. テキスト設定オブジェクトを作成します。
5. 新しいテキスト設定オブジェクトを使用して、テキストインデックスを作成します。

注記

テキストインデックスを作成または再表示する場合は、conversion_error オプションを ON に設定する必要があります。

関連情報

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]

1.3.2.4.3 テキスト設定オブジェクトの例

テキスト設定オブジェクトで文字列がどのように単語に分割されるのかは、システムプロシージャ sa_char_terms および sa_nchar_terms を使用してテストできます。

データベースのすべてのテキスト設定オブジェクト、およびその設定に関する一覧については、SYSTEXTCONFIG システムビューを問い合わせてください (例:`SELECT * FROM SYSTEXTCONFIG`)。

デフォルトのテキスト設定オブジェクト

NCHAR データに使用する default_nchar 、および非 NCHAR データに使用する default_char という、2 つのデフォルトのテキスト設定オブジェクトが用意されています。これらの設定は、テキスト設定オブジェクトまたはテキストインデックスを初めて作成する際に登録されます。

インストール時における default_char と default_nchar の設定を以下の表に示します。これらの設定は、ほとんどの文字ベースの言語で最適であるという理由で選択されています。デフォルトのテキスト設定オブジェクトの設定を変更しないことを強くお奨めします。

設定	インストールされる値
TERM BREAKER	0 (GENERIC)
MINIMUM TERM LENGTH	1

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

設定	インストールされる値
MAXIMUM TERM LENGTH	20
STOPLIST	(空)

デフォルトのテキスト設定オブジェクトを削除すると、テキストインデックスやテキスト設定オブジェクトを次回作成する際に自動的に作成されます。

データベースサーバによってデフォルトのテキスト設定オブジェクトが作成されると、日付値と時刻値が文字列に変換される方法に影響を与えるデータベースオプションは、現在の設定からテキスト設定オブジェクトに保存されます。

テキスト設定オブジェクトの例

次の表は、さまざまなテキスト設定オブジェクトの設定、インデックス付けされる対象に与える影響、および全文クエリ文字列の解釈方法を示したものです。すべての例で、文字列 `'I'm not sure I understand'` を使用しています。

設定	インデックス付けされる単語	クエリ解釈
TERM BREAKER GENERIC MINIMUM TERM LENGTH 1 MAXIMUM TERM LENGTH 20 STOPLIST ''	`I m not sure I understand`	`("I m" AND NOT sure) AND I AND understand'` 元の文字列の 'not' は 'not' というワードではなく、演算子として解釈されます。
TERM BREAKER GENERIC MINIMUM TERM LENGTH 2 MAXIMUM TERM LENGTH 20 STOPLIST 'not and'	`sure understand`	`'understand'` 'not' はフレーズ "i am" と "sure" の間の演算子 (AND NOT) として解釈されるため、'sure' は削除されます。フレーズ "i am" には短すぎて削除される単語が含まれているため、AND NOT 条件の右側 ('sure') も削除されます。これにより、'understand' のみが残ります。
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 3 STOPLIST 'not and'	`sur ure und nde der ers rst sta tan`	`'und AND nde AND der AND ers AND rst AND sta AND tan'` が含まれます。 あいまい検索の場合: `'und OR nde OR der OR ers OR rst OR sta OR tan'`
TERM BREAKER GENERIC MINIMUM TERM LENGTH 1 MAXIMUM TERM LENGTH 20 STOPLIST 'not and'	`I m sure I understand`	`'("I m" AND NOT sure) AND I AND understand'` が含まれます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

設定	インデックス付けされる単語	クエリ解釈
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 20 STOPLIST 'not and'	20 文字以上の単語がないため、インデックス付けされるものはありません。 これは、MAXIMUM TERM LENGTH の GENERIC テキストインデックスへの影響と NGRAM テキストインデックスへの影響が異なることを示しています。NGRAM テキストインデックスの場合、MAXIMUM TERM LENGTH は、テキストインデックスに挿入される N-gram の長さを設定します。	クエリ文字列から 20 文字の N-gram を構成できないため、検索を実行すると空の結果セットが返されます。

CONTAINS 文字列の解釈方法の例

次の表は、テキスト設定オブジェクトの文字列の設定がどのように解釈されるかについて、例を示したものです。

解釈される文字列の列にあるカッコ内の数字は、各単語の位置情報を表しています。この数字は、説明の目的でマニュアルに記載されています。格納される実際の単語にはカッコ内の数字は含まれません。

TERM BREAKER GENERIC MINIMUM TERM LENGTH 3 MAXIMUM TERM LENGTH 20	`'w*'`	`'"w*(1)"'`
	`'we*'`	`'"we*(1)"'`
	`'wea*'`	`'"wea*(1)"'`
	`'we* -the'`	`'"we*(1)" -"the(1)"'`
	`'we* the'`	`"we*(1)" & "the(1)"'`
	`'for* | wonderl*'`	`'"for*(1)" | "wonderl*(1)"'`
	`'wonderlandwonderlandwonderland*'`	`''`
	`'"tr* weather"'`	`'"weather(1)"'`
	`'"tr* the weather"'`	`'"the(1) weather(2)"'`

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	'"wonderlandwonderlandwonderland* wonderland"'	'"wonderland(1)"'
	'"wonderlandwonderlandwonderland* weather"'	'"weather(1)"'
	'"the_wonderlandwonderlandwonderland* weather"'	'"the(1) weather(3)"'
	'the_wonderlandwonderlandwonderland* weather'	'"the(1)" & "weather(1)"'
	'"light_a* the end" & tunnel'	'"light(1) the(3) end(4)" & "tunnel(1)"'
	light_b* the end" & tunnel'	'"light(1) the(3) end(4)" & "tunnel(1)"'
	'"light_at_b* end"'	'"light(1) end(4)"'
	'and_te*'	'"and(1) te*(2)"'
	'a_long_and_t* & journey'	'"long(2) and(3) t*(4)" & "journey(1)"'
	'weather -is'	'"weather(1)"'
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 3	'w*'	'"w*(1)"'
	'we*'	'"we*(1)"'
	'wea*	'"wea(1)"'
	'we* -the'	'"we*(1)" -"the(1)"'
	'we* the'	'"we*(1)" & "the(1)"'
	'for | la*'	'"we*(1)" & "the(1)"'
	'weath*'	'"for(1)" | "la*(1)"'

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	`'"ful weat*"'`	`'"wea(1) eat(2) ath(3)"'`
	`'"wo* la*"'`	`'"wo*(1)" & "la*(2)"'`
	`'"la* won* "'`	`'"la*(1)" & "won(2)"'`
	`'"won* weat*"'`	`'"won(1)" & "wea(2) eat(3)"'`
	`'"won* weat"'`	`'"won(1)" & "wea(2) eat(3)"'`
	`'"weat* wo* "'`	`'"wea(1) eat(2)" & "wo*(3)"'`
	`'"wo* weat"'`	`'"wo*(1)" & "wea(2) eat(3)"'`
	`'"weat wo* "'`	`'"wea(1) eat(2) wo*(3)"'`
	`'w* NEAR[1] f*'`	`'"w*(1)" & "f*(1)"'`
	`'weat* NEAR[1] f*'`	`"wea(1) eat(2)" & "f*(1)"'`
	`'f* NEAR[1] weat*'`	`'"f*(1)" & "wea(1) eat(2)"'`
	`'weat NEAR[1] f*'`	`'"wea(1) eat(2)" & "f*(1)"'`
	`'for NEAR[1] weat*'`	`'"for(1)" & "wea(1) eat(2)"'`
	`'weat* NEAR[1] for'`	`'"wea(1) eat(2)" & "for(1)"'`
	`'and_tedi*'`	`'"and(1) ted(2) edi(3)"'`
	`'and_t*'`	`'"and(1) t*(2)"'`

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	'"and_tedi*"'	'"and(1) ted(2) edi(3)"'
	'"and-t*"'	'"and(1) t*(2)"'
	'"ligh* at_the_end of_the tun* nel"'	'"lig(1) igh(2)" & ("the(4) end(5) the(7) tun(8)" & "nel(9)")'
	'"ligh* at_the_end_of_the_tun* nel"'	'"lig(1) igh(2)" & ("the(4) end(5) the(7) tun(8)" & "nel(9)")'
	'"at_the_end of_the tun* ligh* nel"'	'"the(2) end(3) the(5) tun(6)" & ("lig(7) igh(8)" & "nel(9)")'
	'l* NEAR[1] and_t*'	"l*(1)" & "and(1) t*(2)"'
	'long NEAR[1] and_t*'	'"lon(1) ong(2)" & "and(1) t*(2)"'
	'end NEAR[3] tunne*'	'"end(1)" & "tun(1) unn(2) nne(3)"'
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 3 SKIPPED TOKENS IN TABLE AND IN QUERIES	'"cat in a hat"'	'"cat(1) hat(4)"'
	'"cat in_a hat"'	'"cat(1) hat(4)"'
	'"cat_in_a_hat"'	'"cat(1) hat(4)"'
	'"cat_in a_hat"'	'"cat(1) hat(4)"'
	'cat in a hat'	'"cat(1)" & "hat(1)"'
	'cat in_a hat'	'"cat(1)" & "hat(1)"'
	'"ice hat"'	'"ice(1) hat(2)"'

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	`'ice NEAR[1] hat'`	`'"ice(1)" NEAR[1] "hat(1)"'`
	`'ear NEAR[2] hat'`	`'"ear(1)" NEAR[2] "hat(1)"'`
	`'"ear a hat"'`	`'"ear(1) hat(3)"'`
	`'"cat hat"'`	`'"cat(1) hat(2)"'`
	`'cat NEAR[1] hat'`	`'"cat(1)" NEAR[1] "hat(1)"'`
	`'ear NEAR[1] hat'`	`'"ear(1)" NEAR[1] "hat(1)"'`
	`'"ear hat"'`	`'"ear(1) hat(2)"'`
	`'"wear a a hat"'`	`'"wea(1) ear(2) hat(5)"'`
	`'weather -is'`	`'"wea(1) eat(2) ath(3) the(4) her(5)"'`

関連情報

テキスト設定オブジェクトに影響するデータベースオプション [285 ページ]

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]

1.3.2.5 テキストインデックスの概念と参照

テキストインデックスには、インデックス付けされたカラム内の単語の位置情報が格納されます。

全文検索を実行すると、テーブルローではなく、テキストインデックスが検索されます。したがって、全文検索を実行する前に、検索するカラムにテキストインデックスを作成する必要があります。テキストインデックスを使用したクエリは、テーブル内のすべての値をスキャンする必要があるクエリよりも高速になる可能性があります。

テキストインデックスを作成する場合、テキストインデックスの作成と再表示に使用されるテキスト設定オブジェクトを指定できます。テキスト設定オブジェクトには、インデックスの構築方法に影響する設定が含まれます。テキスト設定オブジェクトを指定しない場合、データベースサーバはデフォルトの設定オブジェクトを使用します。

テキストインデックスの再表示タイプも指定できます。再表示タイプによって、テキストインデックスの再表示の頻度が決まります。最後に再表示されたテキストインデックスが最も正確な結果を返します。ただし、再表示には時間がかかるため、パフォー

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

マンスに影響する可能性があります。たとえば、インデックス付けされたテーブルを頻繁に更新する場合、基本となるデータが変更されるたびにテキストインデックスを再表示するように設定すると、パフォーマンスに影響する可能性があります。

テキストインデックスの単語の位置情報が維持されていることを確認するために、VALIDATE TEXT INDEX 文を使用できます。位置情報が維持されていない場合は、エラーが生成されます。

既存のテキストインデックスの設定を表示するには、sa_text_index_stats システムプロシージャを使用します。

このセクションの内容:

テキストインデックスの再表示タイプ [293 ページ]
テキストインデックスを作成するときには、再表示タイプ (即時、自動、手動のいずれか) も選択する必要があります。

関連情報

テキスト設定オブジェクトの概念と参照 [279 ページ]

1.3.2.5.1 テキストインデックスの再表示タイプ

テキストインデックスを作成するときには、再表示タイプ (即時、自動、手動のいずれか) も選択する必要があります。

テキストインデックスを作成するときには、再表示タイプも選択する必要があります。テキストインデックスでは、即時、自動、手動の 3 つの再表示タイプがサポートされています。再表示タイプは、テキストインデックスの作成時に定義します。即時テキストインデックスの場合を除き、再表示タイプはテキストインデックスの作成後に変更できます。

IMMEDIATE REFRESH

IMMEDIATE REFRESH テキストインデックスは、基本となるテーブルまたはマテリアライズドビューのデータが変更されると再表示されます。これは基本テーブルでデータを常に最新の状態に保つ必要がある場合や、インデックス付けされたカラムが比較的短い場合、またはデータ変更の頻度が低い場合にのみ、使用することをお奨めします。

テキストインデックスのデフォルトの再表示タイプは IMMEDIATE REFRESH です。マテリアライズドビューのテキストインデックスは、IMMEDIATE REFRESH のみサポートしています。

AUTO REFRESH または MANUAL REFRESH のテキストインデックスを、IMMEDIATE REFRESH テキストインデックスに変更することはできません。変更したい場合は、該当するテキストインデックスを削除して、IMMEDIATE REFRESH テキストインデックスとして再作成する必要があります。

IMMEDIATE REFRESH テキストインデックスはすべての独立性レベルをサポートします。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、この初期再表示中に、テーブルまたはマテリアライズドビューに排他ロックがかけられます。

AUTO REFRESH

AUTO REFRESH テキストインデックスは、指定された間隔で自動的に再表示されます。データがある程度古くなってもかまわない場合に使用することをお奨めします。古いインデックスを問い合わせると、一致するローのうち、最後に再表示されてから変更されていないものが返されます。したがって、最後の再表示以降に挿入、削除、更新されたローはクエリで返されません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

AUTO REFRESH テキストインデックスは、次のいずれかの条件が当てはまる場合、指定した間隔よりも頻繁に再表示されることがあります。

- 最後の再表示からの時間が再表示間隔より長い場合。
- 保留中のすべてのローの合計長 (sa_text_index_stats システムプロシージャによって返される pending_length) が合計インデックスサイズ (sa_text_index_stats によって返される doc_length) の 20% を超える場合。
- 削除された長さが合計インデックスサイズ (doc_length) の 50 % を超える場合。この場合には、増分更新ではなく常に再構築が実行されます。

AUTO REFRESH テキストインデックスは、独立性レベル 0 を使用して再表示されます。

AUTO REFRESH テキストインデックスには、作成時にはデータが含まれていないため、最初の再表示が行われるまで利用できません。最初の再表示は通常、テキストインデックスが作成されてから 1 分以内に行われます。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もできます。

AUTO REFRESH テキストインデックスは、dbunload で -g オプションを指定しない限り、再ロード時に再表示されません。

MANUAL REFRESH

MANUAL REFRESH テキストインデックスは、ユーザの実行によってのみ再表示されます。基本となるテーブルのデータがめったに変更されない場合、データの古さの程度が大きくても許容できる場合、またはイベントや条件を満たした後に再表示したい場合に使用することをお奨めします。古いインデックスを問い合わせると、一致するローのうち、最後に再表示されてから変更されていないものが返されます。したがって、最後の再表示以降に挿入、削除、更新されたローはクエリで返されません。

MANUAL REFRESH テキストインデックスには、独自の再表示方針を定義できます。次の例では、引数として渡される再表示間隔と、AUTO REFRESH テキストインデックスに適用されるルールと同様のルールを使用して、すべての MANUAL REFRESH テキストインデックスが再表示されます。

```
CREATE PROCEDURE refresh_manual_text_indexes(
   refresh_interval UNSIGNED INT )
BEGIN
 FOR lp1 AS c1 CURSOR FOR
   SELECT ts.*
   FROM SYS.SYSTEXTIDX ti JOIN sa_text_index_stats( ) ts
   ON ( ts.index_id = ti.index_id )
   WHERE ti.refresh_type = 1 -- manual refresh indexes only
 DO
   BEGIN
    IF last_refresh_utc IS null
    OR cast(pending_length as float) / (
       IF doc_length=0 THEN NULL ELSE doc_length ENDIF) > 0.2
    OR DATEDIFF( MINUTE, CURRENT UTC TIMESTAMP, last_refresh_utc )
       > refresh_interval THEN
     EXECUTE IMMEDIATE 'REFRESH TEXT INDEX ' || text-index-name || ' ON "'
     || table-owner || '"."' || table-name || '"';
    END IF;
   END;
  END FOR;
END;
```

いつでも、sa_text_index_stats システムプロシージャを使用して、再表示が必要かどうか、再表示を完全再構築にするか、増分更新にするかを決定できます。

MANUAL REFRESH テキストインデックスには、作成時にはデータが含まれていないため、再表示するまで利用できません。MANUAL REFRESH テキストインデックスを再表示するには、REFRESH TEXT INDEX 文を使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

MANUAL REFRESH テキストインデックスは、dbunload で -g オプションを指定しない限り、再ロード時に再表示されません。

関連情報

テキスト設定オブジェクトの作成時または変更時の指定項目 [280 ページ]
テキストインデックスの作成 [253 ページ]

1.3.2.6 チュートリアル: GENERIC テキストインデックスに対する全文検索の実行

GENERIC 単語区切りを使用するテキストインデックスに対して、全文検索を実行します。

前提条件

CREATE TEXT CONFIGURATION と CREATE TABLE のシステム権限が必要です。さらに、SELECT ANY TABLE システム権限、または、テーブル MarketingInformation に対する SELECT 権限も必要です。

手順

1. Interactive SQL を起動します。スタート > プログラム > *SQL Anywhere17* > 管理ツール > *Interactive SQL* をクリックします。
2. 接続ウィンドウで、次の項目に次のように入力します。
    a. 認証ドロップダウンリストで データベースをクリックします。
    b. ユーザ *ID* 項目に、**DBA** と入力します。
    c. パスワード項目に、**sql** と入力します。
    d. アクションドロップダウンリストで、*ODBC* データソースを使用した接続を選択します。
    e. SQL Anywhere 17 Demo データソースを選択し、*OK* をクリックします。
3. 次の文を実行して、myTxtConfig というテキスト設定オブジェクトを作成します。FROM 句を含めて、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を指定する必要があります。

```
CREATE TEXT CONFIGURATION myTxtConfig FROM default_char;
```

4. 次の文を実行して、because、about、therefore、only という単語を含むストップリストを追加することにより、テキスト設定オブジェクトをカスタマイズします。次に、単語の最大長を 30 に設定します。

```
ALTER TEXT CONFIGURATION myTxtConfig
   STOPLIST 'because about therefore only';
ALTER TEXT CONFIGURATION myTxtConfig
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    MAXIMUM TERM LENGTH 30;
```

5. SQL Central を起動します。 *[スタート]* > *[プログラム]* > *SQL Anywhere17* > *[管理ツール]* > *SQL Central* をクリックします。
6. 接続 > *SQL Anywhere17* に接続 をクリックします。
7. 接続ウィンドウで、次の項目に次のように入力します。
   a. 認証ドロップダウンリストで データベースをクリックします。
   b. ユーザ *ID* 項目に、**DBA** と入力します。
   c. パスワード項目に、**sql** と入力します。
   d. アクションドロップダウンリストで、*ODBC* データソースを使用した接続を選択します。
   e. SQL Anywhere 17 Demo データソースを選択し、*OK* をクリックします。
8. MarketingInformation テーブルのコピーを作成します。
   a. テーブルフォルダを展開します。
   b. *MarketingInformation* を右クリックし、コピーをクリックします。
   c. テーブルフォルダを右クリックし、貼り付けをクリックします。
   d. 名前項目に **MarketingInformation1** と入力します。
   e. *OK* をクリックします。
9. Interactive SQL で次の文を実行し、新しいテーブルにデータを設定します。

```
INSERT INTO MarketingInformation1
   SELECT * FROM GROUPO.MarketingInformation;
```

10. サンプルデータベースの MarketingInformation1 テーブルの Description カラムに、myTxtConfig テキスト設定オブジェクトを参照するテキストインデックスを作成します。再表示間隔を 24 時間に設定します。

```
CREATE TEXT INDEX myTxtIndex ON MarketingInformation1 ( Description )
   CONFIGURATION myTxtConfig
   AUTO REFRESH EVERY 24 HOURS;
```

11. 次の文を実行して、テキストインデックスを再表示します。

```
REFRESH TEXT INDEX myTxtIndex ON MarketingInformation1;
```

12. 次の文を実行して、テキストインデックスをテストします。
   a. この文により、テキストインデックスで **cotton** または **cap** という単語が検索されます。結果はスコアの降順でソートされます。テキストインデックスで、**cap** という単語が出現する頻度は **cotton** よりも低いため、**cap** のスコアの方が高くなります。

```
SELECT ID, Description, ct.*
   FROM MarketingInformation1
     CONTAINS ( Description, 'cotton | cap' ) ct
   ORDER BY score DESC;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball **Cap**</title></head><body lang=EN-US> A lightweight wool **cap** with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner. </body></html>	2.2742084275032632
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball **Cap**</title></head><body lang=EN-US> This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days. </body></html>	1.6980426550094467

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>	0.9461597363521859
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>	0.9244136988525732

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
906	`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p><span style='font-size:10.0pt;font-family:Arial'>Lightweight 100% organically grown` **cotton** `construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</span></p></body></html>`	0.9134171046194403
909	`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80%` **cotton**`/20% polyester blend makes it easy to keep them clean. </body></html>`	0.8856420222728282

b. 次の文により、テキストインデックスで cotton という単語が検索されます。visor というワードも含むローは破棄されます。CONTAINS 句では述部が使用されるため、結果は記録されません。

```
SELECT ID, Description
   FROM MarketingInformation1
   WHERE CONTAINS( Description, 'cotton -visor' );
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>

c. 次の文は、`cotton` という単語について各ローをテストします。ローにこの単語が含まれている場合は Results カラムに 1 が表示され、含まれていない場合は 0 が返されます。

```
SELECT ID, Description, IF CONTAINS ( Description, 'cotton' )
      THEN 1
      ELSE 0
      ENDIF AS Results
   FROM MarketingInformation1;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Results
901	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+. </body></html>	0
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin. </body></html>	0

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Results
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin. </body></html>	0
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days. </body></html>	0

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Results
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner. </body></html>	0
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>	1

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Results
907	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it. </body></html>	0
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>	1

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Results
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>	1
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>	1

13. Interactive SQL と SQL Central を閉じます。

結果

GENERIC テキストインデックスの全文検索を実行しました。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

(省略可) サンプルデータベース (*demo.db*) を元の状態にリストアします。

関連情報

テキスト設定オブジェクトの概念と参照 [279 ページ]
テキストインデックスの概念と参照 [292 ページ]

1.3.2.7 チュートリアル: あいまい全文検索の実行

NGRAM 単語区切りを使用するテキストインデックスに対して、あいまい全文検索を実行します。

前提条件

CREATE TEXT CONFIGURATION と CREATE TABLE のシステム権限が必要です。さらに、SELECT ANY TABLE システム権限、または、テーブル MarketingInformation に対する SELECT 権限も必要です。

手順

1. Interactive SQL を起動します。 スタート > プログラム > *SQL Anywhere17* > 管理ツール > *Interactive SQL* をクリックします。
2. 接続ウィンドウで、次の項目に入力します。
   a. 認証ドロップダウンリストで データベースをクリックします。
   b. ユーザ *ID* 項目に、**DBA** と入力します。
   c. パスワード項目に、**sql** と入力します。
   d. アクションドロップダウンリストで、*ODBC* データソースを使用した接続を選択します。
   e. SQL Anywhere 17 Demo データソースを選択し、接続をクリックします。
3. 次の文を実行して、myFuzzyTextConfig というテキスト設定オブジェクトを作成します。FROM 句を含めて、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を指定する必要があります。

```
CREATE TEXT CONFIGURATION myFuzzyTextConfig FROM default_char;
```

4. 次の文を実行して、単語区切りを NGRAM に変更し、単語の最大長を 3 に設定します。N-gram を使用してあいまい検索を実行します。

```
ALTER TEXT CONFIGURATION myFuzzyTextConfig
   TERM BREAKER NGRAM;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ALTER TEXT CONFIGURATION myFuzzyTextConfig
   MAXIMUM TERM LENGTH 3;
```

5. SQL Central を起動します。 [スタート] > [プログラム] > SQL Anywhere17 > [管理ツール] > SQL Central をクリックします。
6. 接続 > SQL Anywhere17 に接続 をクリックします。
7. 接続ウィンドウで、次の項目に入力します。
   a. 認証ドロップダウンリストで データベースをクリックします。
   b. ユーザ ID 項目に、**DBA** と入力します。
   c. パスワード項目に、**sql** と入力します。
   d. アクションドロップダウンリストで、ODBC データソースを使用した接続を選択します。
   e. SQL Anywhere 17 Demo データソースを選択し、OK をクリックします。
8. MarketingInformation テーブルのコピーを作成します。
   a. SQL Central で、テーブルフォルダを展開します。
   b. MarketingInformation を右クリックし、コピーをクリックします。
   c. テーブルフォルダを右クリックし、貼り付けをクリックします。
   d. 名前項目に **MarketingInformation2** と入力します。OK をクリックします。
9. Interactive SQL で、次の文を実行して、MarketingInformation2 テーブルにデータを追加します。

```
INSERT INTO MarketingInformation2
   SELECT * FROM GROUPO.MarketingInformation;
```

10. 次の文を実行して、MarketingInformation2 テーブルの Description カラムに、myFuzzyTextConfig テキスト設定オブジェクトを参照するテキストインデックスを作成します。

```
CREATE TEXT INDEX myFuzzyTextIdx ON MarketingInformation2 ( Description )
   CONFIGURATION myFuzzyTextConfig;
```

11. 次の文を実行して、**coten**: に似た単語を検索します。

```
SELECT MarketingInformation2.Description, ct.*
    FROM MarketingInformation2 CONTAINS ( MarketingInformation2.Description,
'FUZZY "coten"' ) ct
    ORDER BY ct.score DESC;
```

説明	スコア
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>	0.9461597363521859

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

説明	スコア
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>	0.9244136988525732
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>	0.9134171046194403
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>	0.8856420222728282
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days. </body></html>	0

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

説明	スコア
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner. </body></html>	0
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+. </body></html>	0
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin. </body></html>	0
<html><head><meta http-equiv=**Content**-Type **content**="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin. </body></html>	0

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

説明	スコア
`<html><head><meta http-equiv=`**Content**`-Type` **content**`="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it. </body></html>`	0

> **注記**
>
> 最後の 6 つのローに、一致する N-gram を含む単語があります。ただし、これらの単語はテーブル内のすべてのローに含まれているため、スコアは割り当てられていません。

12. Interactive SQL と SQL Central を閉じます。

結果

あいまい全文検索を実行しました。

次のステップ

(省略可) サンプルデータベース (*demo.db*) を元の状態にリストアします。

関連情報

全文あいまい検索 [275 ページ]

テキスト設定オブジェクトの概念と参照 [279 ページ]

テキストインデックスの概念と参照 [292 ページ]

全文検索結果のスコア [276 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.8 チュートリアル: NGRAM テキストインデックスに対する非あいまい全文検索の実行

NGRAM 単語区切りを使用するテキストインデックスに対して、非あいまい全文検索を実行します。中国語、日本語、韓国語データの全文検索を作成する際もこの手順で実行できます。

前提条件

CREATE TEXT CONFIGURATION と CREATE TABLE のシステム権限が必要です。さらに、SELECT ANY TABLE システム権限、または、テーブル MarketingInformation に対する SELECT 権限も必要です。

コンテキスト

マルチバイト文字セットを含むデータベースでは、全角カンマや全角スペースなどの一部の句読表記文字やスペース文字は、英数字として処理される場合があります。

手順

1. Interactive SQL を起動します。スタート > プログラム > *SQL Anywhere17* > 管理ツール > *Interactive SQL* をクリックします。
2. 接続ウィンドウで、次の項目に入力します。
   a. 認証ドロップダウンリストで データベースをクリックします。
   b. ユーザ *ID* 項目に、**DBA** と入力します。
   c. パスワード項目に、**sql** と入力します。
   d. アクションドロップダウンリストで、*ODBC* データソースを使用した接続を選択します。
   e. SQL Anywhere 17 Demo データソースを選択し、*OK* をクリックします。
   f. 接続をクリックします。
3. 次の文を実行して、myNcharNGRAMTextConfig という NCHAR テキスト設定オブジェクトを作成します。

```
CREATE TEXT CONFIGURATION myNcharNGRAMTextConfig FROM default_nchar;
```

4. 次の文を実行して、TERM BREAKER アルゴリズムを NGRAM に変更し、MAXIMUM TERM LENGTH を 2 に設定します。

```
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig
   TERM BREAKER NGRAM;
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig
   MAXIMUM TERM LENGTH 2;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

中国語、日本語、韓国語のデータについては、N の値を 2 または 3 とすることをお奨めします。検索では 1 文字または 2 文字に制限されているため、N 値を 1 に設定します。N 値を 1 に設定することで、長いクエリの実行に時間がかかる場合があります。

5. SQL Central を起動します。 *[スタート]* › *[プログラム]* › *SQL Anywhere17* › *[管理ツール]* › *SQL Central* をクリックします。
6. 接続 › *SQL Anywhere17* に接続 をクリックします。
7. 接続ウィンドウで、次の項目に入力します。
   a. 認証ドロップダウンリストで データベースをクリックします。
   b. ユーザ *ID* 項目に、**DBA** と入力します。
   c. パスワード項目に、**sql** と入力します。
   d. アクションドロップダウンリストで、*ODBC* データソースを使用した接続を選択します。
   e. SQL Anywhere 17 Demo データソースを選択し、*OK* をクリックします。
   f. 接続をクリックします。
8. MarketingInformation テーブルのコピーを作成します。
   a. テーブルフォルダを展開します。
   b. *MarketingInformation* を右クリックし、コピーをクリックします。
   c. テーブルフォルダを右クリックし、貼り付けをクリックします。
   d. 名前項目に **MarketingInformationNgram** と入力します。
   e. *OK* をクリックします。
9. Interactive SQL で、次の文を実行して、MarketingInformationNgram テーブルにデータを追加します。

```
INSERT INTO MarketingInformationNgram
   SELECT * FROM GROUPO.MarketingInformation;
COMMIT;
```

10. 次の文を実行して、myNcharNGRAMTextConfig テキスト設定オブジェクトを使用することで、MarketingInformationNgram テーブルの Description カラムに IMMEDIATE REFRESH テキストインデックスを作成します。

```
CREATE TEXT INDEX ncharNGRAMTextIndex
   ON MarketingInformationNgram( Description )
      CONFIGURATION myNcharNGRAMTextConfig;
```

11. テキストインデックスをテストします。
   a. 次の文により、2-GRAM のテキストインデックスで **sw** を含む単語が検索されます。結果はスコアの降順でソートされます。

```
SELECT M.Description, ct.*
   FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'sw' ) ct
   ORDER BY ct.score DESC;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Description	Score
`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>`**Sw**`eatshirt</title></head><body lang=EN-US><p><span style='font-size:10.0pt;font-family:Arial'>Lightweight 100% organically grown cotton hooded `**Sw**`eatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</span></p></body></html>`	2.262071918398649
`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>`**Sw**`eatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean. </body></html>`	1.5556043490424176

b. 次の文は、**ams** を含む単語を検索します。結果はスコアの降順でソートされます。

```
SELECT M.Description, ct.*
   FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'ams' ) ct
   ORDER BY ct.score DESC;
```

2-GRAM テキストインデックスでは、前述の文はセマンティック上、次の文と同義になります。

```
SELECT M.Description, ct.*
   FROM MarketingInformationNgram AS M
CONTAINS( M.Description, '"am ms"' ) ct
   ORDER BY ct.score DESC;
```

どちらの文も次の結果を返します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown cotton hooded sweatshirt with taped neck se**ams**. Comes pre-washed for softness and to lessen shrinkage. </body></html>	1.6619019465461564
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched se**ams** for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean. </body></html>	1.5556043490424176

c. 次の文は、v の後に任意の英数字が続く単語を検索します。インデックス付けされたデータで、ve の出現する頻度の方が高いため、2-GRAM の ve を含むローのスコアは、vi を含むローよりも低くなります。

```
SELECT M.ID, M.Description, ct.*
   FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'v*' ) ct
   ORDER BY ct.score DESC;
```

結果はスコアの降順でソートされます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
901	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> We'**ve** impro**ve**d the design of this perennial fa**vo**rite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+. </body></html>	3.3416789108071976
907	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>**Vi**sor</title></head><body lang=EN-US> A polycarbonate **vi**sor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to gi**ve** you a snug yet comfortable fit e**ve**ry time you wear it. </body></html>	2.1123084896159376

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> A lightweight wool cap with mesh side **ve**nts for breathable comfort during aerobic acti**vi**ties. Moisture-absorbing headband liner. </body></html>	1.6750365447462499
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying cotton shorts pro**vi**de all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>	0.9244136988525732

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>**Vi**sor</title></head><body lang=EN-US> Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>	0.9134171046194403
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> This fashionable hat is ideal for glacier tra**ve**l, sea-kayaking, and hiking. With concealed draw cord for windy days. </body></html>	0.7313071661212746

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
903	`<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p><span style='font-size:10.0pt;font-family:Arial'>A sporty, casual shirt made of recycled water bottles. It will ser`**ve**` you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</span></p></body></html>`	0.6799436746197272

d. 次の文は、各ローで v を含む任意の単語を検索します。2 番目の文の後、変数には文字列 `av OR ev OR iv OR ov OR rv OR ve OR vi OR vo` が含まれます。結果はスコアの降順でソートされます。インデックス付けされたすべてのローに N-gram があると、スコアは 0 になります。

   これは、ホワイトスペースや英数字以外の文字の前にある 1 文字を検索するための唯一の方法です。

```
CREATE VARIABLE query NVARCHAR (100);
SELECT LIST (term, ' OR ' )
INTO query
   FROM sa_text_index_vocab_nchar( 'ncharNGRAMTextIndex',
'MarketingInformationNgram', 'dba' )
   WHERE term LIKE '%v%';
SELECT M.ID, M.Description, ct.*
   FROM MarketingInformationNgram AS M
   CONTAINS( M.Description, query ) ct
   ORDER BY ct.score DESC;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
901	<html><head><meta http-equ**iv**=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> We'**ve** impr**ove**d the design of this perennial f**avo**rite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+. </body></html>	6.654350268810443
907	<html><head><meta http-equ**iv**=Content-Type content="text/html; charset=windows-1252"><title>**Vi**sor</title></head><body lang=EN-US> A polycarbonate **vi**sor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to g**ive** you a snug yet comfortable fit **eve**ry time you wear it. </body></html>	4.265623837817126

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin. </body></html>	2.9386676702799504
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>	2.5481193655722336

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
904	<html><head><meta http-equ**iv**=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> This fashionable hat is ideal for glacier tr**ave**l, sea-kayaking, and hiking. With concealed draw cord for windy days. </body></html>	2.4293498211307214
905	<html><head><meta http-equ**iv**=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> A lightweight wool cap with mesh side **ve**nts for breathable comfort during aerobic act**ivi**ties. Moisture-absorbing headband liner. </body></html>	1.6750365447462499

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>	0.9134171046194403
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin. </body></html>	0

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>	0
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean. </body></html>	0

e. 次の文は、Description カラムで ea、ka、ki を含むローを検索します。

```
SELECT M.ID, M.Description, ct.*
   FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'ea ka ki' ) ct
  ORDER BY ct.score DESC;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	Description	Score
904	<html><h**ea**d><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></h**ea**d><body lang=EN-US> This fashionable hat is id**ea**l for glacier travel, s**ea**-**ka**yaking, and hi**ki**ng. With conc**ea**led draw cord for windy days. </body></html>	3.4151032739119733

f. 次の文は、Description カラムで ve と vi を含み、gg は含まないローを検索します。

```
SELECT M.ID, M.Description, ct.*
   FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 've & vi -gg' ) ct
   ORDER BY ct.score DESC;
```

ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US> A lightweight wool cap with mesh side **ve**nts for breathable comfort during aerobic acti**vi**ties. Moisture-absorbing headband liner. </body></html>	1.6750365447462499

12. Interactive SQL と SQL Central を閉じます。

結果

NGRAM テキストインデックスの全文検索を実行しました。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

(省略可) サンプルデータベース (*demo.db*) を元の状態にリストアします。

関連情報

テキスト設定オブジェクトの概念と参照 [279 ページ]
テキストインデックスの概念と参照 [292 ページ]

1.3.2.9 高度: 全文検索での単語の削除

テキストインデックスは、テキストインデックスの作成に使用されるテキスト設定オブジェクトに定義された設定に従って、構築されます。

次の条件のうち 1 つ以上が当てはまる場合、単語はテキストインデックスに含まれません。

- 単語がストップリストに含まれます
- 単語が単語の最小長より短いです (GENERIC のみ)
- 単語が単語の最大長より長いです

同じ規則はクエリ文字列にも適用されます。削除された単語は、フレーズの末尾または先頭で 0 個以上の単語と一致できます。たとえば、`'the'` という単語がストップリストに含まれているとします。

- この単語が AND、OR、または NEAR のどちらかの側にある場合、演算子と単語の両方が削除されます。たとえば、`'the AND apple'`、`'the OR apple'`、`'the NEAR apple'` を検索することは、`'apple'` を検索することと同義になります。
- この単語が AND NOT の右側にある場合、AND NOT と単語の両方が削除されます。たとえば、`'apple AND NOT the'` を検索することは、`'apple'` を検索することと同義になります。
  単語が AND NOT の左側にある場合は、式全体が削除され、ローは返されません。例: `'orange and the AND NOT apple' = 'orange'`
- 単語がフレーズ内にある場合、そのフレーズは削除される単語の位置において、任意の単語と一致させることができます。たとえば、`'feed the dog'` の検索は、`'feed the dog'`、`'feed my dog'`、`'feed any dog'` などに一致します。

検索している単語がすべてテキストインデックスに含まれていない場合は、ローが返されません。たとえば、`'the'` と `'a'` の両方がストップリストにあるとします。`'a OR the'` の検索では、ローが返されません。

関連情報

テキスト設定オブジェクトの概念と参照 [279 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.10 高度: 外部単語区切りライブラリとプレフィルタライブラリ

独自の外部単語区切りライブラリとプレフィルタライブラリを作成して使用できます。

このセクションの内容:

外部単語区切りまたはプレフィルタライブラリを使用する理由 [326 ページ]

外部単語区切りとプレフィルタライブラリを使用すると、インデックス付けされる前のデータに対して独自の外部単語区切りとプレフィルタを実行できます。

全文パイプラインのワークフロー [327 ページ]

テキストインデックスを作成し、更新し、問い合わせるワークフローは、パイプラインと呼ばれます。

外部プレフィルタライブラリ [328 ページ]

外部プレフィルタライブラリを作成して使用できます。

外部単語区切りライブラリ [331 ページ]

外部単語区切りライブラリを作成して使用できます。

1.3.2.10.1 外部単語区切りまたはプレフィルタライブラリを使用する理由

外部単語区切りとプレフィルタライブラリを使用すると、インデックス付けされる前のデータに対して独自の外部単語区切りとプレフィルタを実行できます。

たとえば、XML 値を含むカラムにテキストインデックスを作成するとします。プレフィルタを使用すると、XML タグが内容でインデックス付けされないように該当の XML タグを除外できます。

テキストインデックスが作成されると、各ドキュメントはテキストインデックスのテキスト設定で指定された、組み込みの単語区切りによって処理され、ドキュメントに含まれている単語、およびドキュメント内の単語の位置が確認されます。

SQL Anywhere での全文検索はテキストインデックスを使用して実行されます。テキストインデックスが構築されたカラムの各値は、ドキュメントと呼ばれます。テキストインデックスが作成されると、各ドキュメントはテキストインデックスのテキスト設定で指定された、組み込みの単語区切りによって処理され、ドキュメントに含まれている単語 (トークンとも呼ばれます) およびドキュメント内の単語の位置が確認されます。組み込みの単語区切りは、クエリ文字列のドキュメント (テキストコンポーネント) に対する単語区切りの実行にも使用されます。たとえば、クエリ文字列 'rain or shine' は、OR 演算子で連結された 2 つのドキュメント 'rain' と 'shine 'で構成されています。テキスト設定で指定された組み込みの単語区切りアルゴリズムは、ストップリストを単語に分割したり、sa_char_terms system システムプロシージャの入力を単語に分割するのにも使用されます。

アプリケーションのニーズによっても異なりますが、組み込みの GENERIC 単語区切りの一部の動作が望ましくないか、または限定的である場合、および NGRAM 単語区切りがアプリケーションのニーズに合っていない場合もあります。たとえば、組み込みの GENERIC 単語区切りでは、言語固有の単語区切りは行われません。カスタムの単語区切りを実装する必要があるその他の理由を次に示します。

言語固有の単語区切りがない

単語の構成内容に関する言語ルールは言語ごとに異なります。このため、単語区切りのルールは言語ごとに異なります。組み込みの単語区切りには、言語固有の単語区切りのルールは用意されていません。

アポストロフィを含む単語の処理

"they'll" という単語は組み込みの GENERIC 単語区切りによって "they ll" として扱われます。ただし、アポストロフィを単語の一部として扱うカスタムの GENERIC 単語区切りを作成できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

単語の置き換えがサポートされていない

単語の置き換えを指定することはできません。たとえば、"they'll" という単語をインデックス付けするとき、they および will の 2 つの単語として保存することができます。同様に、単語の置換えを使用すると、大文字と小文字が区別されるデータベースで大文字と小文字を区別しない検索を実行できます。

全文インデックスの作成と更新を行うときに、カスタムおよびサードパーティのプレフィルタと単語区切りライブラリにアクセスするための API が提供されています。これは、外部ライブラリを使用して XML、PDF、Word のようなドキュメントフォーマットを処理したり、内容にインデックスを付ける前に不要な単語や内容を削除できることを意味します。

独自のライブラリの設計をサポートするために、サンプルのプレフィルタライブラリと単語区切りライブラリが `Samples` ディレクトリに用意されています。または、API を使用してサードパーティのライブラリにアクセスできます。データベースサーバを実行しているシステムに Microsoft Office がインストールされている場合は、Word や Microsoft Excel などの Office ドキュメント用の IFilter を使用できます。サーバに Acrobat Reader がインストールされている場合は、PDF IFilter を使用できます。

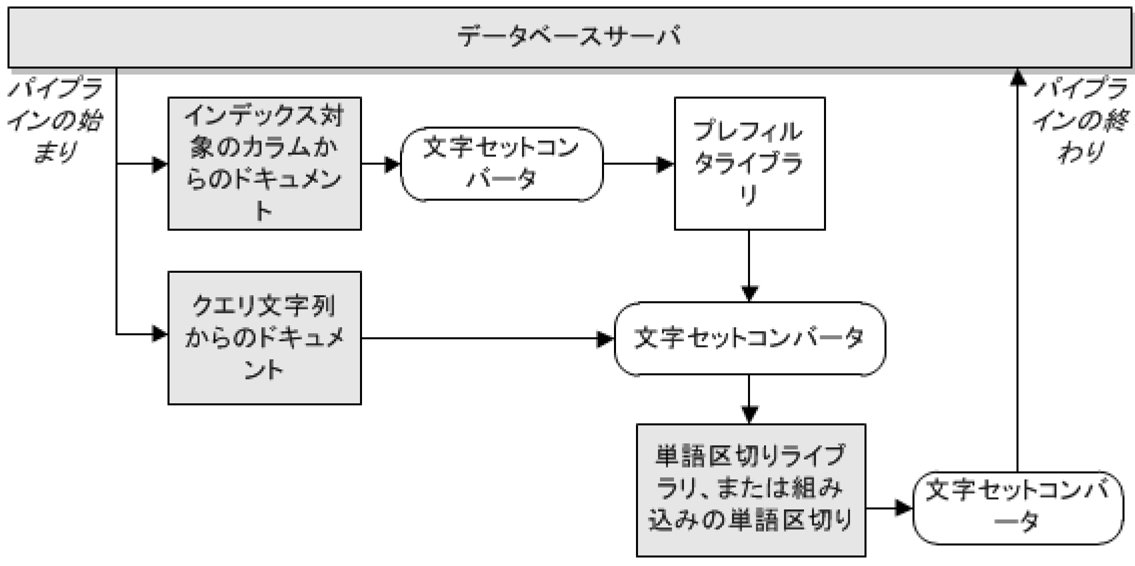
i 注記

NGRAM の外部単語区切りはサポートされていません。

1.3.2.10.2 全文パイプラインのワークフロー

テキストインデックスを作成し、更新し、問い合わせるワークフローは、パイプラインと呼ばれます。

次の図は、データベースサーバ内でインデックス付けするためにドキュメントから単語のストリームへデータが変換される方法を示しています。パイプラインの必須部分は明るいグレーで表されています。矢印はパイプラインでのデータのフローを表しています。関数呼び出しは反対方向に伝達されます。



© 2016 SAP SE or an SAP affiliate company.All rights reserved.

パイプラインの動作方法の概要

1. 各ドキュメントの処理は、データベースサーバがパイプラインの終わり (単語区切りまたは文字セットコンバーターのどちらか) で begin_document メソッドを呼び出すことにより開始されます。パイプライン内の各コンポーネントは、自身の begin_document の起動から戻る前に自身のプロデューサにより begin_document を呼び出します。
2. データベースサーバは begin_document が正常に完了した後、パイプラインの終わりで get_words を呼び出します。
    - get_words の実行中に、単語区切りはプロデューサにより get_next_piece を呼び出し、処理するデータを取得します。パイプラインにプレフィルタが存在する場合、データは get_next_piece 呼び出しの間にフィルタされます。
    - 単語区切りはプロデューサから受け取ったデータを単語区切りのルールに従って単語に分割します。
3. データベースサーバは get_words 呼び出しから返された単語に、ストップリスト制限だけでなく、最小単語長と最大単語長の設定も適用します。
4. データベースサーバは単語が返されなくなるまで get_words を呼び出し続けます。単語が返されなくなった時点で、データベースサーバは end_document を呼び出します。この呼び出しは、パイプライン内で begin_document 呼び出しと同じ方法で伝達されます。

> **注記**
>
> 文字セットコンバータは、必要に応じてデータベースサーバによってパイプラインに透過的に追加されます。

事前フィルタと単語区切りのコードサンプル

SQL Anywhere インストールの `ExternalLibrariesFullText` ディレクトリには、参照用に、プレフィルタと単語区切りのサンプルコードが含まれています。このディレクトリは、`Samples` ディレクトリ内にあります。

関連情報

外部プレフィルタライブラリのワークフロー [330 ページ]
外部単語区切りライブラリを設計する方法 [332 ページ]

1.3.2.10.3 外部プレフィルタライブラリ

外部プレフィルタライブラリを作成して使用できます。

このセクションの内容:

外部プレフィルタを使用するよう SQL Anywhere を設定する方法 [329 ページ]
データが外部プレフィルタライブラリを経由するようにするには、ALTER TEXT CONFIGURATION 文を使用してライブラリとそのエントリポイント関数を指定します。組み込みのプレフィルタアルゴリズムはありません。

外部プレフィルタライブラリを設計する方法 [329 ページ]
プレフィルタライブラリは C/C++ で実装され、

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.10.3.1 外部プレフィルタを使用するよう **SQL Anywhere** を設定する方法

データが外部プレフィルタライブラリを経由するようにするには、ALTER TEXT CONFIGURATION 文を使用してライブラリとそのエントリポイント関数を指定します。組み込みのプレフィルタアルゴリズムはありません。

```
ALTER TEXT CONFIGURATION my_text_config
   PREFILTER EXTERNAL NAME 'my_prefilter@myprefilterLibrary.dll'
```

この例は、my_text_config テキスト設定オブジェクトを使用してテキストインデックスを構築または更新するときに使用するプレフィルタインスタンスを、`myprefilterLibrary.dll` ライブラリの my_prefilter エントリポイント関数を使用して取得するようにデータベースサーバに指示します。

関連情報

a_text_source interface [339 ページ]

1.3.2.10.3.2 外部プレフィルタライブラリを設計する方法

プレフィルタライブラリは C/C++ で実装され、

次の条件を満たしている必要があります。

- プレフィルタインタフェース定義ヘッダファイル `extpfapiv1.h` を含みます。
- a_text_source インタフェースを実装します。
- a_text_source (プレフィルタ) のインスタンスおよびプレフィルタでサポートされている文字セットのラベルを初期化し、返すエントリポイント関数があります。

プレフィルタの呼び出しシーケンス

次の呼び出しシーケンスは、プレフィルタのコンシューマによって、処理されている各ドキュメントに対して実行されます。

```
begin_document(a_text_source*)
get_next_piece(a_text_source*, buffer**, len*)
get_next_piece(a_text_source*, buffer**, len*)
...
end_document(a_text_source*)
```

> **注記**
>
> end_document は、間に begin_document 呼び出しを挟まなくても複数回呼び出すことができます。たとえば、インデックス付けされるドキュメントの 1 つが空の場合、データベースサーバでは、begin_document を呼び出さずにこのドキュメントの end_document を呼び出すことがあります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

get_next_piece 関数はフォーマット情報や受信バイトストリームからのイメージなどの不要なデータを除外し、自己割り付けバッファ内のフィルタされたデータの次のチャンクを返す必要があります。

このセクションの内容:

外部プレフィルタライブラリのワークフロー [330 ページ]
次のフローチャートは、get_next_piece 関数が呼び出されたときの論理フローを示しています。

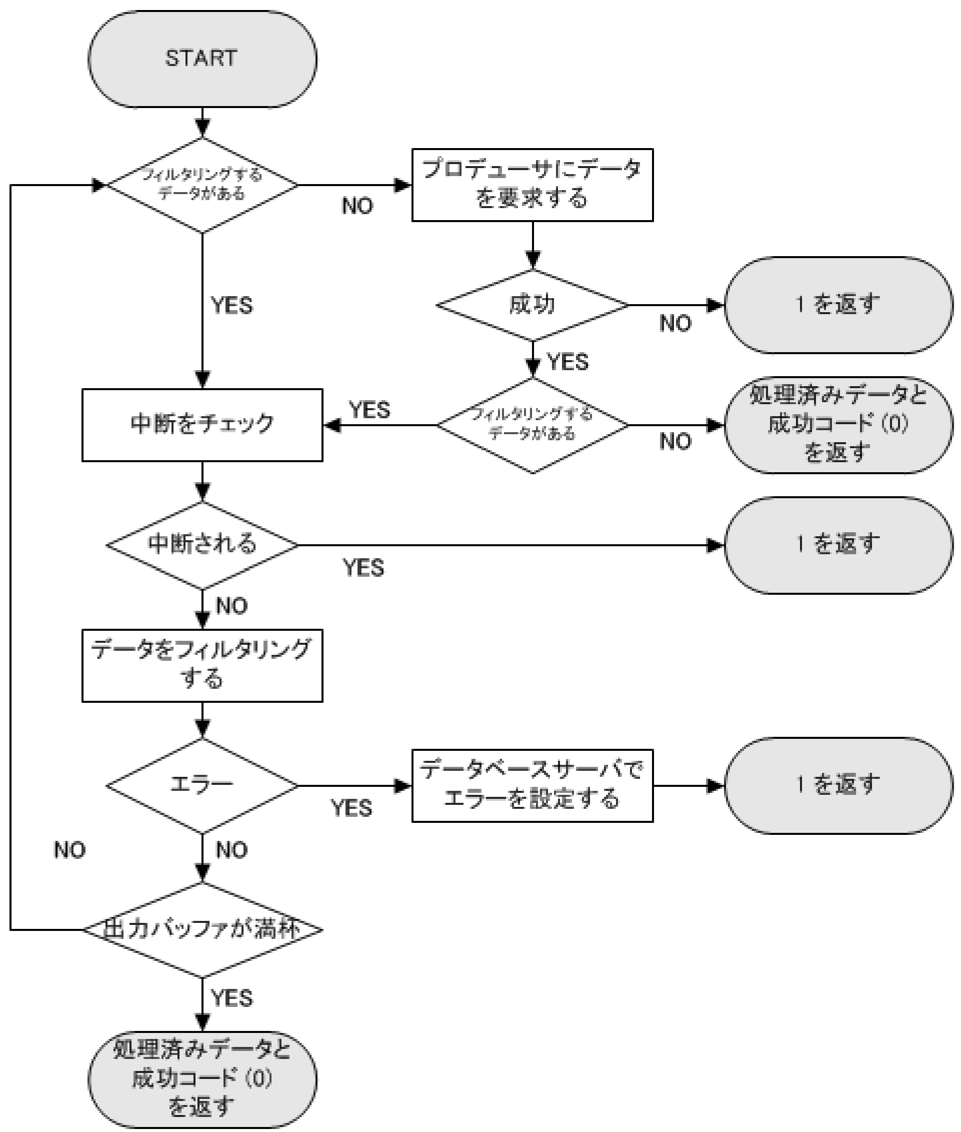
関連情報

全文パイプラインのワークフロー [327 ページ]
a_text_source interface [339 ページ]

1.3.2.10.3.2.1 外部プレフィルタライブラリのワークフロー

次のフローチャートは、get_next_piece 関数が呼び出されたときの論理フローを示しています。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.



関連情報

全文パイプラインのワークフロー [327 ページ]
a_text_source interface [339 ページ]

1.3.2.10.4 外部単語区切りライブラリ

外部単語区切りライブラリを作成して使用できます。

このセクションの内容:

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

外部単語区切りを使用するよう SQL Anywhere を設定する方法 [332 ページ]
デフォルトでは、テキスト設定オブジェクトを作成すると、そのテキスト設定オブジェクトに関連付けられたデータには組み込みの単語区切りが使用されます。

外部単語区切りライブラリを設計する方法 [332 ページ]
外部単語区切りライブラリは C/C++ で実装する必要があります。

1.3.2.10.4.1 外部単語区切りを使用するよう SQL Anywhere を設定する方法

デフォルトでは、テキスト設定オブジェクトを作成すると、そのテキスト設定オブジェクトに関連付けられたデータには組み込みの単語区切りが使用されます。

データが外部単語区切りライブラリを経由するようにするには、次のように ALTER TEXT CONFIGURATION 文を使用してライブラリとそのエントリポイント関数を指定します。

```
ALTER TEXT CONFIGURATION my_text_config
   TERM BREAKER GENERIC EXTERNAL NAME 'my_termbreaker@termbreaker'
```

この例は、my_text_config テキスト設定オブジェクトに関連付けられたテキストインデックスを構築、更新、または問い合わせるとき、テキスト設定オブジェクトのストップリストの解析時、sa_char_terms システムプロシージャへの入力の処理時に使用する単語区切りインスタンスを、単語区切りライブラリの my_termbreaker エントリポイント関数を使用して取得するようにデータベースサーバに指示します。

関連情報

a_word_source インタフェース [344 ページ]

1.3.2.10.4.2 外部単語区切りライブラリを設計する方法

外部単語区切りライブラリは C/C++ で実装する必要があります。

また、外部単語区切りライブラリは次の条件を満たす必要があります。

- 単語区切りインタフェース定義ヘッダファイル `exttbapiv1.h` をインクルードします。
- a_word_source インタフェースを実装します。
- a_word_source (単語区切り) のインスタンス、および単語区切りでサポートされている文字セットのラベルを初期化して返すエントリポイント関数があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

単語区切りの呼び出しシーケンス

次の呼び出しシーケンスは、単語区切りのコンシューマによって、処理されている各ドキュメントに対して実行されます。

```
begin_document(a_word_source*, asql_uint32);
get_words(a_word_source*, a_term**, uint32 *num_words)
get_words(a_word_source*, a_term**, uint32 *num_words)
...
end_document(a_word_source*)
```

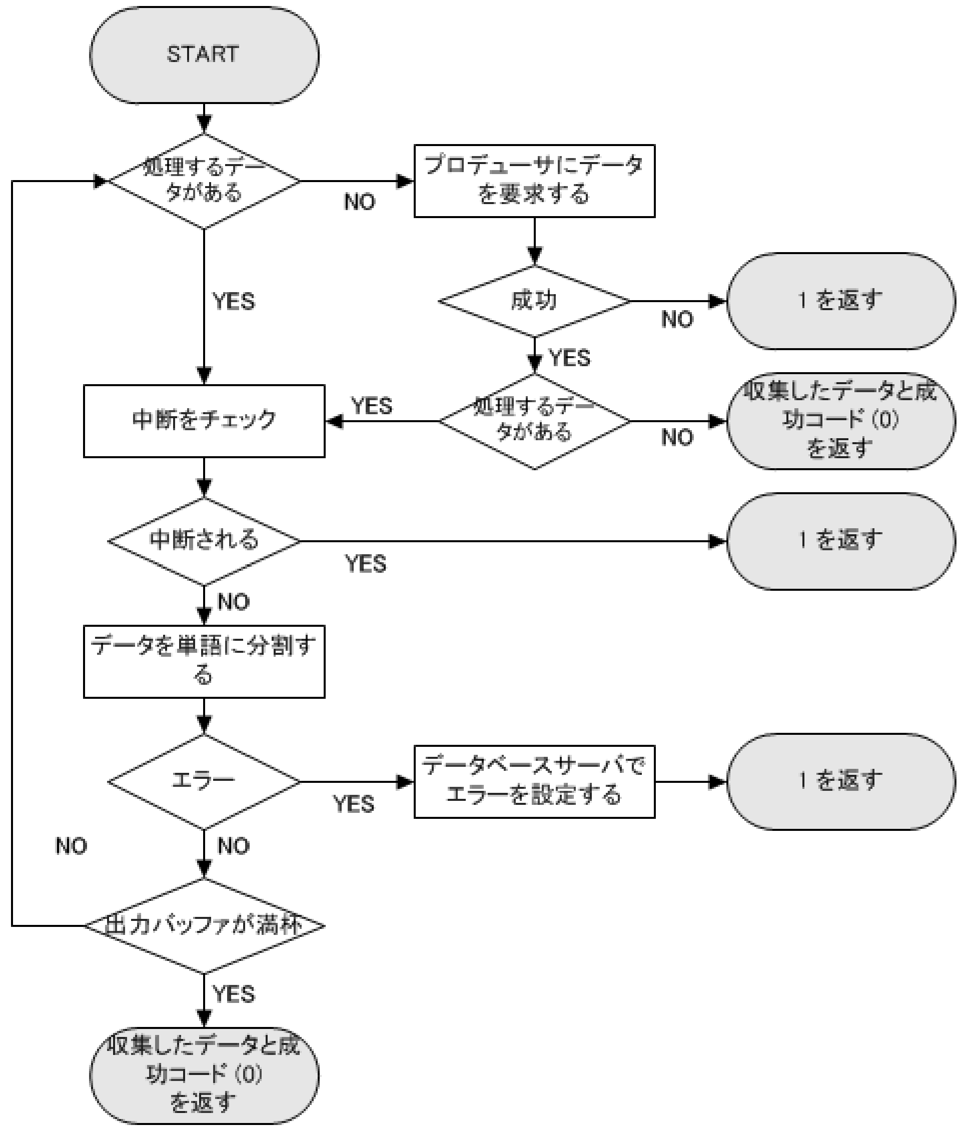
get_words 関数は、a_term 構造体の配列がデータで満たされるか、または処理するデータがなくなるまで、データが単語に分割されるように、プロデューサで get_next_piece を呼び出す必要があります。

> **注記**
>
> end_document は、間に begin_document 呼び出しを挟まなくても複数回呼び出すことができます。たとえば、インデックス付けされるドキュメントの 1 つが空の場合、データベースサーバでは、begin_document を呼び出さずにこのドキュメントの end_document を呼び出すことがあります。

外部単語区切りライブラリのワークフロー

次のフローチャートは、get_words 関数が呼び出されたときの論理フローを示しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.



関連情報

全文パイプラインのワークフロー [327 ページ]

a_word_source インタフェース [344 ページ]

a_text_source interface [339 ページ]

a_term 構造体 [347 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.11 高度: 外部全文ライブラリの API

テキストインデックスで事前フィルタまたは単語区切りの外部ライブラリを作成し使用するには、次の手順を実行します。

- SQL Anywhere C/C++ インタフェースを実装します。
- 上記の手順で作成したコードをコンパイルおよびリンクして、動的にロードできるライブラリを作成します。
- データベースにテキスト設定オブジェクトを作成し、事前フィルタか単語区切りまたはその両方の外部ライブラリ内のエントリポイント関数を指定するように変更します。
  エントリポイント関数は、基本となるドキュメント (カラム値) が変更されるときテキストインデックスの挿入/削除中に使用される事前フィルタおよび単語区切りのオブジェクトを取得するために使用されます。外部単語区切りライブラリの場合、エントリポイント関数は、語区切りを使用するテキストインデックスに対するクエリの解析にも使用されます。

このセクションの内容:

a_server_context 構造体 [336 ページ]
: 複数のコールバックがデータベースサーバによってサポートされ、エラーレポート、中断処理およびメッセージロギングを実行するために a_server_context 構造体を介して全文外部ライブラリに送られます。

a_init_pre_filter 構造体 [338 ページ]
: a_init_pre_filter 構造体は、外部プレフィルタエントリポイント関数のインスタンスの入力要件および出力要件をネゴシエートするために使用されます。

a_text_source interface [339 ページ]
: 外部事前フィルタライブラリは、全文インデックスのデータ設定または更新のためにドキュメントの事前フィルタリングを実行する場合に a_text_source インタフェースを実装する必要があります。

a_init_term_breaker 構造体 [342 ページ]
: a_init_term_breaker 構造体は、外部単語区切りのインスタンスの入力要件および出力要件をネゴシエートするために使用されます。

a_term_breaker_for 列挙体 [343 ページ]
: パイプラインをテキストインデックスの更新中に使用するために構築するか、テキストインデックスの問い合わせに使用するために構築するかを指定する場合に、a_term_breaker_for 列挙体を使用します。

a_word_source インタフェース [344 ページ]
: 外部単語区切りライブラリは、テキストインデックス操作用に単語区切りを実行するための a_word_source インタフェースを実装する必要があります。

a_term 構造体 [347 ページ]
: a_term 構造体は、単語、単語の長さ、単語の位置を保存します。

extpf_use_new_api エントリポイント関数 (事前フィルタ) [348 ページ]
: extpf_use_new_api エントリポイント関数は、データベースサーバに、外部事前フィルタライブラリに実装されているインタフェースバージョンについて通知します。

exttb_use_new_api エントリポイント関数 (単語区切り) [348 ページ]
: exttb_use_new_api エントリポイント関数は、外部単語区切りライブラリに実装されているインタフェースバージョンについての情報を提供します。

extfn_post_load_library グローバルエントリポイント関数 [349 ページ]
: extfn_post_load_library グローバルエントリポイント関数は、ライブラリ内のいずれかの関数が呼び出される前にライブラリ全体にわたる設定を行うライブラリ固有の要件がある場合に必要です。

extfn_pre_unload_library グローバルエントリポイント関数 [349 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

extfn_pre_unload_library グローバルエントリポイント関数は、ライブラリがアンロードされる前にライブラリ全体にわたるクリーンアップを行うライブラリ固有の要件がある場合にのみ必要です。

プレフィルタエントリポイント関数 [350 ページ]
プレフィルタエントリポイント関数は、外部プレフィルタのインスタンスを初期化し、データの文字セットをネゴシエートします。

単語区切りエントリポイント関数 [351 ページ]
単語区切りエントリポイント関数は、外部単語区切りのインスタンスを初期化し、データの文字セットをネゴシエートします。

1.3.2.11.1 a_server_context 構造体

複数のコールバックがデータベースサーバによってサポートされ、エラーレポート、中断処理およびメッセージロギングを実行するために a_server_context 構造体を介して全文外部ライブラリに送られます。

構文

```
typedef struct a_server_context {
    void (SQL_CALLBACK *set_error)( a_server_context  *this
                                   , a_sql_uint32     error_code
                                   , const char       *error_string
                                   , short            error_string_length
                                  );
    a_sql_uint32 (SQL_CALLBACK *get_is_cancelled)( a_server_context *this
                                    );
    void (SQL_CALLBACK *log_message)( a_server_context *this
                                    , const char       *message_string
                                    , short            message_string_length
                                    );
    void *_context;
} a_server_context, *p_server_context;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

メンバー

メンバー名	タイプ	説明
set_error	void	このメソッドを使用すると、外部事前フィルタおよび単語区切りによって、エラーコードとエラー文字列を指定してデータベースサーバにエラーを設定できます。データベースサーバは現在の操作をロールバックし、次の形式でユーザにエラーコードと文字列を返します。 `"Error from external library: -<error_code>: <error_string>"` error_code には 17000 よりも大きい正の整数を指定します。 error_string には NULL で終了する文字列を指定します。 str_len には error_string の長さをバイト単位で指定します。
get_is_cancelled	a_sql_uint32	外部事前フィルタおよび単語区切りでこのメソッドを定期的に呼び出して、現在の操作が中断されているかどうかをチェックする必要があります。このメソッドでは、現在の操作が中断された場合は 1 が返され、中断されていない場合は 0 が返されます。1 が返された場合、呼び出し元は以降の処理を停止し、すぐに戻る必要があります。
log_message	void	このメソッドを使用すると、外部事前フィルタおよび単語区切りによって、データベースサーバログにメッセージを記録できます。 message には NULL で終了する文字列を指定します。 msg_len にはメッセージの長さをバイト単位で指定します。
_context	void	内部で使用されます。データベースサーバコンテキストへのポインタ。

備考

a_server_context 構造体 は、SQL Anywhere インストールディレクトリの SDK¥Include サブディレクトリのヘッダファイル exttxtcmn.h によって定義されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

a_server_context 構造体によって公開されるメソッドを呼び出す場合は、オペレーティングシステムの同期プリミティブを外部ライブラリで使用しないようにしてください。

1.3.2.11.2 a_init_pre_filter 構造体

a_init_pre_filter 構造体は、外部プレフィルタエントリポイント関数のインスタンスの入力要件および出力要件をネゴシエートするために使用されます。

この構造体は、プレフィルタエントリポイント関数にパラメータとして渡されます。

構文

```
typedef struct a_init_pre_filter {
   a_text_source           *in_text_source;    /* IN */
   a_text_source           *out_text_source;   /* OUT */
   const char              *desired_charset;   /* IN */
   char                    *actual_charset;    /* OUT */
   short                   is_binary;          /* IN */
} a_init_pre_filter;
```

メンバー

名前	タイプ	説明
in_text_source	`a_text_source *`	作成する外部事前フィルタ (a_text_source オブジェクト) のプロデューサへのポインタ。事前フィルタエントリポイント関数の呼び出し元によって指定されます。
out_text_source	`a_text_source *`	事前フィルタエントリポイント関数によって指定される外部事前フィルタ (a_text_source オブジェクト) へのポインタ。
desired_charset	`const char *`	エントリポイント関数の呼び出し元が、事前フィルタの出力があると期待する文字セット。is_binary フラグが 0 の場合、特にネゴシエートされないかぎり、事前フィルタへの入力の文字セットでもあります。
actual_charset	`char *`	外部事前フィルタライブラリが出力を生成する、(外部ライブラリによってネゴシエーションの一部として指定された) 文字セット。is_binary が 0 の場合、事前フィルタへの入力の実際の文字セットでもあります。可能な限り、ライブラリで desired_charset のデータの受け入れと作成を実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

名前	タイプ	説明
is_binary	short	入力データがバイナリ (1) であるか、desired_charset (0) であるかを示します。データがバイナリの場合、データベースサーバはパイプラインでのプレフィルタの前に文字セットの変換を通知しません。

備考

a_init_pre_filter 構造体は、SQL Anywhere インストールディレクトリの SDK¥Include サブディレクトリのヘッダファイル extpfapiv1.h によって定義されます。

関連情報

a_text_source interface [339 ページ]
a_word_source インタフェース [344 ページ]
プレフィルタエントリポイント関数 [350 ページ]

1.3.2.11.3 a_text_source interface

外部事前フィルタライブラリは、全文インデックスのデータ設定または更新のためにドキュメントの事前フィルタリングを実行する場合に a_text_source インタフェースを実装する必要があります。

構文

```
typedef struct a_text_source {
   a_sql_uint32 ( SQL_CALLBACK *begin_document )( a_text_source *This );
   a_sql_uint32 ( SQL_CALLBACK *get_next_piece )(
                                  a_text_source *This
                                  , unsigned char ** buffer
                                  , a_sql_uint32* len );
   a_sql_uint32 ( SQL_CALLBACK *end_document )( a_text_source *This);
   a_sql_uint64 ( SQL_CALLBACK *get_document_size )( a_text_source *This );
   a_sql_uint32 ( SQL_CALLBACK *fini_all )( a_text_source *This );
   a_server_context      *_context;
   // Only one of the following two members can have a valid pointer in a given
implementation.
   // These members point to the current module's producer
   a_text_source         *_my_text_producer;
   a_word_source         *_my_word_producer;
   // Following members have been reserved for
   // future use ONLY
   a_text_source         *_my_text_consumer;
   a_word_source         *_my_word_consumer;
} a_text_source, *p_text_source;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

メンバー

メンバー	タイプ	説明
begin_document	a_sql_uint32	ドキュメントの処理に必要なセットアップ手順を実行します。このメソッドは、成功の場合は 0 を返し、エラーが発生したか、使用可能なドキュメントがなくなった場合は 1 を返します。
get_next_piece	a_sql_uint32	フィルタされた入力バイトストリームのフラグメントをフラグメントの長さとともに返します。このメソッドは、指定されたドキュメントに対して複数回呼び出され、ドキュメントのすべての入力データが消費されるか、エラーが発生するまで、各呼び出しでドキュメントの後続のチャンクを返す必要があります。 バッファは、生成されたデータを指すように事前フィルタによって入力される OUT パラメータです。メモリは事前フィルタによって管理されます。 len は、生成されたデータの長さを示す OUT パラメータです。
end_document	a_sql_uint32	指定されたドキュメントに対するフィルタリングの完了を示し、必要に応じてドキュメント固有のクリーンアップを実行します。
get_document_size	a_sql_uint64	事前フィルタによって生成されたドキュメントの合計長 (バイト単位) を返します。a_text_source オブジェクトは、現在のドキュメントのこれまでに生成されたバイトの合計数の現在の数を保持する必要があります。
fini_all	a_sql_uint32	すべてのドキュメントの処理が完了し、パイプラインが閉じられようとしているときにデータベースサーバによって呼び出されます。fini_all では最後のクリーンアップ手順が実行されます。
_context	a_server_context *	このメンバーを使用して、a_init_pre_filter 構造体内のエントリポイント関数に提供されたデータベースサーバのコンテキストを保持します。事前フィルタモジュールではこのコンテキストを使用してデータベースサーバとの直接通信を確立します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

メンバー	タイプ	説明
_my_text_producer	a_text_source *	このメンバーを使用して、a_init_pre_filter 構造体内のエントリポイント関数に提供された事前フィルタの a_text_source プロデューサへのポインタを保存します。このポインタは、文字セット変換が必要な場合は、エントリポイント関数が実行された後に、データベースサーバによって置き換えられる可能性があります。このため、テキストプロデューサへのこのポインタのみが事前フィルタによって使用されます。
_my_word_producer	a_word_source *	今後の使用のために予約されており、NULL に初期化する必要があります。
_my_text_consumer	a_text_source *	今後の使用のために予約されており、NULL に初期化する必要があります。
_my_word_consumer	a_word_source *	今後の使用のために予約されており、NULL に初期化する必要があります。

備考

a_text_source インタフェースはストリームベースのデータです。データはプロデューサから順番に引き出され、各バイトは 1 回だけ表示されます。

a_text_source インタフェースは、SQL Anywhere インストールディレクトリの `SDK¥Include` サブディレクトリのヘッダファイル `extpfapiv1.h` によって定義されます。

複数の関数の呼び出しにまたがるオペレーティングシステムの同期プリミティブは、外部ライブラリに保持しないようにしてください。

関連情報

a_server_context 構造体 [336 ページ]
a_init_pre_filter 構造体 [338 ページ]
プレフィルタエントリポイント関数 [350 ページ]

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

1.3.2.11.4 a_init_term_breaker 構造体

a_init_term_breaker 構造体は、外部単語区切りのインスタンスの入力要件および出力要件をネゴシエートするために使用されます。

この構造体は、単語区切りエントリポイント関数にパラメータとして渡されます。

構文

```
typedef struct a_init_term_breaker
{
    a_text_source       *in_text_source;
    const char          *desired_charset;
    a_word_source       *out_word_source;
    char                *actual_charset;
    a_term_breaker_for  term_breaker_for;
} a_init_term_breaker, *p_init_term_breaker;
```

メンバー

メンバー	タイプ	説明
in_text_source	`a_text_source *`	作成する外部単語区切り (a_text_source オブジェクト) のプロデューサへのポインタ。
out_word_source	`a_word_source *`	エントリポイント関数によって指定される外部単語区切り (a_word_source オブジェクト) へのポインタ。
desired_charset	`const char *`	エントリポイント関数の呼び出し元が、単語区切りの出力があると期待する文字セット。is_binary フラグが 0 の場合、特にネゴシエートされないかぎり、単語区切りへの入力の文字セットでもあります。
actual_charset	`char *`	外部単語区切りライブラリが出力を生成する、(外部ライブラリによってネゴシエーションの一部として指定された) 文字セット。is_binary が 0 の場合、単語区切りへの入力の実際の文字セットでもあります。可能な限り、ライブラリで desired_charset のデータの受け入れと作成を実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

メンバー	タイプ	説明
term_breaker_for	`a_term_breaker_for`	単語区切りを初期化する目的: **TERM_BREAKER_FOR_LOAD** テキストインデックスに対する作成操作、挿入操作、更新操作、および削除操作に使用されます。プレフィルタが指定された場合、入力はプレフィルタリングされる可能性があります。 **TERM_BREAKER_FOR_QUERY** sa_char_term システムプロシージャへのクエリ要素、ストップリスト、および入力の解析に使用されます。TERM_BREAKER_FOR_QUERY の場合、テキストインデックスに対して外部プレフィルタライブラリが指定されていても、プレフィルタリングは行われません。

備考

a_init_term_breaker 構造体は、SQL Anywhere インストールディレクトリの `SDK¥Include` サブディレクトリのヘッダファイル `exttbapiv1.h` によって定義されます。

関連情報

a_term_breaker_for 列挙体 [343 ページ]
単語区切りエントリポイント関数 [351 ページ]
a_text_source interface [339 ページ]
a_word_source インタフェース [344 ページ]

1.3.2.11.5 a_term_breaker_for 列挙体

パイプラインをテキストインデックスの更新中に使用するために構築するか、テキストインデックスの問い合わせに使用するために構築するかを指定する場合に、a_term_breaker_for 列挙体を使用します。

パラメータ

TERM_BREAKER_FOR_LOAD

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テキストインデックスに対する作成操作、挿入操作、更新操作、および削除操作に使用されます。

TERM_BREAKER_FOR_QUERY

sa_char_term システムプロシージャへのクエリ要素、ストップリスト、および入力の解析に使用されます。TERM_BREAKER_FOR_QUERY の場合、テキストインデックスに対して外部プレフィルタライブラリが指定されていても、プレフィルタリングは行われません。

備考

データベースサーバは外部単語区切りを初期化するときに a_init_term_breaker::term_breaker_for の値を設定します。

```
typedef enum a_term_breaker_for {
    TERM_BREAKER_FOR_LOAD = 0,
    TERM_BREAKER_FOR_QUERY
} a_term_breaker_for;
```

a_term_breaker_for 列挙体は、SQL Anywhere インストールディレクトリの `SDK¥Include` サブディレクトリのヘッダファイル `exttbapiv1.h` によって定義されます。

関連情報

a_init_term_breaker 構造体 [342 ページ]

1.3.2.11.6 a_word_source インタフェース

外部単語区切りライブラリは、テキストインデックス操作用に単語区切りを実行するための a_word_source インタフェースを実装する必要があります。

構文

```
typedef struct a_word_source {
   a_sql_uint32 ( SQL_CALLBACK *begin_document )(
                                a_word_source *This
                                , a_sql_uint32 has_prefix );
   a_sql_uint32 ( SQL_CALLBACK *get_words )(
                                a_word_source *This
                                , a_term** words
                                ,a_sql_uint32 *num_words );
   a_sql_uint32 ( SQL_CALLBACK *end_document )(
                                a_word_source *This );
   a_sql_uint32 ( SQL_CALLBACK *fini_all )(
                                a_word_source *This );
   a_server_context    *_context;
   a_text_source       *_my_text_producer;
   a_word_source       *_my_word_producer;
   a_text_source       *_my_text_consumer;
   a_word_source       *_my_word_consumer;
} a_word_source, *p_word_source;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

メンバー

メンバー	タイプ	説明
begin_document	`a_sql_uint32`	ドキュメントの処理に必要なセットアップ手順を実行します。トークン化されているドキュメントがプレフィクスクエリ単語である場合、パラメータ has_prefix は 1、not true、または TRUE に設定されます。has_prefix が TRUE に設定された場合、単語区切りは少なくとも 1 つの単語 (空の可能性があります) を返す必要があります。 パイプラインの初期化の目的が TERM_BREAKER_FOR_QUERY である場合、has_prefix は 1、not true、または TRUE のみに設定できます。 プレフィクスのトークン化の結果は、フレーズの最後の単語が実際のプレフィクス文字列であるフレーズとして扱われます。
get_words	`a_sql_uint32`	a_term 構造体の配列へのポインタを返します。このメソッドは、ドキュメントのすべての内容が単語に分解されるまで、指定されたドキュメントのループで呼び出されます。 データベースサーバでは、ドキュメント内の連続する 2 つの単語の位置の差異が 1 であると予測します。単語区切りが独自のストップリスト処理を実行している場合に、2 つの連続する単語間の差異として 1 より大きい値が返される可能性があります。これは予期される動作で、許容可能です。ただし、位置の差異が 1 で連続していない数値の場合、任意の位置が全文クエリの実行方法に影響を与え、以降の全文クエリに予期しない結果をもたらす可能性があります。
end_document	`a_sql_uint32`	パイプラインによるドキュメントの処理の完了を示し、ドキュメント固有のクリーンアップを実行します。
fini_all	`a_sql_uint32`	すべてのドキュメントの処理が完了し、パイプラインが閉じられようとしているときにデータベースサーバによって呼び出されます。fini_all では最後のクリーンアップ手順が実行されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

メンバー	タイプ	説明
_context	a_server_context *	a_init_term_breaker 構造体内のエントリポイント関数に提供されたデータベースサーバのコンテキスト。単語区切りモジュールではこのコンテキストを使用してデータベースサーバとの直接通信を確立します。
_my_text_producer	a_text_source *	a_init_term_breaker 構造体内のエントリポイント関数に提供された単語区切りの a_text_source プロシージャへのポインタ。このポインタは、文字セット変換が必要な場合は、エントリポイント関数が実行された後に、データベースサーバによって置き換えられる可能性があります。このため、テキストプロデューサへのこのポインタのみが単語区切りによって使用されます。
_my_word_producer	a_word_source *	今後の使用のために予約されており、NULL に初期化する必要があります。
_my_text_consumer	a_text_source *	今後の使用のために予約されており、NULL に初期化する必要があります。
_my_word_consumer	a_word_source *	今後の使用のために予約されており、NULL に初期化する必要があります。

備考

a_word_source インタフェースは、SQL Anywhere インストールディレクトリの `SDK¥Include` サブディレクトリのヘッダファイル `exttbapiv1.h` によって定義されます。

複数の関数の呼び出しにまたがるオペレーティングシステムの同期プリミティブは、外部ライブラリに保持しないようにしてください。

関連情報

a_server_context 構造体 [336 ページ]
a_term 構造体 [347 ページ]
a_init_term_breaker 構造体 [342 ページ]
a_text_source interface [339 ページ]
単語区切りエントリポイント関数 [351 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.11.7 a_term 構造体

a_term 構造体は、単語、単語の長さ、単語の位置を保存します。

構文

```
typedef struct a_term
{
    unsigned char  *word;
    a_sql_uint32   len;
    a_sql_uint32   ch_len;
    a_sql_uint64   pos;
} a_term, *p_term;
```

メンバー

メンバー	型	説明
term	unsigned char *	インデックス付けされる単語。
len	a_sql_uint32	単語の長さ (バイト)。
ch_len	a_sql_uint32	単語の長さ (文字数)。
pos	a_sql_uint64	ドキュメント内の単語の位置。 データベースサーバでは、ドキュメント内の連続する 2 つの単語の位置の差異が 1 であると予測します。単語区切りが独自のストップリスト処理を実行している場合に、2 つの連続する単語間の差異として 1 より大きい値が返される可能性があります。これは予期される動作で、許容可能です。ただし、位置の差異が 1 で連続していない数値の場合、任意の位置が全文クエリの実行方法に影響を与え、以降の全文クエリに予期しない結果をもたらす可能性があります。

備考

各 a_term 構造体は、バイト長、文字長、ドキュメント内の位置による注釈が付けられた単語を表します。

a_term 要素の配列へのポインタは、a_word_source インタフェースの一部として実装された get_words メソッドによって OUT パラメータに返されます。

a_term 構造体 は、SQL Anywhere インストールディレクトリの `SDK¥Include` サブディレクトリのヘッダファイル `exttbapiv1.h` によって定義されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.2.11.8 extpf_use_new_api エントリポイント関数 (事前フィルタ)

extpf_use_new_api エントリポイント関数は、データベースサーバに、外部事前フィルタライブラリに実装されているインタフェースバージョンについて通知します。

現在、バージョン 1 のインタフェースのみがサポートされています。

この関数は外部事前フィルタライブラリに必要です。

構文

```
extern "C" a_sql_uint32 ( extpf_use_new_api )( void );
```

戻り値

関数は符号なし 32 ビット整数値を返します。戻り値は、`extpfapiv1.h` で定義した EXTPF_V1_API のインタフェースバージョン番号です。

備考

この関数の一般的な実装を次に示します。

```
extern "C" a_sql_uint32 ( extpf_use_new_api )( void )
{
   return EXTPF_V1_API;
}
```

1.3.2.11.9 exttb_use_new_api エントリポイント関数 (単語区切り)

exttb_use_new_api エントリポイント関数は、外部単語区切りライブラリに実装されているインタフェースバージョンについての情報を提供します。

現在、バージョン 1 のインタフェースのみがサポートされています。

この関数は外部単語区切りライブラリに必要です。

構文

```
extern "C" a_sql_uint32 (exttb_use_new_api )( void );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

戻り値

関数は符号なし 32 ビット整数値を返します。戻り値は、`exttbapiv1.h` で定義した EXTTB_V1_API のインタフェースバージョン番号です。

備考

この関数の一般的な実装を次に示します。

```
extern "C" a_sql_uint32 ( exttb_use_new_api )( void )
{
   return EXTTB_V1_API;
}
```

1.3.2.11.10 extfn_post_load_library グローバルエントリポイント関数

extfn_post_load_library グローバルエントリポイント関数は、ライブラリ内のいずれかの関数が呼び出される前にライブラリ全体にわたる設定を行うライブラリ固有の要件がある場合に必要です。

この関数が実装され外部ライブラリで公開された場合、外部ライブラリがロードされバージョンチェックが実行された後、この関数はデータベースサーバによって実行されます。その後、外部ライブラリ内の定義されたその他の関数が呼び出されます。

構文

```
extern "C" void ( extfn_post_load_library )( void );
```

備考

外部の単語区切りライブラリと事前フィルタライブラリの両方でこの関数を実装できます。

1.3.2.11.11 extfn_pre_unload_library グローバルエントリポイント関数

extfn_pre_unload_library グローバルエントリポイント関数は、ライブラリがアンロードされる前にライブラリ全体にわたるクリーンアップを行うライブラリ固有の要件がある場合にのみ必要です。

この関数が実装され外部ライブラリで公開された場合、外部ライブラリをアンロードする前にデータベースサーバによってすぐに実行されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

構文

```
extern "C" void ( extfn_pre_unload_library )( void );
```

備考

外部の単語区切りライブラリと事前フィルタライブラリの両方でこの関数を実装できます。

1.3.2.11.12 プレフィルタエントリポイント関数

プレフィルタエントリポイント関数は、外部プレフィルタのインスタンスを初期化し、データの文字セットをネゴシエートします。

構文

```
extern "C" a_sql_uint32 ( SQL_CALLBACK *entry-point-function )( a_init_pre_filter
*data );
```

戻り値

エラーの場合は 1、正常に実行された場合は 0。

パラメータ

entry-point-function

プレフィルタのエントリポイント関数の名前。

data

a_init_pre_filter 構造体へのポインタ。

備考

この構造体は外部プレフィルタライブラリに実装される必要があります。また、複数のスレッドで同時に実行できるため再入力可能である必要があります。

この関数の呼び出し元 (データベースサーバ) は、事前フィルタのプロデューサとして機能する a_text_source オブジェクトへのポインタを提供します。呼び出し元は、入力の文字セットも提供します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

この関数は、外部プレフィルタ (a_text_source 構造体) へのポインタを提供します。この関数は、必要に応じて actual_charset フィールドを変更して、入力 (バイナリでない場合) および出力データの文字セットもネゴシエートします。

desired_charset と actual_charset が同じでない場合、`data->is_binary` フィールドが 1 でないかぎり、データベースサーバは入力データに対して文字セット変換を実行します。`is_binary` が 0 の場合には、入力データは `actual_charset` によって指定された文字セットになります。

文字セットの変換が必要になると、パフォーマンスが低下する原因になる可能性があります。

このエントリポイント関数は、ユーザによって ALTER TEXT CONFIGURATION...PREFILTER EXTERNAL NAME を呼び出して指定されます。

関連情報

a_init_pre_filter 構造体 [338 ページ]
a_init_pre_filter 構造体 [338 ページ]
a_text_source interface [339 ページ]

1.3.2.11.13 単語区切りエントリポイント関数

単語区切りエントリポイント関数は、外部単語区切りのインスタンスを初期化し、データの文字セットをネゴシエートします。

構文

```
extern "C" a_sql_uint32 ( SQL_CALLBACK *entry-point-function )
( a_init_term_breaker *data );
```

戻り値

エラーの場合は 1、正常に実行された場合は 0。

パラメータ

entry-point-function

単語区切りのエントリポイント関数の名前。

data

a_init_term_breaker 構造体へのポインタ。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

備考

この構造体は外部単語区切りライブラリに実装される必要があります。また、複数のスレッドで同時に実行できるため再入力可能である必要があります。

この関数の呼び出し元は、単語区切りのプロデューサとして機能する a_text_source オブジェクトへのポインタを提供します。呼び出し元は、入力の文字セットも提供する必要があります。

この関数は、外部単語区切り (a_word_source 構造体) へのポインタおよびサポートされている文字セットを呼び出し元に提供します。

desired_charset と actual_charset が同じでない場合、データベースサーバは actual_charset によって指定された文字セットへの単語区切り入力を変換します。

文字セットの変換は、パフォーマンスが低下する原因になる可能性があります。

関連情報

a_word_source インタフェース [344 ページ]
a_text_source interface [339 ページ]
a_init_term_breaker 構造体 [342 ページ]

1.3.3 チュートリアル: テーブルデータの行列変換

クエリの FROM 句で PIVOT 句を使用して、テーブル式のテーブルデータを行列変換します。

前提条件

行列変換するテーブルの SELECT 権限が必要です。

コンテキスト

テーブル内のデータの配置を回転し、読み取りやすく分析しやすい方法でデータをグループ化します。

手順

1. *Interactive SQL* を使用してサンプルデータベース (`demo.db`) に接続します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. Employees というテーブルがあると仮定します。このテーブルには、自社の従業員に関する情報 (給与、部門、従業員が居住している州など) が格納されています。各部門について、西海岸の 4 つの州 (オレゴン、カリフォルニア、アリゾナ、ユタ) の中で月給コストが最も高い州を確認しようとしています。次のクエリを実行した後、電卓を使用するか、または自分で答えを計算することができます。

```
SELECT DepartmentID, State, SUM( Salary ) TotalSalary
   FROM Employees
   WHERE State IN ( 'OR', 'CA', 'AZ', 'UT' )
   GROUP BY DepartmentID, State
ORDER BY DepartmentID, State, TotalSalary;
```

DepartmentID	State	TotalSalary
100	UT	306,318.690
200	CA	156,600.000
200	OR	47,653.000
200	UT	37,900.000
300	AZ	93,732.000
300	UT	31,200.000
400	OR	80,339.000
400	UT	107,129.000
500	AZ	85,300.800
500	OR	54,790.000
500	UT	59,479.000

3. 代わりに DepartmentID カラムのテーブルを行列変換して、給与情報を集約することもできます。DepartmentID カラムで行列変換するということは、異なる DepartmentID の値が異なるローに表示されるのではなく、州ごとに集約された部門の給与情報とともに Department の各カラム値が結果セットのカラムとなるということです。この操作を行うには、次の PIVOT 文を実行します。

```
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;
```

この結果では、最初の結果セットにある DepartmentID に使用できる値が、カラム名の一部として使用されています (100_TotalSalary など)。カラム名は、"部門 X の合計給与" を意味します。

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

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

STATE	100_TotalSalary	200_TotalSalary	300_TotalSalary	400_TotalSalary	500_TotalSalary
UT	306,318.690	37,900.000	31,200.000	107,129.000	59,479.000

4. 部門の従業員数を把握していないため、合計給与だけでは十分な情報ではない場合があります。たとえば、カリフォルニア州の部門 200 の合計給与額は $156,600 です。これは、従業員 1 人に高給が支払われている場合も、あるいは従業員 10 人に少額ずつ支払われている場合もあります。結果を明確にするには、次のような文を実行して、部門ごとの従業員数が結果に含まれるように指定します。元のデータセットの DepartmentID カラムにある値についてデータの行列変換を行いますが、新規の集約を追加することになります (この場合は COUNT 操作)。

```
SELECT *
FROM ( SELECT DepartmentID, State, Salary
       FROM Employees
       WHERE State IN ( 'OR', 'CA', 'AZ', 'UT' )
     ) MyPivotSourceData
   PIVOT (
      SUM( Salary ) TotSal, COUNT(*) EmCt
      FOR DepartmentID IN ( 100, 200, 300, 400, 500 )
   ) MyPivotedData
ORDER BY State;
```

STATE	100_TotSal	200_TotSal	300_TotSal	400_TotSal	500_TotSalary	100_EmCt	200_EmCt	300_EmCt	400_EmCt	500_EmCt
AZ	(NULL)	(NULL)	93,732.000	(NULL)	85,300.800	0	0	2	0	2
CA	(NULL)	156,600.000	(NULL)	(NULL)	(NULL)	0	3	0	0	0
OR	(NULL)	47,653.000	(NULL)	80,339.000	54,790.000	0	1	0	2	2
UT	306,318.690	37,900.000	31,200.000	107,129.000	59,479.000	5	1	1	2	2

5. 次の PIVOT 例では、SalesOrderItems テーブルのクエリを実行し、LineID 別に販売活動を確認します。LineID は、ID 値 1 がインサイドセールスを表し、2 が Web サイトでの販売を表しています。

```
SELECT * FROM (
     ( SELECT ProductID, LineID, Quantity FROM GROUPO.SalesOrderItems
       WHERE ShipDate BETWEEN '2000-03-31' AND '2000-04-30' )
     ) MyPivotSourceData
   PIVOT
     ( SUM( Quantity ) TotalQuantity
       FOR LineID IN ( 1 InsideSales, 2 Website )
     ) MyPivotedData
ORDER BY ProductID;
```

ProductID	InsideSales_TotalQuantity	WebsiteSales_TotalQuantity
300	120	(NULL)
301	12	108
302	12	(NULL)
400	312	(NULL)
401	36	228

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ProductID	InsideSales_TotalQuantity	WebsiteSales_TotalQuantity
500	24	60
501	(NULL)	48
600	132	(NULL)
601	(NULL)	132
700	(NULL)	(NULL)

InsideSales では製品 400 の販売で成果を出し、一方 WebsiteSales では製品 402 の販売で成果を出していることが、結果から読み取れます。

6. 次の 2 つの文は同じ結果を返します。しかし、PIVOT 句を使用してデータの配置を回転したほうが、もう一方の SQL を使用して同等の結果にたどりつこうとするより効率的であることがわかります。結果における唯一の差異は、PIVOT の例では、州を表すローに指定した部門 (100 および 200) の給与情報が含まれないという結果になる点です。

   PIVOT 句を使用してクエリを実行し、DepartmentID カラムからデータを回転します。

```
SELECT * FROM ( SELECT DepartmentID, State, Salary FROM Employees )
MyPivotSourceData
       PIVOT (
          SUM( Salary ) TotalSalary
          FOR DepartmentID IN ( 100, 200 )
       ) MyPivotedData
    ORDER BY State;
```

   別の SQL を使用して同等のクエリを実行しても、同様の結果となります。

```
 SELECT MyPivotedData.State, MyPivotedData."100_TotalSalary",
MyPivotedData."200_TotalSalary"
    FROM (
    SELECT __grouped_query_block.State,
    MAX( CASE WHEN __grouped_query_block.DepartmentID = 100 THEN
__grouped_query_block.TotalSalary ELSE NULL END ) AS "100_TotalSalary",
    MAX( CASE WHEN __grouped_query_block.DepartmentID = 200 THEN
__grouped_query_block.TotalSalary ELSE NULL END ) AS "200_TotalSalary"

    FROM ( SELECT DepartmentID, State, SUM( Salary ) AS TotalSalary
    FROM Employees   MyPivotSourceData
             WHERE DepartmentID IN ( 100, 200 )
            GROUP BY DepartmentID, State
    ) AS __grouped_query_block( DepartmentID, State, TotalSalary )
    GROUP BY __grouped_query_block.State
    ) AS MyPivotedData( State, "100_TotalSalary", "200_TotalSalary")
    ORDER BY MyPivotedData.State;
```

1.3.4 クエリ結果の要約、グループ化、ソート

クエリ結果のグループ化およびソートを可能にするプロシージャ、文、句が、いくつかサポートされています。

このセクションの内容:

クエリ結果を要約する集合関数 [356 ページ]
  集合関数は、指定されたカラム中の値の要約を表示します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

GROUP BY 句: クエリ結果のグループへの編成 [360 ページ]
GROUP BY 句は、テーブルの出力をグループに分けます。

HAVING 句: データグループの選択 [365 ページ]
HAVING 句は、クエリが返すローを制限します。

ORDER BY 句: クエリ結果のソート [366 ページ]
ORDER BY 句によって、クエリ結果を 1 つ以上のカラムでソートできます。

UNION、INTERSECT、EXCEPT を使用した、クエリ結果に対する集合操作 [371 ページ]
UNION、INTERSECT、EXCEPT は、2 つ以上のクエリの結果に対して集合操作を実行します。

1.3.4.1 クエリ結果を要約する集合関数

集合関数は、指定されたカラム中の値の要約を表示します。

GROUP BY 句、HAVING 句、ORDER BY 句を使用すれば、集合関数を使用してクエリの結果をグループ化およびソートでき、UNION 演算子を使用すれば、クエリ結果を結合できます。

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
```

テーブル内のすべてのロー、WHERE 句によって指定されるテーブルのサブセット、またはテーブル内の 1 つ以上のローグループに、集合関数を適用できます。データベースサーバは、集合関数が適用されたローのセットごとに 1 つの値を生成します。

サポートされる集合関数の一部を次に示します。

AVG(expression)

返されたローについて提供された式の平均。

COUNT(expression)

提供されたグループで、式が NOT NULL のロー数。

COUNT(*)

各グループの中のロー数。

LIST(string-expr)

各ローグループの中の `string-expr` に対するすべての値で構成されている、カンマで区切られたリストを含む文字列。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

MAX(expression)

返されたローの最大値。

MIN(expression)

返されたローの最小値。

STDDEV(expression)

返されたローの標準偏差。

SUM(expression)

返されたローの合計。

VARIANCE(expression)

返されたローについての式の分散。

AVG、SUM、LIST、COUNT とともにオプションのキーワード、DISTINCT を使用して、重複した値を削除してから、集合関数を適用できます。

構文が参照する式は、通常はカラム名です。より一般的な式の場合もあります。

たとえば、次の文を使用して、単価に 1 ドル加算した場合の、全製品の平均価格を調べることができます。

```
SELECT AVG ( UnitPrice + 1 )
FROM Products;
```

例

次のクエリは、Employees テーブルの中の年俸から、支払い給料の総額を計算します。

```
SELECT SUM( Salary )
FROM Employees;
```

集合関数を使用するには、関数名を入力し、その後ろに式を続けます。この式の値に対して、関数が作用します。この式 (この例では `Salary` カラム) はその関数の引数であり、カッコ内に指定します。

このセクションの内容:

[集合関数を使用できる場所 [358 ページ]](#)
集合関数は、SELECT リストまたはグループ化されたクエリブロックの HAVING 句で使用できます。

[集合関数とデータ型 [358 ページ]](#)
一部の集合関数は特定の種類のデータにだけ意味を持ちます。

[COUNT(*) [358 ページ]](#)
COUNT(*) は、重複を除外しないで、指定されたテーブルのロー数を返します。

[DISTINCT を伴う集合関数 [359 ページ]](#)
クエリに DISTINCT キーワードを指定すると、SUM、AVG、または COUNT の計算前に重複値が削除されます。

[集合関数と NULL [359 ページ]](#)
集合関数が作用しているカラムにある NULL は、NULL を含めてカウントする COUNT(*) 以外の関数では無視されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.4.1.1 集合関数を使用できる場所

集合関数は、SELECT リストまたはグループ化されたクエリブロックの HAVING 句で使用できます。

WHERE 句や JOIN 条件の中では、集合関数は使用できません。しかし、SELECT リストの集合関数による SELECT クエリブロックには、多くの場合、その集合関数が適用されるローを制限する WHERE 句があります。

GROUP BY 句を含まない SELECT クエリブロックで集合関数を使用すると、その関数がすべてのローで操作される場合でも WHERE 句によって定義されたローのサブセットで操作される場合でも、常に 1 つの値が生成されます。

同一の SELECT リストで 2 つ以上の集合関数を使用でき、単一の SELECT クエリブロックで 2 つ以上の集合関数を作成できます。

関連情報

HAVING 句: データグループの選択 [365 ページ]

1.3.4.1.2 集合関数とデータ型

一部の集合関数は特定の種類のデータにだけ意味を持ちます。

たとえば、SUM と AVG は、数値カラムにしか使用できません。

しかし、MIN は、文字型カラムの最小値 (つまり、アルファベットの始めに最も近い値) の検索に使用できます。

```
SELECT MIN( Surname )
   FROM Contacts;
```

1.3.4.1.3 COUNT(*)

COUNT(*) は、重複を除外しないで、指定されたテーブルのロー数を返します。

NULL の入っているローを含め、各ローを個別にカウントします。この関数は、引数として式を必要としません。定義上、この関数は、特定のカラムに関する情報を使用しないからです。

次の文は、Employees テーブルの全従業員数を検出します。

```
SELECT COUNT( * )
   FROM Employees;
```

他の集合関数と同じように、COUNT(*) も、SELECT リストにある他の集合関数や WHERE 句などと結合できます。次に例を示します。

```
SELECT COUNT( * ), AVG( UnitPrice )
   FROM Products
   WHERE UnitPrice > 10;
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

COUNT()	AVG(Products.UnitPrice)
5	18.2

1.3.4.1.4 DISTINCT を伴う集合関数

クエリに DISTINCT キーワードを指定すると、SUM、AVG、または COUNT の計算前に重複値が削除されます。

DISTINCT キーワードは、SUM、AVG、COUNT のオプションです。たとえば、連絡先のある都市の数を検出するには、次の文を実行します。

```
SELECT COUNT( DISTINCT City )
   FROM Contacts;
```

COUNT(DISTINCT Contacts.City)
16

クエリ内で DISTINCT を伴って集合関数を 2 つ以上使用できます。各 DISTINCT は個別に評価されます。次に例を示します。

```
SELECT COUNT( DISTINCT GivenName ) "first names",
       COUNT( DISTINCT Surname ) "last names"
   FROM Contacts;
```

first names	last names
48	60

1.3.4.1.5 集合関数と NULL

集合関数が作用しているカラムにある NULL は、NULL を含めてカウントする COUNT(*) 以外の関数では無視されます。

カラム内のすべての値が NULL であれば、COUNT(column_name) は 0 を返します。

WHERE 句に指定されている条件を満たすローがない場合、COUNT は値 0 を返します。その他の関数は、すべて NULL を返します。例を示します。

```
SELECT COUNT( DISTINCT Name )
   FROM Products
   WHERE UnitPrice > 50;
```

COUNT(DISTINCT Name)
0

```
SELECT AVG( UnitPrice )
   FROM Products
   WHERE UnitPrice > 50;
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

AVG(Products.UnitPrice)
(NULL)

1.3.4.2 GROUP BY 句: クエリ結果のグループへの編成

GROUP BY 句は、テーブルの出力をグループに分けます。

1 つ以上のカラム名によって、または計算カラムの結果によって、ローをグループ化することができます。

> **注記**
>
> WHERE 句と GROUP BY 句を使用する場合は、WHERE 句を GROUP BY 句より前に置きます。GROUPBY 句が存在する場合は、常に HAVING 句より前に置いてください。HAVING 句は指定されているが GROUP BY 句は指定されていない場合、GROUP BY () 句が指定されたと見なされます。

HAVING 句と WHERE 句の両方を 1 つのクエリに使用できます。条件が HAVING 句に置かれている場合は、グループが構成されたあとでのみ結果のローを論理的に制限します。条件が WHERE 句に置かれている場合は、グループが構成される前に論理が評価されるので、時間が節約されます。

このセクションの内容:

GROUP BY のあるクエリを実行する方法 [361 ページ]
GROUP BY 句の ROLLUP サブ句は、いくつかの方法で使用することができます。

複数のカラムを使用した GROUP BY [362 ページ]
式を組み合わせてテーブルをグループ化することができます。

WHERE 句と GROUP BY キーワード [362 ページ]
WHERE 句で GROUP BY 句を指定して、結果をグループ化することができます。

集合関数を伴う GROUP BY [363 ページ]
集合関数を含む文には、通常、GROUP BY 句が使用されます。その場合、集合関数はグループごとに 1 つの値を生成します。

GROUP BY と SQL/2008 標準 [363 ページ]
SQL/2008 標準の構文には、SQL Anywhere の構文よりもかなり多くの制約があります。

関連情報

クエリの集合関数 [195 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.4.2.1 GROUP BY のあるクエリを実行する方法

GROUP BY 句の ROLLUP サブ句は、いくつかの方法で使用することができます。

次のような形式の単一テーブルのクエリを考えてみます。

```
SELECT select-list
FROM table
WHERE where-search-condition
GROUP BY [ group-by-expression | ROLLUP (group-by-expression) ]
HAVING having-search-condition
```

このクエリは、次のように実行します。

WHERE 句を適用する

テーブルのローのサブセットで構成される中間結果が生成されます。

結果をグループに分割する

GROUP BY 句の指示どおりに、各グループに対してローが 1 つある第 2 の中間結果が生成されます。生成された各ローには、グループごとの group-by-expression と、select-list および having-search-condition の集合関数の計算結果が含まれます。

ROLLUP 演算があれば適用する

ROLLUP 演算の一部として計算された小計ローが、結果セットに追加されます。

HAVING 句を適用する

第 2 の中間結果で HAVING 句の基準に満たないローは、この時点で削除されます。

プロジェクトの結果を表示する

最終結果セットの表示に必要なカラムのみを取得することで、第 2 の中間結果から最終結果を生成します。select-list にある式に対応するカラムだけが表示されます。最終結果セットは、第 2 の中間結果セットの射影です。

このプロセスでは、GROUP BY 句のあるクエリについて、いくつかの要件が作成されます。

- WHERE 句が最初に評価される。したがって、どの集合関数も、WHERE 句を満たすローについてのみ評価されます。
- 最終結果セットは、分割されたローを保持している第 2 の中間結果から構築される。第 2 の中間結果は、group-by-expression に一致するローを保持しています。したがって、集合関数ではない式が select-list にある場合、同様に group-by-expression にもその式がなければなりません。射影の段階では関数の評価は行われません。
- group-by-expression にある式を、select-list に含まないことも可能です。その式は、結果に射影されます。

関連情報

GROUPING SETS のショートカットとしての ROLLUP と CUBE [440 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.4.2.2 複数のカラムを使用した GROUP BY

式を組み合わせてテーブルをグループ化することができます。

次のクエリは、まず名前別にグループ化し、次にサイズ別にグループ化した製品の平均価格をリストします。

```
SELECT Name, Size, AVG( UnitPrice )
   FROM Products
   GROUP BY Name, Size;
```

Name	Size	AVG(Products.UnitPrice)
Baseball Cap	One size fits all	9.5
Sweatshirt	Large	24
Tee Shirt	Large	14
Tee Shirt	One size fits all	14
...	...	...

1.3.4.2.3 WHERE 句と GROUP BY キーワード

WHERE 句で GROUP BY 句を指定して、結果をグループ化することができます。

WHERE 句は、GROUP BY 句より先に評価されます。WHERE 句で条件を満たさないローが削除されてから、グループ化が行われます。次に例を示します。

```
SELECT  Name, AVG( UnitPrice )
   FROM Products
   WHERE ID > 400
   GROUP BY Name;
```

クエリ結果の生成に使われるグループには、ID の値が 400 より大きいローだけが含まれます。

例

次に、1 つのクエリの中で、WHERE 句、GROUP BY 句、HAVING 句を使用する例を示します。

```
SELECT Name, SUM( Quantity )
   FROM Products
   WHERE Name LIKE '%shirt%'
   GROUP BY Name
   HAVING SUM( Quantity ) > 100;
```

Name	SUM(Products.Quantity)
Tee Shirt	157

次に例を示します。

- WHERE 句によって、*shirt* という語を含む名前 (Tee Shirt、Sweatshirt) があるローだけが選択されます。
- GROUP BY 句によって、共通の名前のローが集められます。
- SUM 集合関数によって、各グループにある製品の総数が計算されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- HAVING 句によって、最終結果から在庫総数が 100 以下のグループが除外されます。

1.3.4.2.4 集合関数を伴う GROUP BY

集合関数を含む文には、通常、GROUP BY 句が使用されます。その場合、集合関数はグループごとに 1 つの値を生成します。

これらの値はベクトル集約値と呼ばれます。これに対して、スカラ集約値は、GROUP BY 句を使用しない集合関数によって生成される 1 つの値です。

例

次のクエリは、製品の種類別に平均価格をリストします。

```
SELECT Name, AVG( UnitPrice ) AS Price
   FROM Products
   GROUP BY Name;
```

Name	Price
Tee Shirt	12.333333333
Baseball Cap	9.5
Visor	7
Sweatshirt	24
...	...

集合関数と 1 つの GROUP BY 句を持つ SELECT 文が生成するベクトル集約値は、結果の各ローにカラムとして表示されます。それとは対照的に、集合関数があって GROUP BY 句がないクエリが生成するスカラ集約値は、カラムとして表示されますが、ローは 1 つだけです。次に例を示します。

```
SELECT AVG( UnitPrice )
   FROM Products;
```

AVG(Products.UnitPrice)
13.3

1.3.4.2.5 GROUP BY と SQL/2008 標準

SQL/2008 標準の構文には、SQL Anywhere の構文よりもかなり多くの制約があります。

GROUP BY について、SQL/2008 標準では次のように定めています。

- GROUP BY 句で指定された各 `group-by-term` は、カラム参照である必要があります。つまり、テーブルからカラムへの参照は、クエリ FROM 句で参照されます。これらの式は、グループ化カラムと呼ばれます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

● SELECT リスト、HAVING 句、または ORDER BY 句内の集合関数でない式は、グループ化カラムであるか、グループ化カラムだけを参照する必要があります。ただし、オプションの SQL/2008 言語機能 T301 "Functional dependencies" がサポートされている場合、このような参照では、グループ化カラムによって機能的に決定されたクエリ FROM 句からカラムを参照できます。

GROUP BY 句では、group-by-term を、列参照、リテラル定数、変数またはホスト変数、スカラ関数、ユーザ定義関数を含む任意の式にできます。たとえば、次のクエリでは、Employee テーブルは Salary カラムに基づいて 3 つのグループに分割され、グループあたり 1 つのローが生成されます。

```
SELECT COUNT() FROM Employees
   GROUP BY (
      IF SALARY < 25000
         THEN 'low range'
      ELSE IF Salary < 50000
         THEN 'mid range'
      ELSE 'high range'
         ENDIF
      ENDIF);
```

クエリ結果に分割値を含めるには、クエリ SELECT リストに group-by-term を追加する必要があります。構文的に有効であるために、データベースサーバでは、SELECT リスト項目の構文と group-by-term の構文が同じであることが確認されます。ただし、構文的に大きい SQL 構成では、この分析は失敗することがあります。さらに、サブクエリを含む式では等しいことの比較は行われません。

次の例では、データベースサーバは 2 つの IF 式が同じであることを検出し、エラーなしに結果を計算します。

```
SELECT (IF SALARY < 25000 THEN 'low range' ELSE IF Salary < 50000 THEN 'mid range'
ELSE 'high range' ENDIF ENDIF), COUNT()
FROM Employees
GROUP BY (IF SALARY < 25000 THEN 'low range' ELSE IF Salary < 50000 THEN 'mid
range' ELSE 'high range' ENDIF ENDIF);
```

ただし、次のクエリには、エラーを返すサブクエリが GROUP BY 句に含まれています。

```
SELECT (Select State from Employees e WHERE e.EmployeeID = e2.EmployeeID),
      COUNT()
   FROM Employees e2
   GROUP BY (Select State from Employees e WHERE EmployeeID = e2.EmployeeID)
```

より簡潔な方法は、SELECT リスト式でエイリアスを使用し、GROUP BY 句内でエイリアスを参照することです。エイリアスを使用すると、SELECT リストと GROUP BY 句に相関サブクエリを含めることができます。次のような形で使用される SELECT リストエイリアスは、ベンダー拡張です。

```
SELECT (
   IF SALARY < 25000
      THEN 'low range'
   ELSE IF Salary < 50000
      THEN 'mid range'
   ELSE 'high range'
      ENDIF
   ENDIF) AS Salary_Range,
   COUNT() FROM Employees GROUP BY Salary_Range;
```

SQL/2008 言語機能 T301 (Functional dependencies) のすべての部分がサポートされているわけではありませんが、GROUP BY の単語に基づいた派生値がサポートされています。SQL Anywhere は、GROUP BY の単語、リテラル定数、(ホ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

スト) 変数を参照する SELECT リスト式を、それらの値を変更する可能性があるスカラ関数の有無に関係なくサポートしています。例として、次のクエリでは、市と州の組合せ別に従業員の数がリストされます。

```
SELECT City || ' ' || State, SUBSTRING(City,1,3), COUNT()
FROM Employees
GROUP BY City, State
```

1.3.4.3 HAVING 句: データグループの選択

HAVING 句は、クエリが返すローを制限します。

HAVING 句は、WHERE 句が SELECT 句の条件を設定するのと同じような方法で、GROUP BY 句の条件を設定します。

HAVING 句の探索条件は、WHERE 探索条件と同じです。ただし、WHERE 探索条件では集合関数を指定できません。たとえば次の使用方法は有効です。

```
HAVING AVG( UnitPrice ) > 20
```

次の使用方法は無効です。

```
WHERE AVG( UnitPrice ) > 20
```

集合関数を伴う HAVING の使用

次の文は、集合関数を持つ HAVING 句を使用する簡単な例です。

2 種類以上のサイズまたは色がある製品をリストするには、1 種類だけの製品を含むグループは省き、Products テーブルのローを名前でグループ化するクエリが必要です。

```
SELECT Name
   FROM Products
   GROUP BY Name
   HAVING COUNT( * ) > 1;
```

Name
Tee Shirt
Baseball Cap
Visor
Sweatshirt

集合関数を伴わない HAVING の使用

HAVING 句は、集合関数がなくても使用できます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のクエリは、製品をグループ化し、その結果セットを name が B で始まるグループだけに限定します。

```
SELECT Name
   FROM Products
   GROUP BY Name
   HAVING Name LIKE 'B%';
```

Name
Baseball Cap

HAVING における 2 つ以上の条件

HAVING 句では、2 つ以上の検索条件を指定できます。これらの条件は、次の例に示すように、AND、OR、NOT 演算子と組み合わされます。

2 種類以上のサイズまたは色があり、1 種類の単価が 10 ドルを超える製品をリストするには、1 種類だけの製品を含むグループと、単価の最大値が 10 ドル以下のグループを省き、Products テーブルのローを名前でグループ化するクエリが必要です。

```
SELECT Name
   FROM Products
   GROUP BY Name
   HAVING COUNT( * ) > 1
   AND MAX( UnitPrice ) > 10;
```

Name
Tee Shirt
Sweatshirt

関連情報

集合関数を使用できる場所 [358 ページ]

1.3.4.4 ORDER BY 句: クエリ結果のソート

ORDER BY 句によって、クエリ結果を 1 つ以上のカラムでソートできます。

それぞれのソートは、昇順 (ASC) でも降順 (DESC) でも可能です。どちらも指定されていない場合は、ASC が使用されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

簡単な例

次のクエリは、name 順に結果を返します。

```
SELECT ID, Name
   FROM Products
   ORDER BY Name;
```

ID	Name
400	Baseball Cap
401	Baseball Cap
700	Shorts
600	Sweatshirt
...	...

2 つ以上のカラムでのソート

ORDER BY 句で 2 つ以上のカラムを指定すると、ソートはネストされます。

次の文は、Products テーブルにある shirt を、name カラムで昇順ソートしてから、各 name の Quantity カラムで降順ソートします。

```
SELECT ID, Name, Quantity
   FROM Products
   WHERE Name like '%shirt%'
   ORDER BY Name, Quantity DESC;
```

ID	Name	Quantity
600	Sweatshirt	39
601	Sweatshirt	32
302	Tee Shirt	75
301	Tee Shirt	54
...	...	...

カラム位置の使用

カラム名の代わりに、SELECT リストの中のカラムの位置番号を使用できます。カラム名と SELECT リスト番号は混在できます。次の文は両方とも、前述の文と同じ結果を生成します。

```
SELECT ID, Name, Quantity
   FROM Products
   WHERE Name like '%shirt%'
   ORDER BY 2, 3 DESC;
SELECT ID, Name, Quantity
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
   FROM Products
   WHERE Name like '%shirt%'
   ORDER BY 2, Quantity DESC
```

SQL のほとんどのバージョンでは、SELECT リストに ORDER BY 項目があることが必須ですが、SQL Anywhere にはそのような制限はありません。次のクエリでは、SELECT リストに Quantity カラムがありませんが、Quantity 順に結果をソートします。

```
SELECT ID, Name
   FROM Products
   WHERE Name like '%shirt%'
   ORDER BY 2, Quantity DESC;
```

ORDER BY と NULL

ORDER BY では、ソート順が昇順の場合、NULL は他のすべての値の前に来ます。

ORDER BY と大文字と小文字の区別

大文字と小文字が混在するデータに対する ORDER BY 句の影響は、データベースの作成時に指定された、データベース照合順と大文字と小文字の区別によって異なります。

このセクションの内容:

SELECT、UPDATE、DELETE クエリブロック内のロー制限句 [368 ページ]
FIRST 句、TOP 句、および LIMIT 句は、WHERE 句を満たすローのサブセットを返したり、または更新や削除を実行したりできるロー制限句です。

ORDER BY 句と GROUP BY 句 [370 ページ]
ORDER BY 句を使用して、特定の方法で GROUP BY の結果を順序付けできます。

1.3.4.4.1 SELECT、UPDATE、DELETE クエリブロック内のロー制限句

FIRST 句、TOP 句、および LIMIT 句は、WHERE 句を満たすローのサブセットを返したり、または更新や削除を実行したりできるロー制限句です。

FIRST 句、TOP 句、および LIMIT 句は、ORDER BY 句を含む SELECT クエリブロック内で使用できます。FIRST 句と TOP 句は、DELETE および UPDATE クエリブロックでも使用できます。

```
row-limitation-option-1 :
FIRST | TOP { ALL | limit-expression } [ START AT startat-expression ]
```

```
row-limitation-option-2 :
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
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 )
```

ロー制限句は、1 つのみ SELECT 句に指定できます。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

このタイプの句は、SELECT、UPDATE、または DELETE クエリブロックと一緒に使用できます。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。startat-expression を指定しない場合、デフォルトは 1 です。

式 limit-expression + startat-expression -1' は、9223372036854775807 = 2^64-1 未満の値に評価される必要があります。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

このタイプの句は SELECT クエリブロック内でのみ使用できます。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。式 limit-expression + offset-expression は、9223372036854775807 = 2^64-1 未満の値に評価される必要があります。

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。

例

次のクエリは、従業員を姓でソートした場合の最初の従業員に関する情報を返します。

```
SELECT FIRST *
FROM Employees
ORDER BY Surname;
```

次のクエリは、名前が姓でソートされた場合の最初の 5 人の従業員を返します。

```
SELECT TOP 5 *
FROM Employees
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
ORDER BY Surname;
SELECT *
FROM Employees
ORDER BY Surname
LIMIT 5;
```

TOP を使用する場合、START AT を使用してオフセットを指定できます。次の文は、姓で降順にソートした場合の 5 番目と 6 番目の従業員をリストします。

```
SELECT TOP 2 START AT 5 *
FROM Employees
ORDER BY Surname DESC;
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT 2 OFFSET 4;
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT 4,2;
```

矛盾のない結果を得るためには、FIRST と TOP は必ず ORDER BY 句と併用してください。FIRST または TOP を ORDER BY なしで使用すると、構文警告の要因になります。また予期しない結果を生成する可能性があります。

```
CREATE OR REPLACE VARIABLE atop INT = 10;
```

次のクエリは、名前が姓でソートされた場合の最初の 5 人の従業員を返します。

```
SELECT TOP (atop -5) *
FROM Employees
ORDER BY Surname;
SELECT *
FROM Employees
ORDER BY Surname
LIMIT (atop-5);
```

次の文は、姓で降順にソートした場合の 5 番目と 6 番目の従業員をリストします。

```
SELECT TOP (atop - 8) START AT (atop -2 -3) *
FROM Employees
ORDER BY Surname DESC;
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT (atop - 8) OFFSET (atop -2 -3 -1);
```

1.3.4.4.2 ORDER BY 句と GROUP BY 句

ORDER BY 句を使用して、特定の方法で GROUP BY の結果を順序付けできます。

例

次のクエリは、各製品の平均価格を検出し、その平均価格順に結果をソートします。

```
SELECT Name, AVG( UnitPrice )
   FROM Products
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    GROUP BY Name
    ORDER BY AVG( UnitPrice );
```

Name	AVG(Products.UnitPrice)
Visor	7
Baseball Cap	9.5
Tee Shirt	12.333333333
Shorts	15
...	...

1.3.4.5 UNION、INTERSECT、EXCEPT を使用した、クエリ結果に対する集合操作

UNION、INTERSECT、EXCEPT は、2 つ以上のクエリの結果に対して集合操作を実行します。

こうした操作の多くは WHERE 句や HAVING 句を使用しても実行できますが、中にはこれらの集合ベースの演算子を使用せずに実行するのは非常に困難な操作もあります。次に例を示します。

- データが正規化されていない場合、たとえテーブルが関連付けられていなくても、異なるように思われる情報を 1 つの結果セットにまとめたいと思うことがあります。
- WHERE 句や HAVING 句内では、集合演算子によって、NULL を扱う方法が異なります。WHERE 句や HAVING 句内では、NOT NULL のエントリが同じである NULL を含む 2 つのローは、同じと見なされません。2 つの NULL 値は、同じと定義されていないためです。集合演算子は、そうした 2 つのローを同じと見なします。

このセクションの内容:

UNION 句: 結果セットの結合 [371 ページ]
UNION 演算子は、2 つ以上のクエリの結果を結合して、単一の結果セットにします。

EXCEPT 句と INTERSECT 句 [372 ページ]
EXCEPT 句は 2 つの結果セット間の違いを返し、INTERSECT 句は 2 つの結果セットの両方にあるローを返します。

集合操作のルール [372 ページ]
UNION 文、EXCEPT 文、INTERSECT 文に適用されるルールがいくつかあります。

集合演算子と NULL 値 [374 ページ]
集合演算子 UNION、EXCEPT、INTERSECT と探索条件内では、NULL を扱う方法が異なります。

1.3.4.5.1 UNION 句: 結果セットの結合

UNION 演算子は、2 つ以上のクエリの結果を結合して、単一の結果セットにします。

デフォルトでは、UNION 演算子は、結果セットから重複しているローを削除します。ALL オプションを使用すると、重複は削除されません。最終的な結果セットにあるカラムは、最初の結果セットのカラムと同じ名前です。UNION 演算子はいくつでも使用できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトでは、UNION 演算子を複数含んでいる文は、左から右に評価されます。カッコを使用して評価順を指定できます。

たとえば、次の 2 つの式は、重複ローを結果セットから削除する方法が異なるため、等しくありません。

```
x UNION ALL ( y UNION z )
```

```
(x UNION ALL y) UNION z
```

1 つ目の式では、y と z の間の UNION の重複が削除されます。そのセットと x の間の UNION では、重複は削除されません。2 つ目の式では、x と y の間の UNION では重複が含められますが、次の z との UNION では削除されます。

1.3.4.5.2 EXCEPT 句と INTERSECT 句

EXCEPT 句は 2 つの結果セット間の違いを返し、INTERSECT 句は 2 つの結果セットの両方にあるローを返します。

UNION 句と同様に、EXCEPT と INTERSECT には ALL 修飾子を指定できます。ALL 修飾子を使用すると、結果セットから重複ローが削除されません。

1.3.4.5.3 集合操作のルール

UNION 文、EXCEPT 文、INTERSECT 文に適用されるルールがいくつかあります。

優先度

UNION 演算子と EXCEPT 演算子の優先度は同じであり、どちらも左から右へ評価されます。INTERSECT 演算子の優先度は UNION 演算子と EXCEPT 演算子よりも高く、複数の INTERSECT 演算子が使用される場合は同じく左から右へ評価されます。

SELECT リストの項目数は同じにする

クエリ内のすべての SELECT リストは、式 (カラム名、算術式、集合関数など) の数を同じにします。次の文は、最初の SELECT リストが 2 番目のリストより長いので無効です。

```
SELECT store_id, city, state
   FROM stores
UNION
   SELECT store_id, city
     FROM stores_east;
```

データ型を一致させる

SELECT リストで対応する式のデータ型を同じにするか、2 種類のデータ型の間で暗黙的データ変換ができるようにします。または、明示的変換を指定します。

たとえば、CHAR データ型のカラムと INT データ型のカラムの間では、明示的変換が指定されなければ UNION、INTERSECT、または EXCEPT は不可能です。しかし、MONEY データ型のカラムと INT データ型のカラムの間では、集合操作が可能です。

カラム順

集合操作の各クエリで、対応する式を同じ順序で並べます。これは、集合演算子が、SELECT 句の各クエリで指定された順に 1 対 1 で式を比較するためです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

複数の集合操作

次の例のように、いくつかの集合操作を一緒に配列できます。

```
SELECT City AS Cities
   FROM Contacts
   UNION
      SELECT City
         FROM Customers
   UNION
      SELECT City
         FROM Employees;
```

UNION 文では、クエリの順番は重要ではありません。INTERSECT では、2 つ以上のクエリがある場合、順番は重要です。EXCEPT では、順番は常に重要です。

カラム見出し

UNION の結果生成されるテーブルのカラム名は、文中の最初のクエリから取得されます。次の例のように、最初のクエリの SELECT リストで結果セットの新しいカラムヘッダを定義します。

```
SELECT City AS Cities
   FROM Contacts
   UNION
      SELECT City
         FROM Customers;
```

次のクエリでは、カラム見出しは UNION 句の最初のクエリで定義した City のままです。

```
SELECT City
   FROM Contacts
   UNION
      SELECT City AS Cities
         FROM Customers;
```

または、WITH 句を使用してカラム名を定義できます。次に例を示します。

```
WITH V( Cities )
AS ( SELECT City
     FROM Contacts
     UNION
        SELECT City
         FROM Customers )
SELECT * FROM V;
```

結果の順序付け

SELECT 文の WITH 句を使用して、SELECT リスト内のカラム名に順序を付けられます。次に例を示します。

```
WITH V( CityName )
AS ( SELECT City AS Cities
     FROM Contacts
     UNION
        SELECT City
           FROM Customers )
SELECT * FROM V
   ORDER BY CityName;
```

また、クエリのリストの最後に単一の ORDER BY 句を使用できますが、次の例のように、カラム名ではなく整数を使用してください。

```
SELECT City AS Cities
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    FROM Contacts
    UNION
       SELECT City
         FROM Customers
    ORDER BY 1;
```

1.3.4.5.4 集合演算子と NULL 値

集合演算子 UNION、EXCEPT、INTERSECT と探索条件内では、NULL を扱う方法が異なります。

この違いが、集合演算子を使用する主な理由の 1 つです。

ローを比較するとき、集合演算子は、NULL 値を互いに等しいものとして扱います。対照的に、探索条件で NULL が NULL と比較された場合、結果は不定 (真ではない) となります。

この違いがもたらす結果の 1 つとして、`query-1 EXCEPT ALL query-2` の結果セット内のロー数が、常に各クエリの結果セット内のロー数の差異であるということです。

テーブル T1 と T2 を例に説明します。各テーブルには、次のカラムがあります。

```
col1 INT,
col2 CHAR(1)
```

テーブルとデータは次のように設定されています。

```
CREATE TABLE T1 (col1 INT, col2 CHAR(1));
CREATE TABLE T2 (col1 INT, col2 CHAR(1));
INSERT INTO T1 (col1, col2) VALUES(1, 'a');
INSERT INTO T1 (col1, col2) VALUES(2, 'b');
INSERT INTO T1 (col1) VALUES(3);
INSERT INTO T1 (col1) VALUES(3);
INSERT INTO T1 (col1) VALUES(4);
INSERT INTO T1 (col1) VALUES(4);
INSERT INTO T2 (col1, col2) VALUES(1, 'a');
INSERT INTO T2 (col1, col2) VALUES(2, 'x');
INSERT INTO T2 (col1) VALUES(3);
```

テーブル内のデータは次のようになっています。

- テーブル T1

col1	col2
1	a
2	b
3	(NULL)
3	(NULL)
4	(NULL)
4	(NULL)

- テーブル T2

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

col1	col2
1	a
2	x
3	(NULL)

T2 にもある T1 のローを要求するクエリの一例を次に示します。

```
SELECT T1.col1, T1.col2
   FROM T1 JOIN T2
   ON T1.col1 = T2.col1
   AND T1.col2 = T2.col2;
```

T1.col1	T1.col2
1	a

ロー (3, NULL) は、結果セットにありません。これは、NULL と NULL の比較が真ではないためです。対照的に、INTERSECT 演算子を使用してこの問題にアプローチすると、結果に NULL を持つローが含まれます。

```
SELECT col1, col2
   FROM T1
   INTERSECT
      SELECT col1, col2
         FROM T2;
```

col1	col2
1	a
3	(NULL)

次のクエリは、探索条件を使用して T2 にはない T1 のローをリストしています。

```
SELECT col1, col2
   FROM T1
   WHERE col1 NOT IN (
       SELECT col1
         FROM T2
         WHERE T1.col2 = T2.col2 )
   OR col2 NOT IN (
       SELECT col2
         FROM T2
         WHERE T1.col1 = T2.col1 );
```

col1	col2
2	b
3	(NULL)
4	(NULL)
3	(NULL)
4	(NULL)

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

T1 の NULL を含むローは、比較によって除外されていません。対照的に、EXCEPT ALL を使用してこの問題にアプローチすると、両方のテーブルに含まれる NULL を持つローが結果から除外されます。この場合、T2 の (3, NULL) ローは、T1 の (3, NULL) ローと同じと認識されています。

```
SELECT col1, col2
   FROM T1
   EXCEPT ALL
      SELECT col1, col2
         FROM T2;
```

col1	col2
2	b
3	(NULL)
4	(NULL)
4	(NULL)

EXCEPT 演算子を使用すると、結果がさらに制限されます。EXCEPT 演算子は、T1 から (3, NULL) のローを両方とも削除し、また (4, NULL) ローの 1 つを重複として除外しています。

```
SELECT col1, col2
   FROM T1
   EXCEPT
      SELECT col1, col2
         FROM T2;
```

col1	col2
2	b
4	(NULL)

1.3.5 ジョイン: 複数テーブルからのデータ検索

複数のテーブルから関連データを取り出すには、SQL JOIN 演算子を使用してジョイン操作を行います。

データベースを作成する場合は、冗長なエントリを多く含む 1 つの大きなテーブルにではなく、別々のテーブルに各オブジェクト固有の情報を配置して、データを正規化します。ジョイン操作では、複数のテーブル (またはビュー) からの情報を使用して 1 つのより大きいテーブルを再作成します。各種のジョインを使用すると、特定のタスクに適したさまざまな仮想テーブルを作成できます。

このセクションの内容:

テーブルのリストの表示 [377 ページ]

Interactive SQL から接続しているデータベースの、すべてのテーブルとそのカラムを表示します。

ジョイン操作 [378 ページ]

ジョインとは、テーブル内のローを、指定したカラムの値と比較することによって結合する操作のことです。

明示的なジョイン条件 (ON 句) [382 ページ]

キーやナチュラルジョインの代わりに、またはこれらとともに、明示的ジョイン条件 (ON 句) を使用してジョインを指定することができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

クロスジョイン [386 ページ]
　2 つのテーブルのクロスジョインによって、両方のテーブルにあるローの組み合わせで可能なものすべてが生成されます。

内部ジョインと外部ジョイン [387 ページ]
　キーワード INNER、LEFT OUTER、RIGHT OUTER、FULL OUTER は、キージョイン、ナチュラルジョイン、ON 句付きジョインを修飾するときに使用できます。

特殊なジョイン [394 ページ]
　サポートされている特殊なジョインは、いくつかのタイプに分類されています。

ナチュラルジョイン [402 ページ]
　NATURAL JOIN を指定すると、データベースサーバでは同じ名前を持つカラムに基づいてジョイン条件が生成されます。

キージョイン [407 ページ]
　一般的なジョインの多くは 2 つのテーブル間で外部キーによって関連付けられます。

1.3.5.1 テーブルのリストの表示

Interactive SQL から接続しているデータベースの、すべてのテーブルとそのカラムを表示します。

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。

手順

1. Interactive SQL では、F7 キーを押すと、接続したデータベース内のテーブルリストを表示できます。
2. テーブルを選択してからカラムを表示をクリックすると、そのテーブルのカラムが表示されます。
3. [Esc] キーを押すとテーブルリストに戻ります。ここでもう一度 [Esc] キーを押すと *SQL* 文 ウィンドウ枠に戻ります。[Enter] キーを押すと、選択されているテーブルまたはカラム名が *SQL* 文 ウィンドウ枠の現在カーソルが置かれている場所にコピーされます。
4. リストを終了するには、[Esc] キーを押します。

結果

接続しているデータベースの全テーブルのリストが表示されます。各テーブルのカラムを表示するオプションがあります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.2　ジョイン操作

ジョインとは、テーブル内のローを、指定したカラムの値と比較することによって結合する操作のことです。

リレーショナルデータベースは別々のタイプのオブジェクトに関する情報を別々のテーブルに保存します。たとえば、あるテーブルには従業員だけの情報があり、別のテーブルには部署関連の情報があります。Employees テーブルには、従業員の名前や住所などの情報が保存されています。Departments テーブルには、部署名や部長名などの情報が入ります。

ほとんどの問い合わせに対する答えは、さまざまなテーブルの情報を組み合わせることによってのみ取得できます。たとえば、「営業部の責任者は誰か」という質問に対する回答を取得するためには、Departments テーブルで適切な従業員を特定し、Employees テーブルでその従業員名を検索します。

ジョインは複数のテーブルからの情報を取り入れた新規の仮想テーブルを作ることによって、そのような質問に答える手段です。たとえば、Employees テーブルと Departments テーブルの情報を組み合わせて、部長のリストを作成できます。FROM 句を使用して、必要な情報の入ったテーブルを特定します。

ジョインを有効なものにするには、各テーブルの適切なカラムを組み合わせてください。部長のリストを作成するには、組み合わせたテーブルの各ローに部署名とその部署を管理する従業員の名前を指定してください。特定のタイプのジョイン操作を指定するか、ON 句を使用して、複合テーブルにカラムをどのように適合させるかを調節します。

このセクションの内容:

[ジョイン条件 [378 ページ]]
　ジョイン条件を使用するとテーブルをジョインできます。ジョイン条件は、カラム内の値と値の関係に基づいて、ジョインしたテーブルからローのサブセットを返す検索条件です。

[ジョインしたテーブル [379 ページ]]
　サポートされているテーブルのジョインは、いくつかのクラスに分類されています。

[2 つのテーブル間のジョイン [380 ページ]]
　簡単な内部ジョインを使用して、2 つのテーブルにジョインすることができます。

[3 つ以上のテーブル間のジョイン [381 ページ]]
　ジョインできるテーブルの数に制限はありません。

[DELETE 文、UPDATE 文、INSERT 文でのジョイン [381 ページ]]
　ジョインは、DELETE 文、UPDATE 文、INSERT 文、SELECT 文で使用できます。

[ANSI 以外のジョイン [382 ページ]]
　ISO/ANSI 標準のジョインのほか、標準以外のジョインも一部サポートされています。

1.3.5.2.1　ジョイン条件

ジョイン条件を使用するとテーブルをジョインできます。ジョイン条件は、カラム内の値と値の関係に基づいて、ジョインしたテーブルからローのサブセットを返す検索条件です。

たとえば、次のクエリでは Products テーブルと SalesOrderItems テーブルからデータが取り出されます。

```
SELECT *
FROM Products JOIN SalesOrderItems
   ON Products.ID = SalesOrderItems.ProductID;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このクエリのジョイン条件は、次の部分です。

```
Products.ID = SalesOrderItems.ProductID
```

このジョイン条件は、両方のテーブルのローに同じ製品 ID がある場合にかぎり、結果セットでこのローを結合できることを示しています。

ジョイン条件には明示的なものと、生成されたものがあります。明示的ジョイン条件とは、ON 句または WHERE 句の中に置かれたジョイン条件のことです。以下のクエリでは ON 句が使用されています。このクエリでは、2 つのテーブルの直積 (すべてのローの組み合わせ) が生成されます。ただし、ID 番号が一致しないローは除外されます。結果は、注文の詳細が記載された顧客リストになります。

```
SELECT *
FROM Customers
JOIN SalesOrders
ON SalesOrders.CustomerID = Customers.ID;
```

これに対し、生成されたジョイン条件とは、KEY JOIN または NATURAL JOIN を指定すると自動的に作成されるジョイン条件のことです。キージョインの場合、ジョイン条件はテーブル間の外部キー関係に基づいて生成されます。ナチュラルジョインの場合、ジョイン条件は名前が同じカラムに基づいて生成されます。

ヒント

キージョイン構文とナチュラルジョイン構文は、どちらもショートカットです。つまり、KEY も NATURAL も指定しないで JOIN キーワードを使用してから、ON 句内で同じジョイン条件を明示的に記述しても、同じ結果が得られます。

キージョインまたはナチュラルジョインを指定した ON 句を使用すると、使用されるジョイン条件は、明示的に指定したジョイン条件と生成されたジョイン条件の論理積になります。ジョイン条件はキーワード AND と組み合わされます。

1.3.5.2.2 ジョインしたテーブル

サポートされているテーブルのジョインは、いくつかのクラスに分類されています。

CROSS JOIN (クロスジョイン)

2 つのテーブルにこのタイプのジョインを指定すると、両方のテーブルにあるローの可能な組み合わせがすべて生成されます。結果セットのサイズは、1 番目のテーブルにあるローの数と 2 番目のテーブルにあるローの数を乗算したものです。クロスジョインは、直積とも呼ばれます。クロスジョインでは ON 句を使用できません。

KEY JOIN (キージョイン)

このタイプのジョイン条件では、テーブル間の外部キー関係が使用されます。ジョインタイプ (INNER、OUTER など) を指定しないで JOIN キーワードを使用する場合や、ON 句がない場合は、キージョインはデフォルトになります。

NATURAL JOIN (ナチュラルジョイン)

このジョインでは、同じ名前のカラムに基づいて、ジョイン条件が自動的に生成されます。

ON 句を使用したジョイン

ON 句内にジョイン条件を明示的に指定すると、このタイプのジョインになります。これをキージョインまたはナチュラルジョインと併用すると、ジョイン条件には生成されたジョイン条件と明示的ジョイン条件の両方が含まれます。KEY や NATURAL の付かない JOIN キーワードと併用すると、生成されるジョイン条件はありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

内部ジョインと外部ジョイン

キージョイン、ナチュラルジョイン、ON 句付きジョインは、INNER、LEFT OUTER、RIGHT OUTER、FULL OUTER を指定して修飾することもできます。デフォルトは INNER です。LEFT、RIGHT、FULL を使う場合、キーワード OUTER はオプションです。

内部ジョインでは、結果の各ローがジョイン条件を満たします。

左外部ジョインまたは右外部ジョインでは、テーブルのどちらか一方のすべてのローの値が保護されます。もう一方のテーブルでは、ジョイン条件を満たさないローに NULL が返されます。たとえば右外部ジョインでは、右側が保護され、左側に NULL が入力されます。

全外部ジョインでは、両方のテーブルのすべてのローが保護され、ジョイン条件を満たさないローに NULL が入ります。

関連情報

明示的なジョイン条件 (ON 句) [382 ページ]

1.3.5.2.3 2 つのテーブル間のジョイン

簡単な内部ジョインを使用して、2 つのテーブルにジョインすることができます。

簡単な内部ジョインの計算方法を理解するために、次のクエリを例にして考えてみましょう。これは、「在庫数と同数の受注数があったのはどの製品のサイズか」という質問への回答です。

```
SELECT DISTINCT Name, Size,
    SalesOrderItems.Quantity
FROM Products JOIN SalesOrderItems
ON Products.ID = SalesOrderItems.ProductID
  AND Products.Quantity = SalesOrderItems.Quantity;
```

Name	Size	Quantity
Baseball Cap	One size fits all	12
Visor	One size fits all	36

このクエリは次のように解釈できます。次に示すのはこのクエリの処理概念の説明であり、ジョインを含むクエリのセマンティックを例証するためのものです。ここで述べる内容は、データベースサーバが実際に結果セットを計算する過程を示すものではありません。

- Products テーブルと SalesOrderItems テーブルの直積を作成します。直積には 2 つのテーブルのローの組み合わせがすべて含まれます。
- 製品 ID が同じではないローはすべて除外されます (ジョイン条件が `Products.ID = SalesOrderItems.ProductID` であるため)。
- 数量が同じでないローはすべて除外されます (ジョイン条件が `Products.Quantity = SalesOrderItems.Quantity` であるため)。
- 3 つのカラム (Products.Name、Products.Size、SalesOrderItems.Quantity) を持つテーブルが作成されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 重複するローがすべて除外されます (キーワードが DISTINCT であるため)。

関連情報

外部ジョイン [388 ページ]

1.3.5.2.4 3 つ以上のテーブル間のジョイン

ジョインできるテーブルの数に制限はありません。

2 つのテーブルをジョインする場合は、比較するカラムのデータ型は同じか互換性のあるものにしてください。

また、3 つ以上のテーブルをジョインする場合、カッコはオプションです。カッコを使用しない場合には、データベースサーバでは文が左から右に評価されます。そのため、A JOIN B JOIN C は (A JOIN B) JOIN C と同じです。また、次の 2 つの SELECT 文は同じです。

```
SELECT *
FROM A JOIN B JOIN C JOIN D;
```

```
SELECT *
FROM ( ( A JOIN B ) JOIN C ) JOIN D;
```

3 つ以上のテーブルをジョインした場合、そのジョインにはテーブル式が含まれます。A JOIN B JOIN C の例では、テーブル式 A JOIN B が C とジョインされます。つまり、概念上は A と B がジョインされ、その結果が C にジョインされます。

テーブル式に外部ジョインが含まれるときは、ジョインの順序が重要になります。たとえば、A JOIN B LEFT OUTER JOIN C は、(A JOIN B) LEFT OUTER JOIN C と解釈されます。テーブル式 A JOIN B が C にジョインされます。このとき、テーブル式 A JOIN B は保護され、テーブル C には NULL が入力されます。

関連情報

外部ジョイン [388 ページ]
テーブル式のキージョイン [411 ページ]
テーブル式のナチュラルジョイン [405 ページ]

1.3.5.2.5 DELETE 文、UPDATE 文、INSERT 文でのジョイン

ジョインは、DELETE 文、UPDATE 文、INSERT 文、SELECT 文で使用できます。

ansi_update_constraints オプションが Off に設定されていれば、ジョインを含むカーソルをいくつか更新できます。SQL Anywhere 7 より前に作成されたデータベースでは、この設定がデフォルトです。バージョン 7 以降を使って作成されたデータベースでは Cursors がデフォルトです。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.2.6 ANSI 以外のジョイン

ISO/ANSI 標準のジョインのほか、標準以外のジョインも一部サポートされています。

- Transact-SQL の外部ジョイン (*= or =*)
- ジョインで重複する相関名 (スタージョイン)
- キージョイン

REWRITE 関数を使うと、ANSI の機能に相当する ANSI 以外のジョインを確認できます。

関連情報

Transact-SQL の外部ジョイン (*= or =*) [392 ページ]
ジョインで重複する相関名 (スタージョイン) [397 ページ]
キージョイン [407 ページ]

1.3.5.3 明示的なジョイン条件 (ON 句)

キーやナチュラルジョインの代わりに、またはこれらとともに、明示的ジョイン条件 (ON 句) を使用してジョインを指定することができます。

ジョインの直後に ON 句を挿入し、ジョイン条件を指定してください。ジョイン条件は、常にその直前にあるジョインを参照します。ON 句は、ジョインのローに制限を適用します。これは、WHERE 句がクエリのローに制限を適用するのと同様です。

ON 句を使用すると、CROSS JOIN よりも使用しやすいジョインを構成できます。たとえば、SalesOrders テーブルと Employees テーブルのジョインに ON 句を適用できます。この場合、取得される結果のすべてのローで、SalesOrders テーブル内の SalesRepresentative が Employees テーブル内のものと同じになるように制限されます。各ローには、注文とその注文を担当する営業担当者についての情報が入っています。

たとえば、次のクエリでは、最初の ON 句を使用して SalesOrders を Customers にジョインします。また、2 番目の ON 句を使用して、テーブル式 (SalesOrders JOIN Customers) をベーステーブル SalesOrderItems にジョインします。

```
SELECT *
FROM SalesOrders JOIN Customers
    ON SalesOrders.CustomerID = Customers.ID
  JOIN SalesOrderItems
    ON SalesOrderItems.ID = SalesOrders.ID;
```

このセクションの内容:

ON 句でのテーブルの参照 [383 ページ]
ON 句で参照されるテーブルは、その ON 句が修飾するジョインの一部である必要があります。

生成されたジョインと ON 句 [384 ページ]
キーワード JOIN を使用し、ジョインタイプを指定していない場合、ON 句を使用しなければ、キージョインがデフォルトです。指定のない JOIN とともに ON 句を使用すると、キージョインはデフォルトにはならず、生成されたジョイン条件は何も適用されません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

明示的ジョイン条件の種類 [385 ページ]
　　ジョイン条件は、そのほとんどが等号に基づいているため等価ジョインと呼ばれます。

ジョイン条件内の WHERE 句 [385 ページ]
　　外部ジョインを使用する場合を除き、ON 句の代わりに WHERE 句でジョイン条件を指定することができます。

1.3.5.3.1 ON 句でのテーブルの参照

ON 句で参照されるテーブルは、その ON 句が修飾するジョインの一部である必要があります。

たとえば、次の構文は無効です。

```
FROM ( A KEY JOIN B ) JOIN ( C JOIN D ON A.x = C.x )
```

ここでの問題は、ジョイン条件 `A.x = C.x` がテーブル A を参照していることです。テーブル A は、このジョイン条件が修飾するジョイン (この場合 `C JOIN D`) の一部ではありません。

ただし、ANSI/ISO 標準の SQL99 と SQL Anywhere 7.0 については、この規則は適用されません。つまり、テーブル式の間にカンマを使用すれば、ジョインの ON 条件は、構文中にその ON 条件より前にある FROM 句内のテーブルを参照できます。このため、次の構文は有効になります。

```
FROM (A KEY JOIN B) , (C JOIN D ON A.x = C.x)
```

例

次の例では、SalesOrders テーブルを Employees テーブルにジョインします。結果の各ローは、SalesOrders テーブルの SalesRepresentative カラムの値と Employees テーブルの EmployeeID カラムの値が一致するローに対応しています。

```
SELECT Employees.Surname, SalesOrders.ID, SalesOrders.OrderDate
FROM SalesOrders
JOIN Employees
ON SalesOrders.SalesRepresentative = Employees.EmployeeID;
```

Surname	ID	OrderDate
Chin	2008	4/2/2001
Chin	2020	3/4/2001
Chin	2032	7/5/2001
Chin	2044	7/15/2000
Chin	2056	4/15/2001
...	...	...

次はこの例に関する説明です。

- このクエリの結果に含まれているのは、648 個のロー (SalesOrders テーブルの各ローに対応) のみです。直積における 48,600 のローのうち、2 つのテーブルで同じ従業員番号を持っているのは 648 のローだけだからです。
- 結果の順序に意味はありません。ORDER BY 句を追加すると、クエリに特定の順序を指定できます。
- ON 句によって、最終的な結果セットには含まれないカラムが組み込まれます。

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

関連情報

キージョイン [407 ページ]
カンマ [386 ページ]

1.3.5.3.2 生成されたジョインと ON 句

キーワード JOIN を使用し、ジョインタイプを指定していない場合、ON 句を使用しなければ、キージョインがデフォルトです。指定のない JOIN とともに ON 句を使用すると、キージョインはデフォルトにはならず、生成されたジョイン条件は何も適用されません。

たとえば、次の例はキージョインです。キーワード JOIN が使用されており、ON 句もない場合はキージョインがデフォルトだからです。

```
SELECT *
FROM A JOIN B;
```

次は、テーブル A とテーブル B のジョインであり、ジョイン条件 `A.x = B.y` も使用されています。したがって、このジョインはキージョインではありません。

```
SELECT *
FROM A JOIN B ON A.x = B.y;
```

KEY JOIN または NATURAL JOIN を指定し、さらに ON 句も使用すると、最終的なジョイン条件は、生成されたジョイン条件と明示的ジョイン条件の論理積になります。たとえば、次の文にはジョイン条件が 2 つ入っています。1 つはキージョインから生成されたジョイン条件で、もう 1 つは ON 句で明示的に記述されたジョイン条件です。

```
SELECT *
FROM A KEY JOIN B ON A.x = B.y;
```

キージョインによって生成されたジョイン条件が `A.w = B.z` の場合、次の文は上の文と同等です。

```
SELECT *
FROM A JOIN B
  ON A.x = B.y
  AND A.w = B.z;
```

関連情報

キージョイン [407 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.3.3　明示的ジョイン条件の種類

ジョイン条件は、そのほとんどが等号に基づいているため等価ジョインと呼ばれます。

次に例を示します。

```
SELECT *
FROM Departments JOIN Employees
   ON Departments.DepartmentID = Employees.DepartmentID;
```

ただし、ジョイン条件の中で必ず等号 (=) を使うわけではありません。LIKE、SOUNDEX、BETWEEN、> (より大きい)、!= (等しくない) などの探索条件を使用できます。

例

次の例は質問の答えを示しています。"在庫数以上の受注があったのはどの製品か" という質問に対する回答です。

```
SELECT DISTINCT Products.Name
FROM Products JOIN SalesOrderItems
ON Products.ID = SalesOrderItems.ProductID
   AND SalesOrderItems.Quantity > Products.Quantity;
```

1.3.5.3.4　ジョイン条件内の WHERE 句

外部ジョインを使用する場合を除き、ON 句の代わりに WHERE 句でジョイン条件を指定することができます。

ただし、外部ジョインがクエリに含まれる場合には、ON 句と WHERE 句には意味の違いが生じます。

ON 句は、FROM 句の一部であるため、WHERE 句よりも前に処理されます。このことは、外部ジョインについて、WHERE 句を使用することによって内部ジョインに変換できる場合を除き、結果には影響を及ぼしません。

ジョイン条件を ON 句にいれるか、WHERE 句に入れるかを決定するときには、次の規則を考慮してください。

- 外部ジョインを指定するときに WHERE 句にジョイン条件を入れると、外部ジョインが内部ジョインに変換されます。
- ON 句内の条件は、これと関連付けられた JOIN で結合するテーブル式内のテーブルのみ参照できます。ただし、WHERE 句内の条件は、その条件がジョインの一部になっていなくても、任意のテーブルを参照できます。
- ON 句はキーワード CROSS JOIN とともには使用できませんが、WHERE 句はいつでも使用できます。
- ジョイン条件が ON 句内にある場合、キージョインはデフォルトにはなりません。ただし、ジョイン条件が WHERE 句内にあると、キージョインをデフォルトにできます。

このマニュアルの例では、ON 句の中でジョイン条件を使用しています。外部ジョインを使用する場合は、これが必要です。他のジョインでも、それらがジョイン条件であって一般的な探索条件ではないことを明確にするために、ON 句の中でジョイン条件が使用されています。

関連情報

外部ジョインとジョインの条件 [389 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

キージョイン [407 ページ]

1.3.5.4 クロスジョイン

2 つのテーブルのクロスジョインによって、両方のテーブルにあるローの組み合わせで可能なものすべてが生成されます。

クロスジョインは、直積とも呼ばれます。

1 番目のテーブルの各ローは、2 番目のテーブルの各ローとともに 1 回だけ出現します。したがって、結果セットのローの数は、1 番目のテーブルにあるローの数と 2 番目のテーブルにあるローの数の積から、WHERE 句による制限で除外されたローの数を減算した数になります。

クロスジョインでは ON 句を使用できません。ただし、WHERE 句で制限を設けることはできます。

クロスジョインには適用しない内部変更子と外部変更子

WHERE 句で追加した制限がある場合を除いて、両テーブルのすべてのローはいつでもクロスジョインの結果として表示されます。したがって、INNER、LEFT OUTER、RIGHT OUTER のキーワードは、クロスジョインには適用できません。

たとえば、次の文では 2 つのテーブルが結合されます。

```
SELECT *
FROM A CROSS JOIN B;
```

このクエリの結果セットには、A のすべてのカラムと B のすべてのカラムが含まれます。A の 1 つのローと B の 1 つのローの組み合わせそれぞれに対して、結果セットに 1 つのローがあります。A が n 個のロー、B が m 個のローである場合は、クエリによって n × m 個のローが返されます。

このセクションの内容:

カンマ [386 ページ]
カンマは、ジョイン演算子と似た動作をします。

1.3.5.4.1 カンマ

カンマは、ジョイン演算子と似た動作をします。

カンマは、CROSS JOIN キーワードとまったく同じ直積を作成します。ただし、JOIN キーワードはテーブル式を生成しますが、カンマはテーブル式のリストを生成します。

次は、2 つのテーブルの簡単な内部ジョインの例です。この例では、カンマと CROSS JOIN キーワードは同義です。

```
SELECT *
FROM A, B, C
WHERE A.x = B.y;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
SELECT *
FROM A CROSS JOIN B CROSS JOIN C
WHERE A.x = B.y;
```

通常は、カンマをキーワード CROSS JOIN の代わりに使用できます。カンマを使用したテーブル式に含まれる生成されたジョイン条件を除いて、カンマ構文はクロスジョイン構文と同義です。

スタージョイン構文の場合、カンマには特殊な用途があります。

関連情報

テーブル式のキージョイン [411 ページ]
ジョインで重複する相関名 (スタージョイン) [397 ページ]

1.3.5.5 内部ジョインと外部ジョイン

キーワード INNER、LEFT OUTER、RIGHT OUTER、FULL OUTER は、キージョイン、ナチュラルジョイン、ON 句付きジョインを修飾するときに使用できます。

デフォルトは INNER です。これらの変更子はクロスジョインには適用されません。

このセクションの内容:

内部ジョイン [387 ページ]
デフォルトのジョインは内部ジョインです。ジョイン条件を満たすローだけが結果セットに含まれます。

外部ジョイン [388 ページ]
ジョインの一方のテーブルのすべてのローを保護するには、外部ジョインを使用します。

Transact-SQL の外部ジョイン (*= or =*) [392 ページ]
Transact-SQL ダイアレクトでは、FROM 句内にカンマで区切られたテーブルのリストを指定し、WHERE 句内で特殊な演算子 *= or =* を使用して外部ジョインを作成します。

1.3.5.5.1 内部ジョイン

デフォルトのジョインは内部ジョインです。ジョイン条件を満たすローだけが結果セットに含まれます。

例

たとえば、次のクエリの結果セットで、それぞれのローには、キージョイン条件を満たす 1 つの Customers ローと 1 つの SalesOrders ローの情報が含まれます。ある顧客が発注しなかった場合は、条件が満たされないので、この顧客に対応するローは結果セットには含まれません。

```
SELECT GivenName, Surname, OrderDate
FROM Customers KEY INNER JOIN SalesOrders
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ORDER BY OrderDate;
```

GivenName	Surname	OrderDate
Hardy	Mums	2000-01-02
Aram	Najarian	2000-01-03
Tommie	Wooten	2000-01-03
Alfredo	Margolis	2000-01-06
...	...	...

内部ジョインとキージョインはデフォルトなので、次のように FROM 句を使用しても前述の例と同じ結果が得られます。

```
SELECT GivenName, Surname, OrderDate
FROM Customers JOIN SalesOrders
ORDER BY OrderDate;
```

1.3.5.5.2 外部ジョイン

ジョインの一方のテーブルのすべてのローを保護するには、外部ジョインを使用します。

外部ジョインを使用しない場合は、ジョイン条件を満たす場合のみローを返すジョインを作成します。これは内部ジョインと呼ばれ、クエリ時に使用されるデフォルトのジョインです。

2 つのテーブルの左または右の外部ジョインを使用すると、一方のテーブルではすべてのローが保護され、他方のテーブルにはジョイン条件が満たされないときに NULL が入力されます。左外部ジョインでは左側のテーブルのローがすべて保護され、右外部ジョインでは右側テーブルのローがすべて保護されます。全外部ジョインでは、両方のテーブルのすべてのローが保護され、両方のテーブルが NULL 入力となります。

左外部ジョインまたは右外部ジョインのそれぞれの側のテーブル式は、保護されたテーブル式と **NULL** 入力テーブル式と呼ばれます。左外部ジョインでは、左側のテーブル式が保護テーブル式で、右側のテーブル式は NULL 入力テーブル式です。全外部ジョインでは、左側と右側の両方のテーブル式が保護テーブル式であり、両方が NULL 入力テーブル式となります。

例

次の文にはすべての顧客が含まれます。顧客が注文していない場合には、注文情報に対応する結果のそれぞれのカラムに NULL 値が入ります。

```
SELECT Surname, OrderDate, City
FROM Customers LEFT OUTER JOIN SalesOrders
   ON Customers.ID = SalesOrders.CustomerID
WHERE Customers.State = 'NY'
ORDER BY OrderDate;
```

Surname	OrderDate	City
Thompson	(NULL)	Bancroft
Reiser	2000-01-22	Rockwood
Clarke	2000-01-27	Rockwood

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Surname	OrderDate	City
Mentary	2000-01-30	Rockland
...	...	...

この文の外部ジョインは次のように解釈できます。ここで説明しているのは概念であり、データベースサーバが実際に結果セットを計算する過程を示すものではありません。

- 顧客からの発注ごとにローが 1 つ返されます。1 つの発注に対して 1 つのローが返されるため、顧客が 2 つ以上注文した場合には、ローも 2 つ以上返されます。これは内部ジョインの結果と同じです。ON 条件は、customer ローと sales order ローを一致させるために使用します。この手順では、WHERE 句は使用されません。
- 注文しなかったそれぞれの顧客のローが 1 行入ります。これにより、Customers テーブルのすべてのローが確実に含まれます。これらのすべてのローに対して、SalesOrders のカラムに NULL が挿入されます。キーワード OUTER が使用されているため、これらのローは追加されますが、内部ジョインには表示されません。この手順では ON 条件も WHERE 句も使用されません。
- WHERE 句を使用して、New York 在住ではない顧客のローをすべて除外します。

このセクションの内容:

外部ジョインとジョインの条件 [389 ページ]
WHERE 句を使用して NULL 入力テーブルを制限すると、通常、そのジョインは内部ジョインと同義になります。

複雑な外部ジョイン [390 ページ]
クエリに外部ジョインを使用したテーブル式が含まれるときは、ジョインの順序が重要になります。

ビューと派生テーブルの外部ジョイン [391 ページ]
外部ジョインは、ビューと派生テーブルに指定できます。

関連情報

Transact-SQL の外部ジョイン (*= or =*) [392 ページ]

キージョイン [407 ページ]

1.3.5.5.2.1 外部ジョインとジョインの条件

WHERE 句を使用して NULL 入力テーブルを制限すると、通常、そのジョインは内部ジョインと同義になります。

これは、ほとんどの探索条件では、入力した探索条件のうち 1 つでも NULL になっていると TRUE と評価できないためです。NULL 入力テーブルに対する WHERE 句での制限によって、それぞれの値は NULL と比較され、結果セットからそのローは除外されます。保護されたテーブルのローは保護されないので、このジョインは内部ジョインです。

これに対する例外は、入力値のいずれかが NULL の場合は TRUE と評価できる比較です。こうした比較には、IS NULL、IS UNKNOWN、IS FALSE、IS NOT TRUE があります。また、ISNULL や COALESCE を含む式もあります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

たとえば、次の文は左外部ジョインを計算します。

```
SELECT *
FROM Customers KEY LEFT OUTER JOIN SalesOrders
   ON SalesOrders.OrderDate < '2000-01-03';
```

これに対し、次の文は内部ジョインを作成します。

```
SELECT Surname, OrderDate
FROM Customers KEY LEFT OUTER JOIN SalesOrders
   WHERE SalesOrders.OrderDate < '2000-01-03';
```

この 2 つの文のうち、最初の文は次のように考えられます。まず、Customers テーブルを SalesOrders テーブルに左外部ジョインします。結果セットには Customers テーブルのすべてのローが入ります。2000 年 1 月 3 日より前に注文をしていない顧客については、sales order フィールドに NULL が入ります。

2 番目の文では、まず Customers と SalesOrders を左外部ジョインします。結果セットには Customers テーブルのすべてのローが入ります。注文をしていない顧客については、sales order フィールドに NULL が入ります。次に、2000 年 1 月 3 日以降に発注した顧客のローだけを選択することによって WHERE 条件が適用されます。発注しなかった顧客については、これらの値が NULL になります。任意の値を NULL と比較した結果は、UNKNOWN と評価されます。したがって、これらのローは削除されて、文は内部ジョインに縮小されます。

1.3.5.5.2.2 複雑な外部ジョイン

クエリに外部ジョインを使用したテーブル式が含まれるときは、ジョインの順序が重要になります。

たとえば、`A JOIN B LEFT OUTER JOIN C` は、`(A JOIN B) LEFT OUTER JOIN C` と解釈されます。テーブル式 `(A JOIN B)` が C にジョインされます。このとき、テーブル式 `(A JOIN B)` は保護され、テーブル C には NULL が入力されます。

次に、以下の文を考えてみます。A、B、C はテーブルです。

```
SELECT *
FROM A LEFT OUTER JOIN B RIGHT OUTER JOIN C;
```

この文を理解するには、まずデータベースサーバでは文が左から右に評価され、カッコが追加される、という規則を思い出してください。結果は次のような設定になります。

```
SELECT *
FROM (A LEFT OUTER JOIN B) RIGHT OUTER JOIN C;
```

次に、両方のジョインが同じタイプになるように、右外部ジョインを左外部ジョインに変換します。これを行うには、右外部ジョインにあるテーブルの位置を単純に逆にします。結果は次のようになります。

```
SELECT *
FROM C LEFT OUTER JOIN (A LEFT OUTER JOIN B);
```

ネストした外部ジョインでは、A が保護テーブルであり、B が NULL 入力テーブルです。最初の外部ジョインでは C が保護テーブルです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

この結合は次のように解釈できます。

- A を B に結合します。このとき、A のローはすべて保護されます。
- 次に、C を A と B のジョインの結果に結合します。このとき、C のローはすべて保護されます。

このジョインには ON 句がないため、デフォルトのキージョインになります。

また、外部ジョインのジョイン条件には、必ず、FROM 句内で先に参照されているテーブルだけを入れます。この制限事項は ANSI/ISO 標準に基づくものであり、あいまいさを排除するためのものです。たとえば、次の 2 つの文は構文的に正しくありません。テーブル自体が参照される前に C がジョイン条件内で参照されるからです。

```
SELECT *
FROM (A LEFT OUTER JOIN B ON B.x = C.x) JOIN C;
```

```
SELECT *
FROM A LEFT OUTER JOIN B ON A.x = C.x, C;
```

関連情報

カンマを含まないテーブル式のキージョイン [412 ページ]

1.3.5.5.2.3 ビューと派生テーブルの外部ジョイン

外部ジョインは、ビューと派生テーブルに指定できます。

次に例を示します。

```
SELECT *
FROM V LEFT OUTER JOIN A ON (V.x = A.x);
```

この例は、次のように解釈できます。

- ビュー V が計算されます。
- ジョイン条件 `V.x = A.x` を使用して V のローをすべて保護すると、計算されたビュー V のすべてのローが A にジョインされます。

例

次の例では、ビュー V を定義します。ここで、ビュー V は、$60,000 を上回る収入がある女性の従業員 ID と部署名を返します。

```
CREATE VIEW V AS
SELECT Employees.EmployeeID, DepartmentName
  FROM Employees JOIN Departments
    ON Employees.DepartmentID = Departments.DepartmentID
  WHERE Sex = 'F' and Salary > 60000;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次に、このビューを使用してそれらの女性が勤務する部署と販売地区のリストを追加します。ビュー V は保護ビューであり、SalesOrders は NULL 入力です。

```
SELECT DISTINCT V.EmployeeID, Region, V.DepartmentName
  FROM V LEFT OUTER JOIN SalesOrders
    ON V.EmployeeID = SalesOrders.SalesRepresentative;
```

EmployeeID	Region	DepartmentName
243	(NULL)	R & D
316	(NULL)	R & D
529	(NULL)	R & D
902	Eastern	Sales
...	...	...

1.3.5.5.3 Transact-SQL の外部ジョイン (*= or =*)

Transact-SQL ダイアレクトでは、FROM 句内にカンマで区切られたテーブルのリストを指定し、WHERE 句内で特殊な演算子 *= or =* を使用して外部ジョインを作成します。

ANSI/ISO SQL 標準に基づき、キーワード LEFT OUTER、RIGHT OUTER、FULL OUTER がサポートされています。また、バージョン 12 以前の Adaptive Server Enterprise との互換性を保つために、tsql_outer_joins データベースオプションが On に設定されている場合は、これらのキーワードに対応する Transact-SQL の *= と =* もサポートされています。

Transact-SQL のセマンティックには、いくつかの制限事項と潜在的な問題があります。Transact-SQL 外部ジョインの詳細については、Transact-SQL 外部ジョインのセマンティックと互換性を参照してください。

外部ジョインを作成するときには、*= 構文と ON 句の構文を混在させないでください。この制限は、クエリで参照されるビューにも適用されます。

注記

Transact-SQL 外部ジョイン演算子 *= と =* は旧式であるため、将来のリリースではサポートから除外されます。

例

次の左外部ジョインは全顧客をリストし、注文があればその日付を取り出します。

```
SELECT GivenName, Surname, OrderDate
FROM Customers, SalesOrders
WHERE Customers.ID *= SalesOrders.CustomerID
ORDER BY OrderDate;
```

この文は次の文と同義です。ここでは ANSI/ISO 構文が使用されています。

```
SELECT GivenName, Surname, OrderDate
FROM Customers LEFT OUTER JOIN SalesOrders
ON Customers.ID = SalesOrders.CustomerID
ORDER BY OrderDate;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

Transact-SQL 外部ジョインの制限事項 [393 ページ]
Transact-SQL 外部ジョインにはいくつか制限があります。

ビューと Transact-SQL 外部ジョイン [394 ページ]
外部ジョインでビューを定義し、外部ジョインの NULL 入力テーブルからのカラムに対する条件でビューに問い合わせると、予期しない結果になる場合があります。

Transact-SQL ジョインに対する NULL の影響 [394 ページ]
Transact-SQL 外部ジョインでは、ジョインされるテーブルまたはビューの NULL 値は、互いに一致することはありません。

1.3.5.5.3.1 Transact-SQL 外部ジョインの制限事項

Transact-SQL 外部ジョインにはいくつか制限があります。

- 外部ジョインと、外部ジョインの NULL 入力テーブルのカラムに対して条件を指定した場合、予想外の結果になることがあります。クエリ内の条件は結果セットからローを除外するのではなく、結果セットに含まれるローの値の方にも影響を与えます。条件を満たさないローについては、NULL 入力テーブルに NULL 値が入ります。
- ANSI/ISO SQL 構文と Transact-SQL 外部ジョイン構文を、1 つのクエリ内で混在させることはできません。ビューが外部ジョインのダイアレクトを使用して定義されている場合、そのビューのすべての外部ジョインクエリに同じダイアレクトを使用する必要があります。
- 1 つの NULL 入力テーブルを Transact-SQL 外部ジョインと通常のジョインの両方、または 2 つの外部ジョインに使用することはできません。たとえば、次の WHERE 句は、テーブル S がこの制限事項に違反しているため無効です。

```
WHERE R.x *= S.x
AND S.y = T.y
```

  外部ジョインと通常のジョイン句の両方で同じテーブルを使用しないようにクエリを書き直すことができない場合には、文を 2 種類のクエリに分けるか、ANSI/ISO SQL 構文のみを使用してください。
- 外部ジョインの NULL 入力テーブルを含むジョイン条件を持つサブクエリは使用できません。たとえば、次の WHERE 句は許可されません。

```
WHERE R.x *= S.y
AND EXISTS ( SELECT *
             FROM T
             WHERE T.x = S.x )
```

注記

Transact-SQL 外部ジョイン演算子 *= と =* は旧式であるため、将来のリリースではサポートから除外されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.5.3.2 ビューと Transact-SQL 外部ジョイン

外部ジョインでビューを定義し、外部ジョインの NULL 入力テーブルからのカラムに対する条件でビューに問い合わせると、予期しない結果になる場合があります。

クエリは NULL 入力テーブルからすべてのローを戻します。その条件に一致しないローはそのローの適切なカラム内に NULL 値を表示します。

次の規則によって、外部ジョインを含むビューを使用してカラムに実行できる更新の種類が決定します。

- INSERT 文と DELETE 文は外部ジョインビューでは使用できません。
- UPDATE 文は外部ジョインビューで使用できます。ビューの定義が WITH CHECK オプションの場合、複数のテーブルからのカラムを含む式の中の WHERE 句に、影響を受けるカラムがあると、更新は失敗します。

1.3.5.5.3.3 Transact-SQL ジョインに対する NULL の影響

Transact-SQL 外部ジョインでは、ジョインされるテーブルまたはビューの NULL 値は、互いに一致することはありません。

NULL 値と他の NULL 値を比較した結果は、FALSE になります。

1.3.5.6 特殊なジョイン

サポートされている特殊なジョインは、いくつかのタイプに分類されています。

このセクションの内容:

セルフジョイン [395 ページ]
セルフジョインでは、異なる相関名を使用して同一テーブルを参照することによって、テーブルがそれ自体にジョインされます。

ジョインで重複する相関名 (スタージョイン) [397 ページ]
スタージョインは、1 つのテーブルまたはビューを複数のテーブルやビューにジョインします。スタージョインを作成するには、同じテーブル名、ビュー名、または相関名を FROM 句内で 2 回以上使用します。

派生テーブルを使用するジョイン [400 ページ]
派生テーブルでは FROM 句内にクエリをネストできます。派生テーブルを使用すると、別のビューやテーブルを作成してそれにジョインすることなく、グループのグループ化を実行したり、グループを使用してジョインを構成することができます。

適用式から生成されるジョイン [401 ページ]
APPLY 式を使用すると、右側が左側に依存するジョインを簡単に指定できます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.6.1　セルフジョイン

セルフジョインでは、異なる相関名を使用して同一テーブルを参照することによって、テーブルがそれ自体にジョインされます。

例

例 1

次のセルフジョインは従業員のペアのリストを作成します。各従業員の名前が全従業員の名前との組み合わせで表示されます。

```
SELECT a.GivenName, a.Surname,
      b.GivenName, b.Surname
FROM Employees AS a CROSS JOIN Employees AS b;
```

GivenName	Surname	GivenName	Surname
Fran	Whitney	Fran	Whitney
Fran	Whitney	Matthew	Cobb
Fran	Whitney	Philip	Chin
Fran	Whitney	Julie	Jordan
...	...	...	...

Employees テーブルには 75 個のローがあるので、このジョインには 75 × 75 = 5625 のローがあります。これには、従業員が自分自身をリストしたローも含まれます。たとえば、次のようなローです。

GivenName	Surname	GivenName	Surname
Fran	Whitney	Fran	Whitney

同じ名前を 2 回含むローを除外する場合は、互いの従業員 ID は同じではならないというジョイン条件を追加します。

```
SELECT a.GivenName, a.Surname,
      b.GivenName, b.Surname
FROM Employees AS a CROSS JOIN Employees AS b
WHERE a.EmployeeID != b.EmployeeID;
```

この重複するローを除くと、ジョインは 75 × 74 = 5550 ローで構成されます。

この新しいジョインは各従業員が自分以外の従業員とペアになったローで構成されます。しかし、各ペアの名前の表示には 2 通りの順番があるので、各ペアは 2 度表示されます。たとえば、前述のジョインには次の 2 つのローがあります。

GivenName	Surname	GivenName	Surname
Matthew	Cobb	Fran	Whitney
Fran	Whitney	Matthew	Cobb

名前の順番が重要でない場合は、同一ペアの表示が一度だけになる (75 × 74) / 2 = 2775 ローのリストを作成できます。

```
SELECT a.GivenName, a.Surname,
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
      b.GivenName, b.Surname
FROM Employees AS a CROSS JOIN Employees AS b
WHERE a.EmployeeID < b.EmployeeID;
```

この文は、従業員 a の EmployeeID が従業員 b の EmployeeID より小さいローのみを選択して、重複する行を削除します。

例 2

次のセルフジョインは相関名 report と manager を使用して、Employees テーブルの 2 つのインスタンスを区別し、従業員とその管理者のリストを作成します。

```
SELECT report.GivenName, report.Surname,
   manager.GivenName, manager.Surname
FROM Employees AS report JOIN Employees AS manager
   ON (report.ManagerID = manager.EmployeeID)
ORDER BY report.Surname, report.GivenName;
```

この文から、次に一部を示すテーブルが作成されます。従業員名は左側の 2 つのカラムに、マネージャ名は右側に表示されます。

GivenName	Surname	GivenName	Surname
Alex	Ahmed	Scott	Evans
Joseph	Barker	Jose	Martinez
Irene	Barletta	Scott	Evans
Jeannette	Bertrand	Jose	Martinez
...	...	...	...

例 3

次のセルフジョインによって、2 つのレベルのレポートを持つすべてのマネージャと 2 番目のレベルのレポート数のリストが生成されます。

```
SELECT higher.managerID, count(*) second_level_reports
FROM employees lower JOIN employees higher
   ON ( lower.managerID = higher.employeeID )
GROUP BY higher.managerID
ORDER BY higher.managerID DESC;
```

上記のクエリの結果には、次のローが含まれています。

ManagerID	second_level_reports
1293	30
902	23
501	22

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.6.2 ジョインで重複する相関名 (スタージョイン)

スタージョインは、1 つのテーブルまたはビューを複数のテーブルやビューにジョインします。スタージョインを作成するには、同じテーブル名、ビュー名、または相関名を FROM 句内で 2 回以上使用します。

スタージョインは、ANSI/ISO SQL 標準の拡張機能です。重複名を使用しても機能は追加されませんが、使用すると特定のクエリを簡単に作成できます。

重複名は、構文が意味をなすように、必ず異なるジョイン内に置きます。同一ジョイン内でテーブル名やビュー名を 2 回使用すると、2 番目のインスタンスは無視されます。たとえば、`FROM A,A` と `FROM A CROSS JOIN A` は、どちらも `FROM A` と解釈されます。

次の例は SQL Anywhere で有効です。A、B、C はテーブルです。この例では、テーブル A の同一インスタンスは B と C のどちらにもジョインされます。スタージョインでジョインを分けるときにはカンマが必要です。スタージョインでのカンマの使用方法は、スタージョインの構文特有のものです。

```
SELECT *
FROM A LEFT OUTER JOIN B ON A.x = B.x,
     A LEFT OUTER JOIN C ON A.y = C.y;
```

これは、次の例と同義です。

```
SELECT *
FROM A LEFT OUTER JOIN B ON A.x = B.x,
     C RIGHT OUTER JOIN A ON A.y = C.y;
```

2 つの例はどちらも次の ANSI/ISO 標準構文と同義です(カッコはオプションです)。

```
SELECT *
FROM (A LEFT OUTER JOIN B ON A.x = B.x)
LEFT OUTER JOIN C ON A.y = C.y;
```

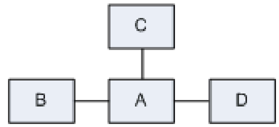
次の例では、テーブル A が 3 つのテーブル B、C、D にジョインされます。

```
SELECT *
FROM A JOIN B ON A.x = B.x,
     A JOIN C ON A.y = C.y,
     A JOIN D ON A.w = D.w;
```

上の例は、次の ANSI/ISO 標準構文と同義です(カッコはオプションです)。

```
SELECT *
FROM ((A JOIN B ON A.x = B.x)
JOIN C ON A.y = C.y)
JOIN D ON A.w = D.w;
```

複雑なジョインは図にするとわかりやすくなります。前述の例は次の図で説明できます。この図から、テーブル B、C、D がテーブル A を介してジョインされることがわかります。



© 2016 SAP SE or an SAP affiliate company.All rights reserved.

i 注記

重複するテーブル名を使用できるのは、extended_join_syntax オプションが On (デフォルト) になっている場合だけです。

-

例

例 1

Rollin Overbey に発注した顧客名リストを作成します。FROM 句では、Employees テーブルのどのカラムも結果に表示されません。また、Customers.ID や Employees.EmployeeID など、ジョインしたどのカラムも結果に表示されません。それでも、FROM 句に Employees テーブルを使用するだけで、このジョインは可能です。

```
SELECT Customers.GivenName, Customers.Surname,
   SalesOrders.OrderDate
FROM   SalesOrders KEY JOIN Customers,
      SalesOrders KEY JOIN Employees
WHERE Employees.GivenName = 'Rollin'
  AND Employees.Surname = 'Overbey'
ORDER BY SalesOrders.OrderDate;
```

GivenName	Surname	OrderDate
Tommie	Wooten	2000-01-03
Michael	Agliori	2000-01-08
Salton	Pepper	2000-01-17
Tommie	Wooten	2000-01-23
...	...	...

次の例は、ANSI/ISO 標準構文の文と同義です。

```
SELECT Customers.GivenName, Customers.Surname,
  SalesOrders.OrderDate
FROM SalesOrders JOIN Customers
  ON SalesOrders.CustomerID =
   Customers.ID
JOIN Employees
  ON SalesOrders.SalesRepresentative =
   Employees.EmployeeID
WHERE Employees.GivenName = 'Rollin'
  AND Employees.Surname = 'Overbey'
ORDER BY SalesOrders.OrderDate;
```

例 2

次の例では、"各顧客が製品ごとに発注した数はどれくらいか、また受注した営業部員の管理者は誰か" という質問に回答します。

回答するために、まず、どの情報を取り出すのかをリストします。ここでは、製品、数量、顧客名、管理者名を取り出します。次に、これらの情報を保持しているテーブルをリストします。ここでは、Products、SalesOrderItems、Customers、Employees になります。SQL Anywhere サンプルデータベースの構造を見ると、これらのテーブルが

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

すべて SalesOrders テーブルを介して関連していることがわかります。SalesOrders テーブルにスタージョインを作成すると、他のテーブルから情報を取り出すことができます。

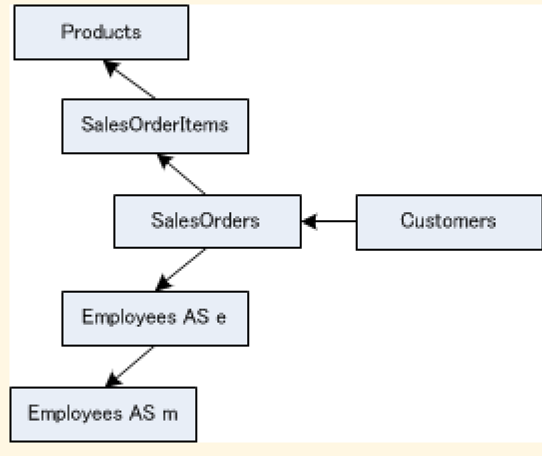
また、管理者名を取得するには、セルフジョインを作成する必要があります。Employees テーブルには、管理者の ID 番号とすべての従業員の名前はありますが、管理者名だけをリストしたカラムがないからです。

次の文は SalesOrders テーブルを中心にしてスタージョインを作成します。このジョインは、結果セットにすべての顧客が含まれるように、すべて外部ジョインになっています。注文しなかった顧客もいますが、そういう顧客の他の値は NULL になっています。結果セットのカラムは、Customers、Products、Quantity ordered、営業部員の管理者名です。

```
SELECT Customers.GivenName, Products.Name,
  SUM(SalesOrderItems.Quantity), m.GivenName
FROM SalesOrders
    KEY RIGHT OUTER JOIN Customers,
   SalesOrders
    KEY LEFT OUTER JOIN SalesOrderItems
    KEY LEFT OUTER JOIN Products,
      SalesOrders
    KEY LEFT OUTER JOIN Employees AS e
    LEFT OUTER JOIN Employees AS m
        ON (e.ManagerID = m.EmployeeID)
WHERE Customers.State = 'CA'
GROUP BY Customers.GivenName, Products.Name, m.GivenName
ORDER BY SUM(SalesOrderItems.Quantity) DESC,
   Customers.GivenName;
```

GivenName	Name	SUM(SalesOrderItems.Quantity)	GivenName
Sheng	Baseball Cap	240	Moira
Laura	Tee Shirt	192	Moira
Moe	Tee Shirt	192	Moira
Leilani	Sweatshirt	132	Moira
...	...	...	...

以下は、このスタージョインを使ったテーブルの図です。矢印は、外部ジョインの方向 (右または左) を示します。顧客の完全なリストはすべてのジョイン全体で保持されます。



© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次に示す ANSI/ISO 標準構文は、例 2 のスタージョインと同義です。

```
SELECT Customers.GivenName, Products.Name,
  SUM(SalesOrderItems.Quantity), m.GivenName
FROM SalesOrders LEFT OUTER JOIN SalesOrderItems
    ON SalesOrders.ID = SalesOrderItems.ID
  LEFT OUTER JOIN Products
    ON SalesOrderItems.ProductID = Products.ID
  LEFT OUTER JOIN Employees as e
    ON SalesOrders.SalesRepresentative = e.EmployeeID
  LEFT OUTER JOIN Employees as m
    ON e.ManagerID = m.EmployeeID
  RIGHT OUTER JOIN Customers
    ON SalesOrders.CustomerID = Customers.ID
WHERE Customers.State = 'CA'
GROUP BY Customers.GivenName, Products.Name, m.GivenName
ORDER BY SUM(SalesOrderItems.Quantity) DESC,
   Customers.GivenName;
```

関連情報

セルフジョイン [395 ページ]

1.3.5.6.3 派生テーブルを使用するジョイン

派生テーブルでは FROM 句内にクエリをネストできます。派生テーブルを使用すると、別のビューやテーブルを作成してそれにジョインすることなく、グループのグループ化を実行したり、グループを使用してジョインを構成することができます。

次の例では、内側の SELECT 文 (カッコに囲まれている) は顧客 ID 値でグループ分けされた派生テーブルを作成します。外側の SELECT 文はこのテーブルに相関名 sales_order_counts を割り当て、ジョイン条件を使用してそれを Customers テーブルとジョインします。

```
SELECT Surname, GivenName, number_of_orders
FROM Customers JOIN
   (  SELECT CustomerID, COUNT(*)
      FROM SalesOrders
      GROUP BY CustomerID  )
   AS sales_order_counts ( CustomerID, number_of_orders )
   ON ( Customers.ID = sales_order_counts.CustomerID )
WHERE number_of_orders > 3;
```

結果は各顧客の注文数を含む、4 件以上の注文をした顧客名のテーブルです。

関連情報

ビューと派生テーブルのキージョイン [416 ページ]
ビューと派生テーブルのナチュラルジョイン [406 ページ]
ビューと派生テーブルの外部ジョイン [391 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.6.4 適用式から生成されるジョイン

APPLY 式を使用すると、右側が左側に依存するジョインを簡単に指定できます。

たとえば、適用式を使用して、テーブル式内のローごとに 1 回ずつプロシージャまたは派生テーブルを評価できます。適用式は SELECT 文の FROM 句内に配置し、ON 句を使用することはできません。

APPLY を使用すると複数のソースからローを結合できます。この動作は JOIN に似ていますが、APPLY には ON 条件を指定することはできません。APPLY と JOIN の主な相違点は、APPLY の右側は左側の現在のローによって変わる場合があるという点です。左側のローごとに、右側が再計算され、結果のローが左側のローに結合されます。左側の 1 つのローが右側の複数の行を返すケースでは、右側から返されたローの数だけ左側が重複する結果となります。

指定できる APPLY のタイプは、CROSS APPLY と OUTER APPLY。CROSS APPLY は、右側の結果を生成する左側のローのみを返します。OUTER APPLY は、CROSS APPLY が返すすべてのローと、(右側に NULL が入力されたため) 右側からローが返されない左側のすべてのローを返します。

適用式の構文は、次のとおりです。

```
table-expression { CROSS | OUTER } APPLY table-expression
```

例

次の例では、部署 ID を入力値として受け取り、その部署内で給与が 80,000 ドルを超えるすべての従業員の名前を返すプロシージャ EmployeesWithHighSalary を作成します。

```
CREATE PROCEDURE EmployeesWithHighSalary( IN dept INTEGER )
  RESULT ( Name LONG VARCHAR )
   BEGIN
    SELECT E.GivenName || ' ' || E.Surname
        FROM Employees E
        WHERE E.DepartmentID = dept AND E.Salary > 80000;
   END;
```

次のクエリでは、OUTER APPLY を使用して Departments テーブルを EmployeesWithHighSalary プロシージャの結果にジョインし、各部署で給与が 80,000 ドルを超えるすべての従業員の名前を返します。また、このクエリでは、右側が NULL のローを返し、給与が 80,000 ドルを超える従業員が存在しない各部署も示します。

```
SELECT D.DepartmentName, HS.Name
    FROM Departments D
    OUTER APPLY EmployeesWithHighSalary( D.DepartmentID ) AS HS;
```

DepartmentName	Name
R & D	Kim Lull
R & D	David Scott
R & D	John Sheffield
Sales	Moira Kelly
Finance	Mary Anne Shea
Marketing	NULL
Shipping	NULL

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のクエリでは、CROSS APPLY を使用して Departments テーブルを EmployeesWithHighSalary プロシージャの結果にジョインします。右側に NULL が指定されたローは含まれません。

```
SELECT D.DepartmentName, HS.Name
    FROM Departments D
    CROSS APPLY EmployeesWithHighSalary( D.DepartmentID ) AS HS;
```

DepartmentName	Name
R & D	Kim Lull
R & D	David Scott
R & D	John Sheffield
Sales	Moira Kelly
Finance	Mary Anne Shea

次のクエリでは、前述のクエリと同じ結果が返されますが、CROSS APPLY の右側として派生テーブルが使用されます。

```
SELECT D.DepartmentName, HS.Name
   FROM Departments D
    CROSS APPLY (
        SELECT E.GivenName || ' ' || E.Surname
            FROM Employees E
            WHERE E.DepartmentID = D.DepartmentID AND E.Salary > 80000
    ) HS( Name );
```

関連情報

キージョイン [407 ページ]
クロスジョイン [386 ページ]
内部ジョインと外部ジョイン [387 ページ]

1.3.5.7 ナチュラルジョイン

NATURAL JOIN を指定すると、データベースサーバでは同じ名前を持つカラムに基づいてジョイン条件が生成されます。

生成されたジョイン条件がベーステーブルのナチュラルジョインに有効になるためには、同じ名前のカラムがどちらのテーブルにも少なくとも 1 つは存在する必要があります。共通するカラム名がなければ、エラーが発生します。

テーブル A と B が共通のカラム名を 1 つ持っており、そのカラムが x であるとします。その場合は次のようになります。

```
SELECT *
FROM A NATURAL JOIN B;
```

これは、次のクエリと同義です。

```
SELECT *
FROM A JOIN B
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
 ON A.x = B.x;
```

テーブル A と B が共通のカラム名を 2 つ持っており、そのカラムが a と b である場合、`A NATURAL JOIN B` は次のクエリと同等です。

```
A JOIN B
 ON A.a = B.a
 AND A.b = B.b;
```

例

例 1

たとえば、テーブル Employees と Departments には共通のカラム名 DepartmentID が 1 つあるため、ナチュラルジョインを使用してこの 2 つのテーブルをジョインできます。

```
SELECT GivenName, Surname, DepartmentName
FROM Employees NATURAL JOIN Departments
ORDER BY DepartmentName, Surname, GivenName;
```

GivenName	Surname	DepartmentName
Janet	Bigelow	Finance
Kristen	Coe	Finance
James	Coleman	Finance
Jo Ann	Davidson	Finance
...	...	...

次の文は同義です。この文では、さきほどの例で生成されたジョイン条件が明示的に指定されています。

```
SELECT GivenName, Surname, DepartmentName
FROM Employees JOIN Departments
  ON (Employees.DepartmentID = Departments.DepartmentID)
ORDER BY DepartmentName, Surname, GivenName;
```

例 2

Interactive SQL で次のクエリを実行します。

```
SELECT Surname, DepartmentName
FROM Employees NATURAL JOIN Departments;
```

Surname	DepartmentName
Whitney	R & D
Cobb	R & D
Breault	R & D
Shishov	R & D
Driscoll	R & D
...	...

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

この例では、データベースサーバが 2 つのテーブルを参照し、共通するカラム名は DepartmentID であると判断します。次の ON CLAUSE は内部的に生成され、ジョインの実行に使用されます。

```
FROM Employees JOIN Departments
   ON Employees.DepartmentID = Departments.DepartmentID
```

NATURAL JOIN は ON 句を入力するための単なるショートカットで、この 2 つのクエリは同じです。

このセクションの内容:

NATURAL JOIN を使用した場合のエラー [404 ページ]
NATURAL JOIN 演算子は、等価ではないカラムを等価と見なすと問題が発生する可能性があります。

ON 句を使用したナチュラルジョイン [405 ページ]
NATURAL JOIN を指定し、かつ ON 句内にジョイン条件を置くと、2 つのジョイン条件の論理積が生成されます。

テーブル式のナチュラルジョイン [405 ページ]
NATURAL JOIN の少なくともどちらか一方の側に複数テーブルの式が 1 つ存在する場合、データベースサーバは、ジョイン演算子の左右にあるカラムを比較して、同じ名前のカラムを検索することでジョイン条件を生成します。

ビューと派生テーブルのナチュラルジョイン [406 ページ]
NATURAL JOIN の左右どちらにもビューまたは派生テーブルを指定できます。これは、ANSI/ISO SQL 標準の拡張機能です。

1.3.5.7.1 NATURAL JOIN を使用した場合のエラー

NATURAL JOIN 演算子は、等価ではないカラムを等価と見なすと問題が発生する可能性があります。

たとえば、次のクエリを実行すると、意図しない結果が生成されます。

```
SELECT *
FROM SalesOrders NATURAL JOIN Customers;
```

このクエリを実行してもローは返されません。データベースサーバは、内部的に次の ON 句を生成します。

```
FROM SalesOrders JOIN Customers
   ON SalesOrders.ID = Customers.ID
```

SalesOrders テーブル内の ID カラムは注文の ID 番号です。Customers テーブル内の ID カラムは顧客の ID 番号です。これらはどれも一致しません。もちろん、一致があったとしても意味がありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.5.7.2 ON 句を使用したナチュラルジョイン

NATURAL JOIN を指定し、かつ ON 句内にジョイン条件を置くと、2 つのジョイン条件の論理積が生成されます。

たとえば、次の 2 つのクエリは同義です。最初のクエリでは、データベースサーバはジョイン条件 `Employees.DepartmentID = Departments.DepartmentID` を生成します。このクエリには明示的ジョイン条件も含まれています。

```
SELECT GivenName, Surname, DepartmentName
FROM Employees NATURAL JOIN Departments
  ON Employees.ManagerID = Departments.DepartmentHeadID;
```

次のクエリは同義です。このクエリでは、前の例で生成されたナチュラルジョイン条件が ON 句で指定されています。

```
SELECT GivenName, Surname, DepartmentName
FROM Employees JOIN Departments
  ON Employees.ManagerID = Departments.DepartmentHeadID
   AND Employees.DepartmentID = Departments.DepartmentID;
```

1.3.5.7.3 テーブル式のナチュラルジョイン

NATURAL JOIN の少なくともどちらか一方の側に複数テーブルの式が 1 つ存在する場合、データベースサーバは、ジョイン演算子の左右にあるカラムを比較して、同じ名前のカラムを検索することでジョイン条件を生成します。

次の文を例にとります。

```
SELECT *
FROM (A JOIN B) NATURAL JOIN (C JOIN D);
```

この文には 2 つのテーブル式があります。テーブル式 `A JOIN B` のカラム名がテーブル式 `C JOIN D` のカラム名と比較されると、一致するカラム名のうちあいまいさのないペアに対してジョイン条件が生成されます。一致するカラム名のうちあいまいさのないペアとは、カラム名が両方のテーブル式に出現することがあっても同一テーブル式内では 2 回出現しないという意味です。

あいまいなカラム名のペアが 1 つでもあると、エラーになります。ただし、カラム名がもう一方のテーブル式内のカラム名とも一致しなければ、カラム名が同じテーブル式で 2 回出現しても構いません。

ナチュラルジョインリスト

ナチュラルジョインの少なくとも片方の側にテーブル式のリストがある場合、そのリスト内の各テーブル式に対して別のジョイン条件が生成されます。

次のテーブルを考えてみます。

- テーブル A はカラム a、b、c で構成されています。
- テーブル B は、カラム a と d で構成されています。
- テーブル C はカラム d と c で構成されています。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

このような場合、データベースサーバではジョイン (A,B) NATURAL JOIN C によって次の 2 つのジョイン条件が生成されます。

```
ON A.c = C.c
 AND B.d = C.d
```

A-C または B-C に共通のカラム名がなければ、エラーが発行されます。

テーブル C がカラム a、d、c で構成されている場合、ジョイン (A,B) NATURAL JOIN C は無効です。その理由は、カラム a が 3 つのテーブルすべてに出現するため、ジョインがあいまいになってしまうためです。

例

次の例は、"販売した製品と販売担当者の情報を売り上げ別に提供してほしい" という質問に対する回答です。

```
SELECT *
FROM ( Employees KEY JOIN SalesOrders )
  NATURAL JOIN ( SalesOrderItems KEY JOIN Products );
```

これは次と同じです。

```
SELECT *
FROM ( Employees KEY JOIN SalesOrders )
  JOIN ( SalesOrderItems KEY JOIN Products )
    ON SalesOrders.ID = SalesOrderItems.ID;
```

1.3.5.7.4 ビューと派生テーブルのナチュラルジョイン

NATURAL JOIN の左右どちらにもビューまたは派生テーブルを指定できます。これは、ANSI/ISO SQL 標準の拡張機能です。

次の文を考えてみます。

```
SELECT *
FROM View1 NATURAL JOIN View2;
```

View1 内のカラムは View2 内のカラムと比較されます。たとえば、両方のビューにカラム EmployeeID が出現し、それと同じ名前を持つカラムがほかになければ、生成されるジョイン条件は (View1.EmployeeID = View2.EmployeeID) になります。

例

次の例で、ナチュラルジョインに使用するビューにはカラムだけでなく式を指定することができ、これらの式やカラムはナチュラルジョインでは同じように扱われることを説明します。まず、カラム x のあるビュー V を次のように作成します。

```
CREATE VIEW V(x) AS
SELECT R.y + 1
FROM R;
```

次に、このビューから派生テーブルへのナチュラルジョインを作成します。派生テーブルには、カラム x があり、相関名 T が付けられています。

```
SELECT *
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM V NATURAL JOIN (SELECT P.y FROM P) as T(x);
```

このジョインは、次のクエリと同義です。

```
SELECT *
FROM V JOIN (SELECT P.y FROM P) as T(x) ON (V.x = T.x);
```

1.3.5.8 キージョイン

一般的なジョインの多くは 2 つのテーブル間で外部キーによって関連付けられます。

最も一般的なジョインは、外部キー値がプライマリキー値と同じになるように制限します。KEY JOIN 演算子は、外部キーの関係に基づいて 2 つのテーブルをジョインします。つまり、データベースサーバは、一方のテーブルのプライマリキーカラムを他方のテーブルの外部キーカラムと同等とする ON 句を生成します。キージョインを使用するには、テーブル間に外部キー関係が必要になります。この関係がない場合は、エラーになります。

キージョインは ON 句のショートカットで、この 2 つのクエリは同じです。ただし、ON 句は KEY JOIN でも使用できます。JOIN を指定しても CROSS、NATURAL、KEY を指定しない場合、または ON 句を使用する場合のデフォルトは、キージョインです。SQL Anywhere サンプルデータベースの図では、テーブル間を結ぶ線は外部キーを表します。KEY JOIN 演算子は、図の中で 1 本の線によって 2 つのテーブルがジョインされているところならどこでも使用できます。

キージョインがデフォルトの場合

次のすべてが当てはまる場合、キージョインがデフォルトになります。

- キーワード JOIN が使用されています。
- キーワード CROSS、NATURAL、または KEY が指定されていません。
- ON 句がありません。

例

たとえば、次のクエリは、データベース内の外部キー関係に基づいてテーブル Products と SalesOrderItems をジョインします。

```
SELECT *
FROM Products KEY JOIN SalesOrderItems;
```

次のクエリは同義です。これには KEY がありませんが、ON 句のない JOIN はデフォルトで KEY JOIN になります。

```
SELECT *
FROM Products JOIN SalesOrderItems;
```

次のクエリも同義になります。ON 句で指定されているジョイン条件が、データベースサーバが SQL Anywhere サンプルデータベースの外部キー関係に基づいてこれらのテーブルに対して生成するジョイン条件と同じになるためです。

```
SELECT *
FROM Products JOIN SalesOrderItems
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ON SalesOrderItems.ProductID = Products.ID;
```

このセクションの内容:

ON 句を使用したキージョイン [408 ページ]
KEY JOIN を指定し、かつ ON 句内にジョイン条件を置くと、2 つのジョイン条件の論理積が生成されます。

複数の外部キー関係がある場合のキージョイン [408 ページ]
データベースサーバによって、外部キー関係に基づいてジョイン条件が生成されるときに、外部キー関係が 2 つ以上ある場合があります。

テーブル式のキージョイン [411 ページ]
データベースサーバは、文中にあるテーブルのペアごとに外部キー関係を調べることで、テーブル式のキージョインに対してジョイン条件を生成します。

ビューと派生テーブルのキージョイン [416 ページ]
キージョインにビューまたは派生テーブルが含まれている場合、データベースサーバではテーブルと同じ基本手順に従います。ただし、次の点が異なります。

キージョイン操作規則 [418 ページ]
キージョインの操作を表すいくつかのルールがあります。

1.3.5.8.1 ON 句を使用したキージョイン

KEY JOIN を指定し、かつ ON 句内にジョイン条件を置くと、2 つのジョイン条件の論理積が生成されます。

次に例を示します。

```
SELECT *
FROM A KEY JOIN B
ON A.x = B.y;
```

A と B のキージョインによって生成されたジョイン条件が `A.w = B.z` の場合、このクエリは次のクエリと同等です。

```
SELECT *
FROM A JOIN B
ON A.x = B.y AND A.w = B.z;
```

1.3.5.8.2 複数の外部キー関係がある場合のキージョイン

データベースサーバによって、外部キー関係に基づいてジョイン条件が生成されるときに、外部キー関係が 2 つ以上ある場合があります。

このような場合、データベースサーバは、外部キーが参照するプライマリキーテーブルの相関名と、外部キーの役割名とを一致させることによって、使用する外部キー関係を決定します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

相関名と役割名

相関名とは、クエリの FROM 句内で使用されるテーブル名またはビュー名のことです。元の名前か、FROM 句で定義されたエイリアスになります。

役割名は外部キーの名前です。役割名は、指定された外部 (子) テーブルに対してユニークでなければなりません。

外部キーに役割名を指定しない場合は、次のようにして名前が割り当てられます。

- プライマリテーブル名と同じ名前の外部キーが存在しない場合は、役割名にはプライマリテーブル名が割り当てられます。
- すでにプライマリテーブル名が別の外部キーによって使用されている場合は、役割名は、外部テーブルに対してユニークな、プライマリテーブル名と 0 埋め込みの 3 桁の数字が連結されたものになります。

外部キーの役割名がわからない場合は、SQL Central で左ウィンドウ枠にあるデータベースコンテナを展開すると検索できます。左ウィンドウ枠でテーブルを選択し、右ウィンドウ枠で 制約タブをクリックします。右ウィンドウ枠にテーブルの外部キーのリストが表示されます。

ジョイン条件の生成

データベースサーバでは、次のように、プライマリキーテーブルの相関名と同じ役割名を持つ外部キーが検索されます。

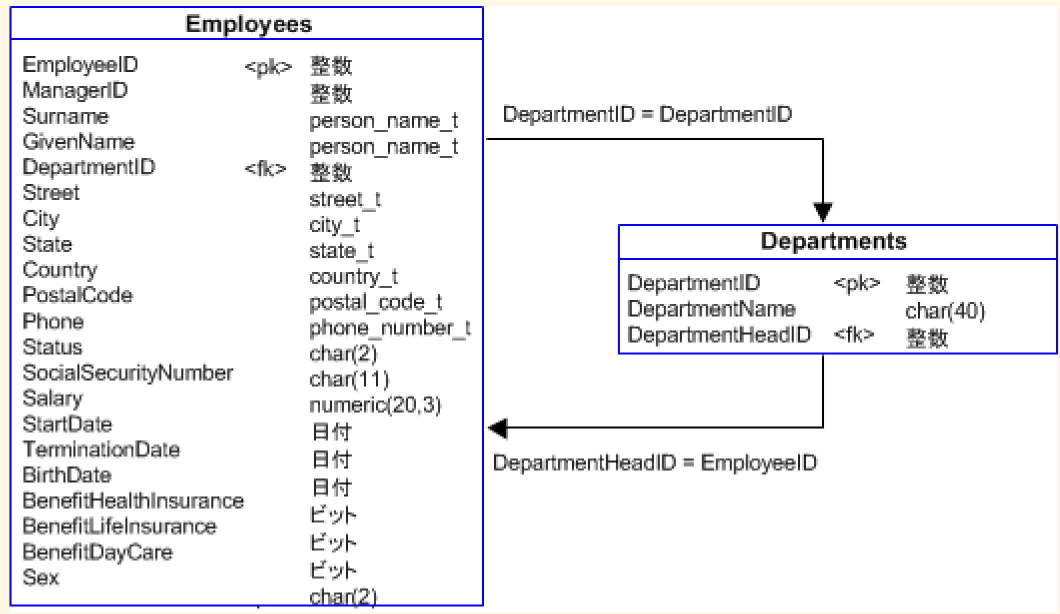
- ジョイン内のテーブルと同じ名前の外部キーが 1 つだけであれば、データベースサーバではその外部キーが使用され、ジョイン条件が生成されます。
- テーブルと同じ名前の外部キーが 2 つ以上見つかると、そのジョインはあいまいとなりエラーが発行されます。
- テーブルと同じ名前の外部キーが 1 つもなければ、名前が一致しなくてもデータベースサーバで外部キー関係が検索されます。外部キー関係が 2 つ以上見つかれば、そのジョインはあいまいとなりエラーが発行されます。

例

例 1

SQL Anywhere サンプルデータベースでは、テーブル Employees と Departments の間で 2 つの外部キー関係が定義されています。Employees テーブルの外部キー FK_DepartmentID_DepartmentID は Departments テーブルを参照し、Departments テーブルの外部キー FK_DepartmentHeadID_EmployeeID は Employees テーブルを参照しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.



次のクエリはあいまいです。外部キー関係が 2 つあり、そのどちらにもプライマリキーテーブル名と同じ役割名が指定されていないからです。そのため、このクエリを実行しようとしても、結果は構文エラー `SQLE_AMBIGUOUS_JOIN (-147)` になります。

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees KEY JOIN Departments;
```

例 2

このクエリは、Departments テーブルに相関名 FK_DepartmentID_DepartmentID を指定して例 1 のクエリを修正したものです。ここで、外部キー FK_DepartmentID_DepartmentID にはその参照テーブルと同じ名前が付けられているため、ジョイン条件を定義するために使用されます。結果には、すべての従業員の姓と所属部署が入っています。

```
SELECT Employees.Surname,
    FK_DepartmentID_DepartmentID.DepartmentName
FROM Employees KEY JOIN Departments
    AS FK_DepartmentID_DepartmentID;
```

次のクエリは上の例と同義です。この例では、Departments テーブルのエイリアスを作成する必要はありません。このクエリでは、上の例で生成されたジョイン条件と同じものが ON 句で指定されています。

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees JOIN Departments
   ON Departments.DepartmentID = Employees.DepartmentID;
```

例 3

ある部署の責任者である従業員をすべてリストする場合は、外部キー FK_DepartmentHeadID_EmployeeID を使用し、例 1 を次のように書き換えます。このクエリは、プライマリキーテーブル Employees に相関名 FK_DepartmentHeadID_EmployeeID を指定することで外部キー FK_DepartmentHeadID_EmployeeID を使用します。

```
SELECT FK_DepartmentHeadID_EmployeeID.Surname, Departments.DepartmentName
FROM Employees AS FK_DepartmentHeadID_EmployeeID
    KEY JOIN Departments;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のクエリは上の例と同義です。このクエリでは、上の例で生成されたジョイン条件が ON 句で指定されています。

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees JOIN Departments
   ON Departments.DepartmentHeadID = Employees.EmployeeID;
```

例 4

外部キーの役割名がプライマリキーテーブル名と同じ場合、相関名は不要です。たとえば、次のように、Employees テーブルに外部キー Departments を定義するとします。

```
ALTER TABLE Employees
   ADD FOREIGN KEY Departments (DepartmentID)
   REFERENCES Departments (DepartmentID);
```

ここで、KEY JOIN が 2 つのテーブル間で指定されていれば、この外部キー関係がデフォルトのジョイン条件になります。外部キー Departments が定義されていれば、次のクエリは例 3 と同義になります。

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees KEY JOIN Departments;
```

注記

この例を Interactive SQL で実行する場合は、次の文を使って SQL Anywhere サンプルデータベースへの変更を取り消す必要があります。

```
ALTER TABLE Employees DROP FOREIGN KEY Departments;
```

関連情報

キージョイン操作規則 [418 ページ]

1.3.5.8.3 テーブル式のキージョイン

データベースサーバは、文中にあるテーブルのペアごとに外部キー関係を調べることで、テーブル式のキージョインに対してジョイン条件を生成します。

次の例では 4 つのペアのテーブルがジョインされています。

```
SELECT *
FROM (A NATURAL JOIN B) KEY JOIN (C NATURAL JOIN D);
```

テーブルのペアは A-C、A-D、B-C、B-D です。データベースサーバはそれぞれのペアの関係を調べてから、テーブル式全体に対して生成されるジョイン条件を作成します。処理方法はテーブル式にカンマを使用するかどうかによって異なります。したがって、次の 2 つの例では生成されたジョイン条件が異なります。`A JOINB` はカンマを含まないテーブル式 であり、`(A,B)` はテーブル式のリストです。

```
SELECT *
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM (A JOIN B) KEY JOIN C;
```

この例は、セマンティック上、次の例と異なります。

```
SELECT *
FROM (A,B) KEY JOIN C;
```

このセクションの内容:

カンマを含まないテーブル式のキージョイン [412 ページ]
ジョインされている 2 つのテーブル式のどちらにもカンマが含まれていない場合、データベースサーバは文中にあるテーブルのペアの外部キー関係を調べ、ジョイン条件を 1 つだけ生成します。

テーブル式リストのキージョイン [413 ページ]
2 つのテーブル式リストのキージョインに対してジョイン条件を生成する場合、データベースサーバは文中のテーブルのペアを調べて、各ペアに対してジョイン条件を生成します。

カンマを含まないテーブル式とリストのキージョイン [415 ページ]
テーブル式リストがカンマを含まないテーブル式を持つキージョインを介してジョインされる場合、データベースサーバはテーブル式リストの各テーブルに対してジョイン条件を生成します。

関連情報

複数の外部キー関係がある場合のキージョイン [408 ページ]

1.3.5.8.3.1 カンマを含まないテーブル式のキージョイン

ジョインされている 2 つのテーブル式のどちらにもカンマが含まれていない場合、データベースサーバは文中にあるテーブルのペアの外部キー関係を調べ、ジョイン条件を 1 つだけ生成します。

たとえば、次のジョインには A-C と B-C という 2 つのテーブルペアがあります。

```
(A NATURAL JOIN B) KEY JOIN C
```

C と `(A NATURAL JOIN B)` をジョインするために、データベースサーバはテーブルペア A-C と B-C の外部キー関係を調べて、ジョイン条件を 1 つだけ生成します。複数の外部キー関係が存在する場合は、次のキージョイン決定規則に基づき、この 2 つのペアに対して 1 つのジョイン条件を生成します。

- まず、参照先となるプライマリキーテーブルのうち、その 1 つの相関名と同じ役割名を持つ単一の外部キーを指定するために A-C および B-C の両方が調べられます。この基準に合う外部キーが 1 つだけ存在する場合には、それが使用されます。テーブルの相関名と同じ役割名の外部キーが 2 つ以上ある場合、そのジョインはあいまいと見なされてエラーが発行されます。
- テーブルの相関名と同じ名前の外部キーがない場合は、そのテーブルの外部キー関係がデータベースサーバで検索されます。見つかった外部キー関係が 1 つであれば、それが使用されます。2 つ以上見つかると、そのジョインはあいまいと見なされてエラーが発行されます。
- 外部キー関係がまったく存在しない場合は、エラーが発行されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

次の例は、営業部員とその部署をすべて検索するクエリです。

```
SELECT Employees.Surname,
        FK_DepartmentID_DepartmentID.DepartmentName
FROM ( Employees KEY JOIN Departments
        AS FK_DepartmentID_DepartmentID )
    KEY JOIN SalesOrders;
```

このクエリは次のように解釈できます。

- データベースサーバはテーブル式 (Employees KEY JOIN Departments as FK_DepartmentID_DepartmentID) を調べ、外部キー FK_DepartmentID_DepartmentID に基づいて、ジョイン条件 Employees.DepartmentID = FK_DepartmentID_DepartmentID.DepartmentID を生成します。
- 次に Employees/SalesOrders と Departments/SalesOrders のテーブルペアを調べます。テーブル SalesOrders と Employees 間、および、テーブル SalesOrders と Departments 間に存在できる外部キーは 1 つだけです。それ以外の場合は、ジョインはあいまいになります。この場合、テーブル SalesOrders と Employees 間には外部キー関係が 1 つだけ存在し (FK_SalesRepresentative_EmployeeID)、テーブル SalesOrders と Departments 間に外部キーは存在しません。したがって、ジョイン条件 SalesOrders.EmployeeID = Employees.SalesRepresentative が生成されます。

したがって、次のクエリは上のクエリと同義です。

```
SELECT Employees.Surname, Departments.DepartmentName
FROM ( Employees JOIN Departments
    ON ( Employees.DepartmentID = Departments.DepartmentID ) )
JOIN SalesOrders
    ON ( Employees.EmployeeID = SalesOrders.SalesRepresentative );
```

1.3.5.8.3.2 テーブル式リストのキージョイン

2 つのテーブル式リストのキージョインに対してジョイン条件を生成する場合、データベースサーバは文中のテーブルのペアを調べて、各ペアに対してジョイン条件を生成します。

最後のジョイン条件は各ペアのジョイン条件の論理積です。各ペア間には外部キー関係が存在する必要があります。

次の例では、2 つのテーブルペア A-C と B-C をジョインします。

```
SELECT *
FROM ( A,B ) KEY JOIN C;
```

データベースサーバでは、2 つのテーブルペア A-C と B-C のそれぞれに対してジョイン条件を生成することで、C と (A,B) をジョインするためのジョイン条件が生成されます。このように複数の外部キー関係が存在する場合は、次のキージョイン規則に従って処理されます。

- 各ペアに対して、プライマリキーテーブルの相関名と同じ名前の役割名を持つ外部キーがデータベースサーバで検索されます。この基準に合う外部キーが 1 つだけ存在する場合には、それが使用されます。2 つ以上見つかると、そのジョインはあいまいと見なされてエラーが発行されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 各ペアに、テーブルの相関名と同じ名前の外部キーがない場合は、そのテーブル間の外部キー関係がデータベースサーバで検索されます。見つかった外部キー関係が 1 つであれば、それが使用されます。2 つ以上見つかると、そのジョインはあいまいと見なされてエラーが発行されます。
- 各ペアに、外部キー関係がまったく存在しない場合は、エラーが発行されます。
- データベースサーバで各ペアにジョイン条件を 1 つだけ決定できる場合は、AND によってジョイン条件が組み合わされます。

例

次は、ある地域で 1 つ以上の注文を受けたすべての営業部員の名前を返すクエリです。

```
SELECT DISTINCT Employees.Surname,
        FK_DepartmentID_DepartmentID.DepartmentName,
        SalesOrders.Region
FROM ( SalesOrders, Departments
        AS FK_DepartmentID_DepartmentID )
    KEY JOIN Employees;
```

Surname	DepartmentName	Region
Chin	Sales	Eastern
Chin	Sales	Western
Chin	Sales	Central
...	...	...

このクエリでは、テーブルの 2 つのペア、SalesOrders と Employees、および Departments AS FK_DepartmentID_DepartmentID と Employees を扱います。

SalesOrders と Employees のペアには、一方のテーブルと同じ役割名の外部キーはありません。ただし、2 つのテーブルに関連した外部キー (FK_SalesRepresentative_EmployeeID) が 1 つあります。これは、2 つのテーブルに関連する唯一の外部キーであり、この外部キーが使用されて、生成されたジョイン条件 `( Employees.EmployeeID = SalesOrders.SalesRepresentative )` になります。

Departments AS FK_DepartmentID_DepartmentID と Employees のペアでは、プライマリキーテーブルと同じ役割名を持つ外部キーは 1 つです。そのキーは FK_DepartmentID_DepartmentID で、クエリ内の Departments テーブルに指定した相関名と一致します。プライマリキーテーブルの相関名と同じ名前の外部キーは他にないため、このテーブルペアのジョイン条件は FK_DepartmentID_DepartmentID を使用して作成されます。生成されるジョイン条件は `(Employees.DepartmentID = FK_DepartmentID_DepartmentID.DepartmentID)` です。2 つのテーブルに関連する外部キーはもう 1 つありますが、この外部キーの名前はどちらのテーブルの名前とも異なるため、要因にはなりません。

最後のジョイン条件は、それぞれのテーブルペアに対して生成されたジョイン条件を 1 つにまとめます。したがって、次のクエリは同義です。

```
SELECT DISTINCT Employees.Surname,
   Departments.DepartmentName,
   SalesOrders.Region
FROM ( SalesOrders, Departments )
    JOIN Employees
    ON Employees.EmployeeID = SalesOrders.SalesRepresentative
    AND Employees.DepartmentID = Departments.DepartmentID;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

複数の外部キー関係がある場合のキージョイン [408 ページ]

1.3.5.8.3.3 カンマを含まないテーブル式とリストのキージョイン

テーブル式リストがカンマを含まないテーブル式を持つキージョインを介してジョインされる場合、データベースサーバはテーブル式リストの各テーブルに対してジョイン条件を生成します。

たとえば、次の文は、テーブル式リストと、カンマを含まないテーブル式のキージョインです。この例では、テーブル式 C NATURAL JOIN D が指定されたテーブル A と、テーブル式 C NATURAL JOIN D が指定されたテーブル B のジョイン条件が生成されます。

```
SELECT *
FROM (A,B) KEY JOIN (C NATURAL JOIN D);
```

(A,B) はテーブル式のリストで、C NATURAL JOIN D はテーブル式です。したがって、データベースサーバは 2 つのジョイン条件を生成する必要があります。1 つは A-C と A-D ペアのジョイン条件で、もう 1 つは B-C と B-D ペアのジョイン条件です。このように外部キー関係が複数存在する場合は、次のキージョイン規則に従って処理されます。

- テーブルペアの各セットに対して、プライマリキーテーブルの 1 つの相関名と同じ名前の役割名を持つ外部キーがデータベースサーバで検索されます。この基準に合う外部キーが 1 つだけ存在する場合には、それが使用されます。2 つ以上見つかると、そのジョインはあいまいになりエラーが発行されます。
- テーブルペアの各セットに、テーブルの相関名と同じ名前の外部キーがない場合は、そのテーブル間の外部キー関係がデータベースサーバで検索されます。存在する外部キー関係が 1 つだけであれば、それが使用されます。2 つ以上見つかると、そのジョインはあいまいになりエラーが発行されます。
- テーブルペアの各セットに、外部キー関係がまったく存在しない場合は、エラーが発行されます。
- データベースサーバで各ペアにジョイン条件を 1 つだけ決定できる場合は、キーワード AND によってジョイン条件が組み合わされます。

例

例 1 - 次の 5 つのテーブルのジョインを考えてみます。

```
((A,B) JOIN (C NATURAL JOIN D) ON A.x = D.y) KEY JOIN E
```

この場合、(A,B) と E の間の条件、または C NATURAL JOIN D と E の間の条件のどちらか 1 つが生成されることで、E に対するキージョイン条件が生成されます。

(A,B) と E の間にジョイン条件が生成される場合は、A-E と B-E に 1 つずつ、合計 2 つのジョイン条件が作成される必要があります。それぞれのテーブルペア内では有効な外部キー関係が検出される必要があります。

C NATURAL JOIN D と E の間にジョイン条件が作成される場合、ジョイン条件は 1 つだけ作成されるため、C-E と D-E のペア内で検出される必要がある外部キー関係は 1 つだけです。

例 2 - 次は、テーブル式とテーブル式リストのキージョインの例です。この例では、営業担当兼管理者である従業員の名前と部署を示します。

```
SELECT DISTINCT Employees.Surname,
        FK_DepartmentID_DepartmentID.DepartmentName
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM ( SalesOrders, Departments
        AS FK_DepartmentID_DepartmentID )
    KEY JOIN ( Employees JOIN Departments AS d
        ON Employees.EmployeeID = d.DepartmentHeadID );
```

データベースサーバは 2 つのジョイン条件を生成します。

- テーブルペア SalesOrders/Employees と SalesOrders/d の間には外部キー関係が 1 つだけ存在する (`SalesOrders.SalesRepresentative = Employees.EmployeeID`)。
- テーブルペア FK_DepartmentID_DepartmentID/Employees と FK_DepartmentID_DepartmentID/d の間には外部キー関係が 1 つだけ存在する (`FK_DepartmentID_DepartmentID.DepartmentID = Employees.DepartmentID`)。

この例は次の文と同義です。次の例では、相関名 `Departments AS FK_DepartmentID_DepartmentID` を作成する必要はありません。相関名が必要になるのは、Employees と Departments のジョインに使用する 2 つの外部キーを明示する場合だけです。

```
SELECT DISTINCT Employees.Surname,
   Departments.DepartmentName
FROM ( SalesOrders, Departments )
    JOIN ( Employees JOIN Departments AS d
       ON Employees.EmployeeID = d.DepartmentHeadID )
    ON SalesOrders.SalesRepresentative = Employees.EmployeeID
       AND Departments.DepartmentID = Employees.DepartmentID;
```

関連情報

テーブル式リストのキージョイン [413 ページ]

1.3.5.8.4 ビューと派生テーブルのキージョイン

キージョインにビューまたは派生テーブルが含まれている場合、データベースサーバではテーブルと同じ基本手順に従います。ただし、次の点が異なります。

- 各キージョインについて、クエリとビューの FROM 句内のテーブルのペアが調べられ、すべてのペアのセットに対してジョイン条件が 1 つ生成されます。この場合、ビュー内の FROM 句にカンマやジョインのキーワードが含まれているかどうかは考慮されません。
- ビューまたは派生テーブルの相関名と同じ役割名を持つ外部キーに基づいてテーブルがジョインされます。
- キージョイン内にビューまたは派生テーブルが含まれる場合、ビューまたは派生テーブル定義には UNION、INTERSECT、EXCEPT、ORDER BY、DISTINCT、GROUP BY などの集合関数や TOP、FIRST、START AT、FOR XML などの Window 関数を含めることはできません。これらが 1 つでも入っていると、エラーが返されます。さらに、派生テーブルは再帰テーブル式として定義することはできません。
  派生テーブルとビューの機能は同じです。派生テーブルの場合、定義済みのビューを参照しないで、テーブルの定義を文中に記述する点だけが異なります。

例

例 1

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の文では、View1 がビューです。

```
SELECT *
FROM View1 KEY JOIN B;
```

View1 の定義は次のいずれかになるため、結果的に、B に対して同じジョイン条件になります (結果セットは異なりますが、ジョイン条件は同じです)。

```
SELECT *
FROM C CROSS JOIN D;
```

```
SELECT *
FROM C,D;
```

```
SELECT *
FROM C JOIN D ON (C.x = D.y);
```

いずれの場合も、View1 と B のキージョインに対してジョイン条件を生成するには、テーブルペア C-B と D-B が調べられ、ジョイン条件が 1 つだけ生成されます。ジョイン条件は複数の外部キー関係に関する規則に基づいて生成されます。ただし、ビューの参照テーブルではなく、ビューの相関名と同じ名前の外部キーを検索することが異なります。

前述のビュー定義のいずれかを使用すると、`View1 KEY JOIN B` の処理を次のように解釈できます。

テーブルペア C-B と D-B が調べられ、ジョイン条件が 1 つだけ生成されます。複数の外部キー関係が存在する場合には、次のキージョイン決定規則に基づいてジョイン条件が生成されます。

- まず、ビューの相関名と同じ役割名を持つ 1 つの外部キーに対して C-B と D-B が両方とも調べられます。この基準に合う外部キーが 1 つだけ存在する場合には、それが使用されます。ビューの相関名と同じ役割名の外部キーが 2 つ以上ある場合、そのジョインはあいまいと見なされてエラーが発行されます。
- ビューの相関名と同じ名前の外部キーがない場合は、そのテーブル間の外部キー関係がデータベースサーバで検索されます。見つかった外部キー関係が 1 つであれば、それが使用されます。2 つ以上見つかると、そのジョインはあいまいと見なされてエラーが発行されます。
- 外部キー関係がまったく存在しない場合は、エラーが発f行されます。

ここで生成されたジョイン条件が `B.y = D.z` であると想定します。次に示す元のジョインを展開できます。たとえば、次の 2 つの文は同じです。

```
SELECT *
FROM View1 KEY JOIN B;
SELECT *
FROM View1 JOIN B ON B.y = View1.z;
```

例 2

次の例は、各部署の管理者に関する全従業員情報を含むビューです。

```
CREATE VIEW V AS
SELECT Departments.DepartmentName, Employees.*
FROM Employees JOIN Departments
  ON Employees.EmployeeID = Departments.DepartmentHeadID;
```

次のクエリはテーブル式に対するビューをジョインします。

```
SELECT *
FROM V KEY JOIN ( SalesOrders,
   Departments FK_DepartmentID_DepartmentID );
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のクエリは前述のクエリと同等です。

```
SELECT *
FROM V JOIN ( SalesOrders,
   Departments FK_DepartmentID_DepartmentID )
ON ( V.EmployeeID = SalesOrders.SalesRepresentative
AND V.DepartmentID =
    FK_DepartmentID_DepartmentID.DepartmentID );
```

関連情報

再帰共通テーブル式 [427 ページ]
複数の外部キー関係がある場合のキージョイン [408 ページ]
テーブル式のキージョイン [411 ページ]

1.3.5.8.5 キージョイン操作規則

キージョインの操作を表すいくつかのルールがあります。

規則 1: 2 つのテーブルのキージョイン

この規則は、`A KEY JOIN B` に適用されます。ここで、A と B はベーステーブルまたはテンポラリテーブルです。

1. B を参照する A のすべての外部キーを見つけます。
   テーブル B の相関名が役割名になる外部キーが存在する場合には、それを優先外部キーとします。
2. A を参照する B のすべての外部キーを見つけます。
   テーブル A の相関名が役割名になる外部キーが存在する場合には、それを優先外部キーとします。
3. 優先キーが 2 つ以上存在する場合、そのジョインはあいまいです。構文エラー `SQLE_AMBIGUOUS_JOIN (-147)` が発行されます。
4. 優先キーが 1 つだけ存在する場合には、この KEY JOIN 式に生成されたジョイン条件を定義するために、この外部キーが選択されます。
5. 優先キーがまったくない場合は、A と B 間にある他の外部キーが使用されます。
   - A と B の間に外部キーが 2 つ以上ある場合、そのジョインはあいまいです。構文エラー `SQLE_AMBIGUOUS_JOIN (-147)` が発行されます。
   - 外部キーが 1 つだけ存在する場合には、この KEY JOIN 式に生成されたジョイン条件を定義するために、この外部キーが選択されます。
   - 外部キーがまったくない場合、ジョインは無効であり、エラーが生成されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

規則 2: カンマを含まないテーブル式のキージョイン

この規則は、`A KEY JOIN B` に適用されます。ここで、A と B はカンマを含まないテーブル式です。

1. テーブルの各ペア (テーブル式 A から 1 つとテーブル式 B から 1 つ) について、すべての外部キーをリストし、テーブル間のすべての優先キーにマークします。優先キー決定規則は上記の規則 1 で指定されています。
2. 優先キーが 2 つ以上存在する場合、そのジョインはあいまいです。構文エラー `SQLE_AMBIGUOUS_JOIN (-147)` が発行されます。
3. 優先キーが 1 つだけ存在する場合には、この KEY JOIN 式に生成されたジョイン条件を定義するために、この外部キーが選択されます。
4. 優先キーがまったくない場合は、テーブルのペア間にある他の外部キーが使用されます。
    - 外部キーが 2 つ以上存在する場合、そのジョインはあいまいです。構文エラー `SQLE_AMBIGUOUS_JOIN (-147)` が発行されます。
    - 外部キーが 1 つだけ存在する場合には、この KEY JOIN 式に生成されたジョイン条件を定義するために、この外部キーが選択されます。
    - 外部キーがまったくない場合、ジョインは無効であり、エラーが生成されます。

規則 3: テーブル式リストのキージョイン

この規則は、`(A1, A2, ...) KEY JOIN ( B1, B2, ...)` に適用されます。ここで、A1、B1 などはカンマを含まないテーブル式です。

1. テーブル式 Ai と Bj の各ペアに対して、規則 1 または規則 2 を適用し、テーブル式 `(Ai KEY JOIN Bj)` 用にユニークに生成されたジョイン条件を検出します。テーブル式のペアの KEY JOIN のいずれかが規則 1 または規則 2 によってあいまいであると判断されると、構文エラーになります。
2. この KEY JOIN 式の生成されたジョイン条件は、手順 1 のジョイン条件の論理積です。

規則 4: カンマを含まないテーブル式とリストのキージョイン

この規則は、`(A1, A2, ...) KEY JOIN ( B1, B2, ...)` に適用されます。ここで、A1、B1 などはカンマを含む可能性のあるテーブル式です。

1. テーブル式 Ai と Bj の各ペアに対して、規則 1、規則 2、または規則 3 を適用し、テーブル式 `(Ai KEY JOIN Bj)` 用にユニークに生成されたジョイン条件を検出します。テーブル式のペアの KEY JOIN のいずれかが規則 1、規則 2、または規則 3 によってあいまいであると判断されると、構文エラーになります。
2. この KEY JOIN 式の生成されたジョイン条件は、手順 1 のジョイン条件の論理積です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.6 共通テーブル式

共通テーブル式は、WITH 句を使用して定義します。WITH 句は、SELECT 文内の SELECT キーワードに先行します。

句の内容は、1 つの SELECT 文のスコープ内のみで認識される 1 つ以上のテンポラリビューを定義します。これらのテンポラリビューは、文内の他の場所から参照できます。この句の構文は、CREATE VIEW 文の構文とよく似ています。

クエリに複数の集合関数が含まれる場合や、ストアドプロシージャ内でプログラム変数を参照するビューがクエリによって定義される場合に、共通テーブル式を使用すると便利であり、共通テーブル式を使用する必要がある場合もあります。さらに、共通テーブル式は値のセットを一時的に格納する際に便利です。

例

どの部署の従業員数が最も多いかを特定する問題を例にとります。サンプルデータベースの Employees テーブルは、ある架空の会社で働くすべての従業員をリストし、それぞれがどの部署で働いているかを示します。次のクエリは、部署 ID コードと各部署の従業員の総数をリストします。

```
SELECT DepartmentID, COUNT( * ) AS n
FROM Employees
GROUP BY DepartmentID;
```

このクエリは、次に示すように従業員の最も多い部署を抽出するために使用できます。

```
SELECT DepartmentID, n
FROM ( SELECT DepartmentID, COUNT( * ) AS n
       FROM Employees
       GROUP BY DepartmentID
     ) AS a
WHERE a.n =
    ( SELECT MAX( n )
      FROM ( SELECT DepartmentID, COUNT( * ) AS n
             FROM Employees
             GROUP BY DepartmentID ) AS b
    );
```

この文は正確な結果をもたらしますが、いくつか欠点もあります。最初の欠点は、サブクエリの繰り返しによってこの文の効率が悪くなることです。2 つ目の欠点は、この文ではサブクエリ間の関係が明確でないことです。

このような問題を回避する 1 つの方法として、ビューを作成し、そのビューを使用してクエリを再作成します。この方法によって、前述の問題を回避できます。

```
CREATE VIEW CountEmployees( DepartmentID, n ) AS
    SELECT DepartmentID, COUNT( * ) AS n
    FROM Employees
    GROUP BY DepartmentID;
SELECT DepartmentID, n
    FROM CountEmployees
    WHERE n = ( SELECT MAX( n ) FROM CountEmployees );
```

この方法の欠点は、ビューの作成時にデータベースサーバがシステムテーブルを更新しなければならないため、オーバヘッドが発生することです。ビューが頻繁に使用される場合は、この方法は合理的です。ただし、ビューが特定の SELECT 文内で 1 回のみ使用される場合は、代わりに次のように共通テーブル式を使用することをおすすめします。

```
WITH CountEmployees( DepartmentID, n ) AS
    ( SELECT DepartmentID, COUNT( * ) AS n
      FROM Employees
      GROUP BY DepartmentID
    )
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
SELECT DepartmentID, n
FROM CountEmployees
WHERE n = ( SELECT MAX( n ) FROM CountEmployees );
```

また、従業員の最も少ない部署を検索するようにクエリを変更すると、クエリが複数のローを返す場合があることがわかります。

```
WITH CountEmployees( DepartmentID, n ) AS
    ( SELECT DepartmentID, COUNT( * ) AS n
      FROM Employees
      GROUP BY DepartmentID
    )
SELECT DepartmentID, n
FROM CountEmployees
WHERE n = ( SELECT MIN( n ) FROM CountEmployees );
```

サンプルデータベースでは、従業員数が最も少ない 9 人編成の部署は 2 つあります。

このセクションの内容:

複数の相関名 [421 ページ]
　共通テーブル式の複数のインスタンスに異なる相関名を付けることができます。

複数のテーブル式 [422 ページ]
　1 つの WITH 句で、2 つ以上の共通テーブル式を定義できます。

共通テーブル式を使用できる条件 [422 ページ]
　共通テーブル式はクエリ本体または任意のサブクエリ内から参照可能ですが、共通テーブル式を定義できるのは 3 か所のみです。

共通テーブル式の一般的な使用例 [423 ページ]
　共通テーブル式は、単一のクエリ内でテーブル式を複数回使用しなければならない場合に役立ちます。

再帰共通テーブル式 [427 ページ]
　再帰を使用すると、ツリーまたはツリーに似たデータ構造を表すテーブルをトラバースすることが簡単になります。

1.3.6.1 複数の相関名

共通テーブル式の複数のインスタンスに異なる相関名を付けることができます。

これによって、共通テーブル式をそれ自体にジョインできます。たとえば、次のクエリは、従業員数が同じ部署を組み合わせます。ただし、サンプルデータベースには、従業員数が同じ部は 2 つしかありません。

```
WITH CountEmployees( DepartmentID, n ) AS
    ( SELECT DepartmentID, COUNT( * ) AS n
      FROM Employees
      GROUP BY DepartmentID
    )
SELECT a.DepartmentID, a.n,
       b.DepartmentID, b.n
FROM   CountEmployees AS a
  JOIN CountEmployees AS b
  ON a.n = b.n AND a.DepartmentID < b.DepartmentID;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

複数のテーブル式 [422 ページ]
共通テーブル式を使用できる条件 [422 ページ]

1.3.6.2 複数のテーブル式

1 つの WITH 句で、2 つ以上の共通テーブル式を定義できます。

これらの定義はカンマで区切る必要があります。次の例は、支払い給与総額の最も少ない部署と、従業員数が最も多い部署をリストします。

```
WITH
    CountEmployees( DepartmentID, n ) AS
    ( SELECT DepartmentID, COUNT( * ) AS n
      FROM Employees
      GROUP BY DepartmentID
    ),
    DepartmentPayroll( DepartmentID, amount ) AS
    ( SELECT DepartmentID, SUM( Salary ) AS amount
      FROM Employees
      GROUP BY DepartmentID
    )
SELECT count.DepartmentID, count.n, pay.amount
FROM CountEmployees AS count
  JOIN DepartmentPayroll AS pay
  ON count.DepartmentID = pay.DepartmentID
WHERE count.n = ( SELECT MAX( n ) FROM CountEmployees )
   OR pay.amount = ( SELECT MIN( amount ) FROM DepartmentPayroll );
```

関連情報

複数の相関名 [421 ページ]
共通テーブル式を使用できる条件 [422 ページ]

1.3.6.3 共通テーブル式を使用できる条件

共通テーブル式はクエリ本体または任意のサブクエリ内から参照可能ですが、共通テーブル式を定義できるのは 3 か所のみです。

最上位レベルの SELECT 文

共通テーブル式は、最上位レベルの SELECT 文内で使用できますが、サブクエリ内では使用できません。

```
WITH DepartmentPayroll( DepartmentID, amount ) AS
     ( SELECT DepartmentID, SUM( Salary ) AS amount
       FROM Employees
       GROUP BY DepartmentID
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    )
SELECT DepartmentID, amount
FROM DepartmentPayroll
WHERE amount = ( SELECT MAX( amount ) FROM DepartmentPayroll );
```

ビュー定義内の最上位レベルの SELECT 文

共通テーブル式は、ビューを定義する最上位レベルの SELECT 文内で使用できますが、サブクエリ内では使用できません。

```
CREATE VIEW LargestDept ( DepartmentID, Size, pay ) AS
    WITH
        CountEmployees( DepartmentID, n ) AS
        ( SELECT DepartmentID, COUNT( * ) AS n
          FROM Employees
          GROUP BY DepartmentID
        ),
        DepartmentPayroll( DepartmentID, amount ) AS
        ( SELECT DepartmentID, SUM( Salary ) AS amount
          FROM Employees
          GROUP BY DepartmentID
        )
    SELECT count.DepartmentID, count.n, pay.amount
    FROM CountEmployees count
        JOIN DepartmentPayroll pay
        ON count.DepartmentID = pay.DepartmentID
    WHERE count.n = ( SELECT MAX( n ) FROM CountEmployees )
        OR pay.amount = ( SELECT MAX( amount ) FROM DepartmentPayroll );
```

INSERT 文内の最上位レベルの SELECT 文

共通テーブル式は、INSERT 文内の最上位レベルの SELECT 文内で使用できますが、INSERT 文内のサブクエリ内では使用できません。

```
CREATE TABLE LargestPayrolls ( DepartmentID INTEGER, Payroll NUMERIC,
CurrentDate DATE );
INSERT INTO LargestPayrolls( DepartmentID, Payroll, CurrentDate )
    WITH DepartmentPayroll( DepartmentID, amount ) AS
        ( SELECT DepartmentID, SUM( Salary ) AS amount
          FROM Employees
          GROUP BY DepartmentID
        )
    SELECT DepartmentID, amount, CURRENT TIMESTAMP
    FROM DepartmentPayroll
    WHERE amount = ( SELECT MAX( amount ) FROM DepartmentPayroll );
```

関連情報

複数の相関名 [421 ページ]
複数のテーブル式 [422 ページ]

1.3.6.4 共通テーブル式の一般的な使用例

共通テーブル式は、単一のクエリ内でテーブル式を複数回使用しなければならない場合に役立ちます。

よくある次のような状況では、共通テーブル式の使用が適切です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 複数の集合関数を含むクエリ
- プログラム変数への参照を含める必要がある、プロシージャ内のビュー
- 値のセットを格納するためにテンポラリビューを使用するクエリ

このリストはすべてを網羅したものではありません。共通テーブル式が役に立つ状況は、ほかにも数多くあります。

このセクションの内容:

複数の集合関数 [424 ページ]
　共通テーブル式は、単一のクエリ内で複数レベルの集約を使用しなければならない場合に役立ちます。

プログラム変数を参照するビュー [425 ページ]
　プログラム変数への参照を含むビューを作成することができます。

値を格納するビュー [426 ページ]
　SELECT 文内、または後で文に使用するプロシージャ内に、一連の値を格納することができます。

1.3.6.4.1　複数の集合関数

共通テーブル式は、単一のクエリ内で複数レベルの集約を使用しなければならない場合に役立ちます。

前項で使用した例がこれに当てはまります。そのタスクは、従業員数が最も多い部署の部署 ID を取り出すことでした。そのために、COUNT 集合関数を使用して各部署の従業員数を計算し、MAX 関数を使用して最も従業員数の多い部署を選択しています。

支払い給与総額が最も多い部を判別するクエリを記述する場合も、似たような状況となります。SUM 集合関数を使用して各部署の支払い給与総額を計算し、MAX 関数を使用してどの部署が一番多いかを判別します。クエリに両方の関数があることが、共通テーブル式が役立つかどうかを判断する手がかりとなります。

```
WITH DepartmentPayroll( DepartmentID, amount ) AS
    ( SELECT DepartmentID, SUM( Salary ) AS amount
      FROM Employees GROUP BY DepartmentID )
SELECT DepartmentID, amount
FROM DepartmentPayroll
WHERE amount = ( SELECT MAX( amount )
              FROM DepartmentPayroll )
```

関連情報

Window 集合関数 [452 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.6.4.2 プログラム変数を参照するビュー

プログラム変数への参照を含むビューを作成することができます。

たとえば、特定の顧客を識別する変数をプロシージャ内に定義するとします。その顧客の購入履歴を問い合わせる場合、同様の情報に何度もアクセスしたり、複数の集合関数を使用する必要が出じるのであれば、その特定の顧客に関する情報を含むビューを作成すると便利です。

ビューのスコープをプロシージャのスコープに制限する方法はないため、プログラム変数を参照するビューを作成することはできません。ビューは、一度作成すると、このような目的以外にも使用できます。しかし、プロシージャのクエリ内で共通テーブル式を使用することもできます。共通テーブル式のスコープはその文に制限されるため、変数を参照してもあいまい性はなく、変数の参照は許可されます。

次の文は、サンプルデータベース内のさまざまな販売担当者の総売上高を選択します。

```
SELECT GivenName || ' ' || Surname AS sales_rep_name,
       SalesRepresentative AS sales_rep_id,
       SUM( p.UnitPrice * i.Quantity ) AS total_sales
FROM Employees LEFT OUTER JOIN SalesOrders AS o
              INNER JOIN SalesOrderItems AS I
              INNER JOIN Products AS p
WHERE OrderDate BETWEEN '2000-01-01' AND '2001-12-31'
GROUP BY SalesRepresentative, GivenName, Surname;
```

前述のクエリは、次のプロシージャ内で使用されている共通テーブル式の基礎となります。調査する販売担当者の ID 番号と年度が入力パラメータです。次のプロシージャからわかるように、WITH 句内では、プロシージャパラメータと宣言済みのすべてのローカル変数を参照できます。

```
CREATE PROCEDURE sales_rep_total (
  IN rep  INTEGER,
  IN yyyy INTEGER )
BEGIN
  DECLARE StartDate DATE;
  DECLARE EndDate   DATE;
  SET StartDate = YMD( yyyy,  1,  1 );
  SET EndDate = YMD( yyyy, 12, 31 );
  WITH total_sales_by_rep ( sales_rep_name,
                            sales_rep_id,
                            month,
                            order_year,
                            total_sales ) AS
  ( SELECT GivenName || ' ' || Surname AS sales_rep_name,
           SalesRepresentative AS sales_rep_id,
           month( OrderDate),
           year( OrderDate ),
           SUM( p.UnitPrice * i.Quantity ) AS total_sales
    FROM Employees LEFT OUTER JOIN SalesOrders o
                   INNER JOIN SalesOrderItems I
                   INNER JOIN Products p
    WHERE OrderDate BETWEEN StartDate AND EndDate
           AND SalesRepresentative = rep
    GROUP BY year( OrderDate ), month( OrderDate ),
             GivenName, Surname, SalesRepresentative )
  SELECT sales_rep_name,
         monthname( YMD(yyyy, month, 1) ) AS month_name,
         order_year,
         total_sales
  FROM total_sales_by_rep
  WHERE total_sales =
    ( SELECT MAX( total_sales) FROM total_sales_by_rep )
  ORDER BY order_year ASC, month ASC;
END;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の文を実行すると、前述のプロシージャが呼び出されます。

```
CALL sales_rep_total( 129, 2000 );
```

1.3.6.4.3 値を格納するビュー

SELECT 文内、または後で文に使用するプロシージャ内に、一連の値を格納することができます。

たとえば、ある会社で販売担当者の成績を分析する際の単位として、四半期ではなく、1 年を 3 期に分けた期間を採用しているとします。四半期という日付の単位は組み込まれていますが、1 年を 3 期に分けた日付の単位はないため、プロシージャ内で日付を格納する必要があります。

```
WITH thirds ( q_name, q_start, q_end ) AS
( SELECT 'T1', '2000-01-01', '2000-04-30' UNION
  SELECT 'T2', '2000-05-01', '2000-08-31' UNION
  SELECT 'T3', '2000-09-01', '2000-12-31' )
SELECT q_name,
       SalesRepresentative,
       count(*) AS num_orders,
       SUM( p.UnitPrice * i.Quantity ) AS total_sales
FROM thirds LEFT OUTER JOIN SalesOrders AS o
    ON OrderDate BETWEEN q_start and q_end
                   KEY JOIN SalesOrderItems AS I
                   KEY JOIN Products AS p
 GROUP BY q_name, SalesRepresentative
 ORDER BY q_name, SalesRepresentative;
```

この方法は、値を定期的に保守する必要があるため、注意して使用する必要があります。たとえば、前述の文は、他の年に対して使用する場合は修正する必要があります。

この方法をプロシージャ内で適用することもできます。次の例は、対象の年を引数として受け取るプロシージャを宣言しています。

```
CREATE PROCEDURE sales_by_third ( IN y INTEGER )
BEGIN
  WITH thirds ( q_name, q_start, q_end ) AS
  ( SELECT 'T1', YMD( y, 01, 01), YMD( y, 04, 30 ) UNION
    SELECT 'T2', YMD( y, 05, 01), YMD( y, 08, 31 ) UNION
    SELECT 'T3', YMD( y, 09, 01), YMD( y, 12, 31 ) )
  SELECT q_name,
         SalesRepresentative,
         count(*) AS num_orders,
         SUM( p.UnitPrice * i.Quantity ) AS total_sales
  FROM thirds LEFT OUTER JOIN SalesOrders AS o
    ON OrderDate BETWEEN q_start and q_end
            KEY JOIN SalesOrderItems AS I
            KEY JOIN Products AS p
  GROUP BY q_name, SalesRepresentative
  ORDER BY q_name, SalesRepresentative;
END;
```

次の文を実行すると、前述のプロシージャが呼び出されます。

```
CALL sales_by_third (2000);
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.6.5 再帰共通テーブル式

再帰を使用すると、ツリーまたはツリーに似たデータ構造を表すテーブルをトラバースすることが簡単になります。

繰り返し実行される場合、共通テーブル式は再帰的であり、各実行によって、完全な結果セットが取得されるまで追加のローが返されます。

WITH 句で WITH の直後に RECURSIVE キーワードを挿入することによって、共通テーブル式を再帰的にすることができます。1 つの WITH 句には、再帰的と非再帰的両方の複数の再帰式を含めることができます。

再帰式を使用しないで単一の文内でそのような構造をトラバースする唯一の方法は、考えられるレベルごとに 1 回ずつテーブルをそれ自体にジョインすることです。

再帰共通テーブル式に関する制限

- 他の再帰共通テーブル式への参照を、再帰共通テーブル式の定義内に含めることはできません。再帰共通テーブル式を相互に再帰的にすることはできないためです。ただし、非再帰共通テーブル式には、再帰テーブル式への参照を含めることができます。また、再帰共通テーブル式には、非再帰共通テーブル式への参照を含めることができます。
- 初期サブクエリと再帰サブクエリ間で唯一サポートされている set 演算子は、UNION ALL です。
- 再帰サブクエリ定義内では、再帰共通テーブル式への自己参照は再帰サブクエリの FROM 句内のみに含めることができ、外部ジョインの NULL 入力側に含めることはできません。
- 再帰サブクエリに DISTINCT 句、GROUP BY 句、または ORDER BY 句を含めることはできません。
- 再帰サブクエリでは集合関数は使用できません。
- 再帰サブクエリの暴走を防ぐために、再帰レベル数が max_recursive_iterations オプションの現在の設定を超えるとエラーが生成されます。このオプションのデフォルト値は 100 です。

例

社内の報告関係を表すテーブルがある場合、特定の 1 人に報告を行うすべての従業員を返すクエリを記述できます。

クエリの記述方法によって、再帰レベル数を制限できます。たとえば、レベル数を制限して、トップレベルの管理者のみを返すことができますが、指揮系統が予期したよりも長い場合、除外される従業員も出てきます。レベル数を制限しない場合は、従業員が除外されることはありません。ただし、ある従業員が直接または間接的に自分自身に対してレポートするなど、実行に循環が必要な場合は、無限の再帰が発生する可能性があります。この状況は、会社の従業員が取締役会の一員である場合などに、社内の管理階層内で発生することがあります。

次のクエリは、管理レベルで従業員をリストする方法を示しています。level 0 は、管理者を持たない従業員を表します。level 1 は、level 0 の管理職の 1 人に直接報告を行う従業員を表し、level 2 は、level 1 の管理職に直接報告を行う従業員を表します。以下、同様に続きます。

```
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
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
      ON   e.ManagerID =  m.EmployeeID
       AND e.ManagerID <> e.EmployeeID
       AND m.mgmt_level < 20 ) )
SELECT * FROM manager
ORDER BY mgmt_level, Surname, GivenName;
```

管理レベルを 20 未満に制限する再帰クエリ内の条件 (`m.mgmt leve < 20`) は停止条件と呼ばれ、重要な予防措置です。これにより、テーブルデータに循環が含まれている場合に、無限再帰を回避できます。

max_recursive_iterations オプションは、暴走する再帰クエリをキャッチするために使用することもできます。このオプションのデフォルト値は 100 です。この反復数を超えた再帰クエリは終了しますが、エラーが発生します。このオプションがあれば停止条件は重要でないように見えるかもしれませんが、通常はそうではありません。各反復の間に選択されるローの数は急激に増える可能性があり、最大に達する前にパフォーマンスに深刻な影響を与えることがあります。再帰クエリ内に停止条件を設けるのは、各状況に適した制限を設定する手段です。

再帰共通テーブル式には、初期サブクエリ (シード) と、各反復の間に結果セットにローを追加する再帰サブクエリが含まれます。この 2 つの部分は、UNION ALL 演算子を使用してのみ接続できます。初期サブクエリは、通常の非再帰クエリで、最初に処理されます。再帰部分には、直前の繰り返しで追加されたローへの参照が含まれます。再帰は、繰り返しによって新しいローが生成されなくなったときに自動的に停止します。直前の繰り返しより前に選択されたローを参照する方法はありません。

再帰サブクエリの SELECT リストは、初期サブクエリの SELECT リストと数およびデータ型が一致する必要があります。データ型の自動変換が行えない場合は、一方のサブクエリの結果を明示的にキャストして、もう一方のサブクエリのデータ型に一致させます。

このセクションの内容:

部品展開の問題 [428 ページ]
　部品展開の問題は、再帰を適用できる典型例です。

再帰共通テーブル式でのデータ型宣言 [430 ページ]
　テンポラリビュー内のカラムのデータ型は、初期サブクエリのデータ型によって定義されます。

最短距離の問題 [431 ページ]
　再帰共通テーブル式を使用して、命令グラフ上の望ましいパスを見つけられます。

複数の再帰共通テーブル式 [434 ページ]
　再帰クエリには、複数の再帰クエリを含めることができますが、それらの再帰クエリが共通の要素を持たず互いに素であることが条件となります。

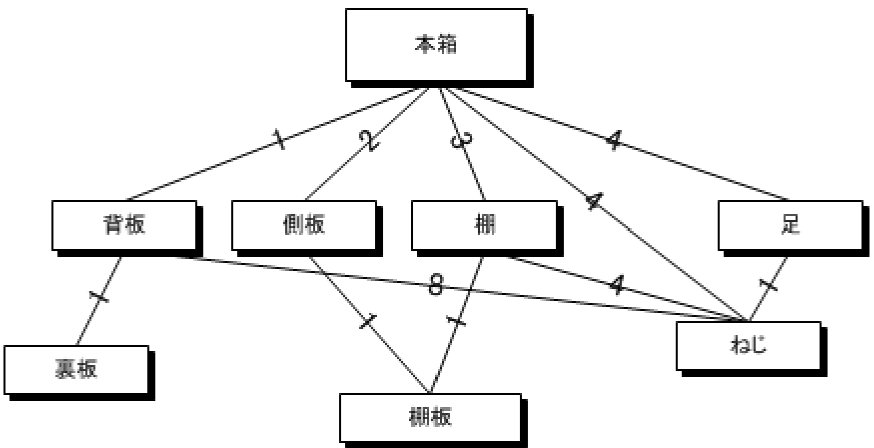
1.3.6.5.1 部品展開の問題

部品展開の問題は、再帰を適用できる典型例です。

この問題では、特定のオブジェクトを組み立てるために必要なコンポーネントが図で表されます。目的は、この図をデータベーステーブルを使用して表し、次に必要な要素部品の総数を計算することです。

たとえば、次の図は単純な本箱のコンポーネントを表しています。この本箱は、3 段の棚、背、そして 4 本のねじで固定される 4 本の足から構成されています。各棚は、4 つのねじで固定される板です。背には、8 つのねじで固定される別の板が使われています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.



次のテーブル内の情報は、本箱の図のエッジを表しています。最初のカラムはコンポーネントを、2 番目のカラムはそのコンポーネントのサブコンポーネントの 1 つを、そして 3 番目のカラムは必要なサブコンポーネント数を示しています。

コンポーネント	サブコンポーネント	数量
本箱	背板	1
本箱	側板	2
本箱	棚	3
本箱	足	4
本箱	ねじ	4
背板	裏板	1
背板	ねじ	8
側板	棚板	1
棚	棚板	1
棚	ねじ	4

次の文を実行して、bookcase テーブルを作成し、コンポーネントデータとサブコンポーネントデータを挿入します。

```
CREATE TABLE bookcase (
      component      VARCHAR(9),
      subcomponent   VARCHAR(9),
      quantity       INTEGER,
    PRIMARY KEY ( component, subcomponent )
);
INSERT INTO bookcase
  SELECT 'bookcase', 'back',      1 UNION
  SELECT 'bookcase', 'side',      2 UNION
  SELECT 'bookcase', 'shelf',     3 UNION
  SELECT 'bookcase', 'foot',      4 UNION
  SELECT 'bookcase', 'screw',     4 UNION
  SELECT 'back',     'backboard', 1 UNION
  SELECT 'back',     'screw',     8 UNION
  SELECT 'side',     'plank',     1 UNION
  SELECT 'shelf',    'plank',     1 UNION
  SELECT 'shelf',    'screw',     4;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の文を実行して、本箱の組み立てに必要なコンポーネント、サブコンポーネント、およびその数量のリストを生成します。

```
SELECT * FROM bookcase
ORDER BY component, subcomponent;
```

次の文を実行して、本箱の組み立てに必要なサブコンポーネントとその数量のリストを生成します。

```
WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT component, subcomponent, quantity
  FROM bookcase WHERE component = 'bookcase'
    UNION ALL
  SELECT b.component, b.subcomponent, p.quantity * b.quantity
  FROM parts p JOIN bookcase b ON p.subcomponent = b.component )
SELECT subcomponent, SUM( quantity ) AS quantity
FROM parts
WHERE subcomponent NOT IN ( SELECT component FROM bookcase )
GROUP BY subcomponent
ORDER BY subcomponent;
```

このクエリの結果を次に示します。

subcomponent	quantity
裏板	1
足	4
棚板	5
ねじ	24

また、このクエリを別の再帰レベルを実行するように書き直すこともできます。こうすると、メインの SELECT 文内でサブクエリを記述する必要がなくなります。次のクエリの結果は、前述のクエリの結果とまったく同じです。

```
WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT component, subcomponent, quantity
  FROM bookcase WHERE component = 'bookcase'
    UNION ALL
  SELECT p.subcomponent, b.subcomponent,
    IF b.quantity IS NULL
    THEN p.quantity
    ELSE p.quantity * b.quantity
    ENDIF
  FROM parts p LEFT OUTER JOIN bookcase b
  ON p.subcomponent = b.component
    WHERE p.subcomponent IS NOT NULL
 )
SELECT component, SUM( quantity ) AS quantity
FROM parts
WHERE subcomponent IS NULL
GROUP BY component
ORDER BY component;
```

1.3.6.5.2 再帰共通テーブル式でのデータ型宣言

テンポラリビュー内のカラムのデータ型は、初期サブクエリのデータ型によって定義されます。

再帰サブクエリのカラムのデータ型は、一致する必要があります。データベースサーバは、再帰サブクエリによって返された値がその初期クエリの値に一致するよう、自動的に変換しようとします。これが不可能な場合、または変換で情報が失われた場合、エラーが生成されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

通常、初期サブクエリがリテラル値または NULL を返す場合、明示的なキャストが必要な場合がほとんどです。初期サブクエリが再帰サブクエリとは異なるカラムから値を選択する場合も、明示的なキャストが必要な場合があります。

キャストは、初期サブクエリのカラムが再帰サブクエリのカラムと同じドメインを持っていない場合に必要な場合があります。初期サブクエリでは、NULL 値に常にキャストを適用する必要があります。

たとえば、構成部品の問題は、初期サブクエリが bookcase テーブルからローを返すことによって、選択されたカラムのデータ型を継承するため、正しく動作します。

ただし、このクエリが次のように書き直された場合は、明示的なキャストが必要です。

```
WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT NULL, 'bookcase', 1
  UNION ALL
  SELECT b.component, b.subcomponent, p.quantity * b.quantity
  FROM parts p JOIN bookcase b
    ON p.subcomponent = b.component
)
SELECT * FROM parts
ORDER BY component, subcomponent;
```

キャストを行わない場合、次の理由でエラーが発生します。

- コンポーネント名の正しいデータ型は VARCHAR ですが、最初のカラムは NULL です。
- 数字 1 は SMALLINT と見なされますが、quantity カラムのデータ型は INT です。

2 番目のカラムには、キャストは不要です。初期クエリのこのカラムがすでに文字列であるためです。

初期サブクエリでデータ型をキャストすれば、クエリを意図したとおりに動作させることができます。

```
WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT CAST( NULL AS VARCHAR ), 'bookcase', CAST( 1 AS INT )
  UNION ALL
  SELECT b.component, b.subcomponent,
         p.quantity * b.quantity
  FROM parts p JOIN bookcase b
    ON p.subcomponent = b.component
)
SELECT * FROM parts
ORDER BY component, subcomponent;
```

関連情報

部品展開の問題 [428 ページ]

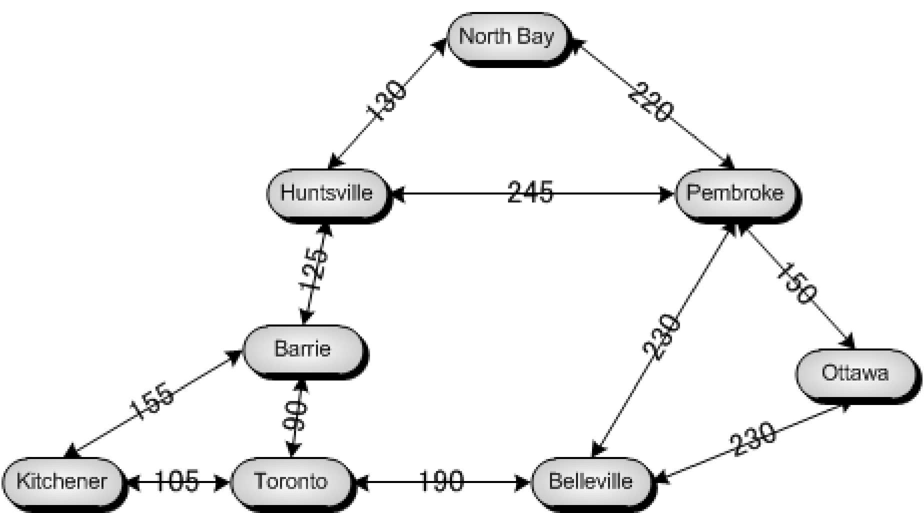
1.3.6.5.3 最短距離の問題

再帰共通テーブル式を使用して、命令グラフ上の望ましいパスを見つけられます。

データベーステーブルの各ローは、命令エッジを表します。各ローは、出発地から目的地まで移動するときの出発地、目的地、およびコストを示します。問題の種類によって、コストは距離であったり、所要時間であったり、または別の基準であることも考えられます。再帰を使用すると、このグラフを介して可能なルートを探索できます。そして、いくつかの可能なルートから目的のルートを選ぶことができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば、キッチナー市とペンブローク市の間を車で移動するときの望ましいルートを見つける問題を考えてみます。いくつも可能なルートがあり、それぞれ異なるいくつかの都市を中継点として通ります。目的は、最短ルートをいくつか見つけ、それらを別の適当な選択肢と比較することです。



まず、このグラフのエッジを表すテーブルを定義し、各エッジに対して 1 ローを挿入します。このグラフのすべてのエッジは双方向であるため、逆方向を表すエッジも挿入します。このために、最初のローセットを選択し、出発地と目的地を入れ替えます。たとえば、一方のローがキッチナー市からトロント市への移動を表し、もう一方のローがトロント市からキッチナー市へ戻る移動を表します。

```
CREATE TABLE travel (
    origin      VARCHAR(10),
    destination VARCHAR(10),
    distance    INT,
  PRIMARY KEY ( origin, destination )
);
INSERT INTO travel
  SELECT 'Kitchener',  'Toronto',    105 UNION
  SELECT 'Kitchener',  'Barrie',     155 UNION
  SELECT 'North Bay',  'Pembroke',   220 UNION
  SELECT 'Pembroke',   'Ottawa',     150 UNION
  SELECT 'Barrie',     'Toronto',     90 UNION
  SELECT 'Toronto',    'Belleville', 190 UNION
  SELECT 'Belleville', 'Ottawa',     230 UNION
  SELECT 'Belleville', 'Pembroke',   230 UNION
  SELECT 'Barrie',     'Huntsville', 125 UNION
  SELECT 'Huntsville', 'North Bay',  130 UNION
  SELECT 'Huntsville', 'Pembroke',   245;
INSERT INTO travel   -- Insert the return trips
  SELECT destination, origin, distance
  FROM travel;
```

次に、再帰共通テーブル式を記述します。移動はキッチナー市から始まるため、初期サブクエリは、キッチナー市を起点とする可能なすべてのパスを、その距離とともに選択することから始まります。

再帰サブクエリは、パスを拡張します。各パスに対し、再帰サブクエリは、直前のセグメントの目的地から続くセグメントを追加し、新しいセグメントの距離を加えて、各ルートの現在のトータルコストを管理します。効率性を考慮して、次の条件のいずれかに該当した場合、ルートは終了します。

- パスが出発地に戻ります。
- パスが直前の地点に戻ります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

- パスが最後の目的地に達します。

この例では、キッチナー市に戻るパスはなく、全てのパスはペンブローク市に達した場合、終了します。

再帰クエリを使用して環状のグラフを探索する場合は、再帰クエリが適切に終了することを確認することが重要です。この例の場合、前述の条件では不十分です。ルートに 2 つの中継地の間を行ったり来たりする移動が無制限に含まれる可能性があるからです。次の再帰クエリでは、指定したルート内の最大セグメント数を 7 つまでに制限することで、ルートの終わりを保証しています。

このクエリ例のポイントは現実的なルートを選択することであるため、メインクエリでは、距離が最短ルート比 50% 増未満のルートのみを選択しています。

```
WITH RECURSIVE
    trip ( route, destination, previous, distance, segments ) AS
( SELECT CAST( origin || ', ' || destination AS VARCHAR(256) ),
         destination, origin, distance, 1
  FROM travel
  WHERE origin = 'Kitchener'
    UNION ALL
  SELECT route || ', ' || v.destination,
         v.destination,            -- current endpoint
         v.origin,                 -- previous endpoint
         t.distance + v.distance,  -- total distance
         segments + 1              -- total number of segments
  FROM trip t JOIN travel v ON t.destination = v.origin
  WHERE v.destination <> 'Kitchener'  -- Don't return to start
    AND v.destination <> t.previous   -- Prevent backtracking
    AND v.origin      <> 'Pembroke'   -- Stop at the end
    AND segments                      -- TERMINATE RECURSION!
          < ( SELECT count(*)/2 FROM travel ) )
SELECT route, distance, segments FROM trip
WHERE destination = 'Pembroke' AND
      distance < 1.5 * ( SELECT MIN( distance )
                         FROM trip
                         WHERE destination = 'Pembroke' )
ORDER BY distance, segments, route;
```

この文を前述のデータセットに対して実行すると、次のような結果となります。

route	distance	segments
Kitchener, Barrie, Huntsville, Pembroke	525	3
Kitchener, Toronto, Belleville, Pembroke	525	3
Kitchener, Toronto, Barrie, Huntsville, Pembroke	565	4
Kitchener, Barrie, Huntsville, North Bay, Pembroke	630	4
Kitchener, Barrie, Toronto, Belleville, Pembroke	665	4
Kitchener, Toronto, Barrie, Huntsville, North Bay, Pembroke	670	5
Kitchener, Toronto, Belleville, Ottawa, Pembroke	675	4

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.6.5.4 複数の再帰共通テーブル式

再帰クエリには、複数の再帰クエリを含めることができますが、それらの再帰クエリが共通の要素を持たず互いに素であることが条件となります。

また、再帰クエリには再帰共通テーブル式と非再帰共通テーブル式を混在させることができます。共通テーブル式が 1 つでも再帰的である場合は、RECURSIVE キーワードを含める必要があります。

たとえば、前述のクエリと同じ結果を返す次のクエリは、別の非再帰共通テーブル式を使用して最短ルートの距離を選択しています。2 つ目の共通テーブル式の定義は、1 つ目の定義からカンマで区切られています。

```
WITH RECURSIVE
  trip ( route, destination, previous, distance, segments ) AS
    ( SELECT CAST( origin || ', ' || destination AS VARCHAR(256) ),
             destination, origin, distance, 1
      FROM travel
      WHERE origin = 'Kitchener'
      UNION ALL
      SELECT route || ', ' || v.destination,
             v.destination,
             v.origin,
             t.distance + v.distance,
             segments + 1
      FROM trip t JOIN travel v ON t.destination = v.origin
      WHERE v.destination <> 'Kitchener'
        AND v.destination <> t.previous
        AND v.origin      <> 'Pembroke'
        AND segments
              < ( SELECT count(*)/2 FROM travel ) ),
  shortest ( distance ) AS                 -- Additional,
    ( SELECT MIN(distance)                 -- non-recursive
      FROM trip                            -- common table
      WHERE destination = 'Pembroke' )     -- expression
SELECT route, distance, segments FROM trip
WHERE destination = 'Pembroke' AND
      distance < 1.5 * ( SELECT distance FROM shortest )
ORDER BY distance, segments, route;
```

非再帰共通テーブル式と同様、再帰式をストアドプロシージャ内で使用した場合、ローカル変数やプロシージャパラメータへの参照を含められます。たとえば、次に定義する best_routes プロシージャは、指定した 2 つの都市の間の最短ルートを特定します。

```
CREATE PROCEDURE best_routes (
   IN initial VARCHAR(10),
   IN final   VARCHAR(10)
)
BEGIN
  WITH RECURSIVE
      trip ( route, destination, previous, distance, segments ) AS
  ( SELECT CAST( origin || ', ' || destination AS VARCHAR(256) ),
           destination, origin, distance, 1
    FROM travel
    WHERE origin = initial
      UNION ALL
    SELECT route || ', ' || v.destination,
           v.destination,            -- current endpoint
           v.origin,                 -- previous endpoint
           t.distance + v.distance,  -- total distance
           segments + 1              -- total number of segments
    FROM trip t JOIN travel v ON t.destination = v.origin
    WHERE v.destination <> initial     -- Don't return to start
      AND v.destination <> t.previous  -- Prevent backtracking
      AND v.origin      <> final       -- Stop at the end
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
      AND segments                        -- TERMINATE RECURSION!
            < ( SELECT count(*)/2 FROM travel ) )
   SELECT route, distance, segments FROM trip
   WHERE destination = final AND
         distance < 1.4 * ( SELECT MIN( distance )
                            FROM trip
                            WHERE destination = final )
   ORDER BY distance, segments, route;
END;
```

次の文を実行すると、前述のプロシージャが呼び出されます。

```
CALL best_routes ( 'Pembroke', 'Kitchener' );
```

1.3.7 OLAP のサポート

OLAP (オンライン分析処理) では、単一の SQL 文内で複雑なデータ分析を実行できます。また、データベースでクエリの量を減らすことでパフォーマンスを向上しながら、結果の値を増やすことができます。

SQL 文と Window 関数の拡張を使用することで、OLAP 機能を利用できます。このような SQL の拡張や機能では、多次元データ分析、データマイニング、時系列分析、傾向分析、コスト配分、ゴールシーク、例外警告などを、通常は単一の SQL 文を使用して、簡単に実行できます。

SELECT 文の拡張

SELECT 文を拡張することにより、入力ローをグループ化したり、グループを分析したり、最終結果セットに検索結果を含めることができます。これらの拡張には、GROUP BY 句 (GROUPING SETS、CUBE、ROLLUP の各サブ句) と WINDOW 句に対する拡張が含まれます。

GROUP BY 句の拡張では、入力ローを複数の方法で分割できるため、さまざまなグループを連結する結果セットを得ることができます。データマイニング用に散在した多次元結果セット (データキューブとも呼ばれる) を作成することもできます。さらに、この拡張により、サブ合計のローと総合計のローを使用して、分析をより便利にすることができます。

WINDOW 句は、入力ローのグループで分析する機会を増やすために Window 関数とともに使用されます。

Window 集合関数

ほとんどの集合関数で、入力ローの処理に合わせて入力ローを上から下に移動する、設定可能なスライドウィンドウの概念がサポートされています。ウィンドウの移動とともにウィンドウ内のデータに対する追加の計算を実行できるため、セマンティック上同等なセルフジョインクエリや相関サブクエリを使用する方法よりも効率的な方法で、詳細な分析を実行できます。

たとえば Window 集合関数を GROUP BY 句の CUBE、ROLLUP、GROUPING SETS 拡張と組み合わせることで、単一の SQL 文内における百分位数、移動平均、累積合計を効率的に計算できます。それ以外の方法では、セルフジョイン、相関サブクエリ、テンポラリテーブル、またはこれら 3 つを組み合わせなければなりません。

Window 集合関数を使用すると、ダウジョーンズ工業平均株価の四半期の移動平均や、部門ごとの全従業員と給与の累積合計などの情報を取得できます。また、平方偏差、標準偏差、相関、回帰などの測定を計算することもできます。

Window ランキング関数

Window ランキング関数を使用すると、今年出荷された総売り上げ上位 10 製品や、最低 15 企業に販売注文した販売担当者の上位 5% といった情報を取得する、単一文の SQL クエリを作ることができます。

このセクションの内容:

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

OLAP パフォーマンスの向上 [436 ページ]
OLAP パフォーマンスを向上させるには、optimization_workload データベースオプションを OLAP に設定して、調査する候補にクラスタド GROUP BY ハッシュ演算子を使用することをオプティマイザに指示します
GROUP BY 句の拡張 [437 ページ]
SELECT 文で標準の GROUP BY 句を使用すると、指定したグループ化の式に従って、結果セット内のローをグループ化できます。
GROUPING SETS のショートカットとしての ROLLUP と CUBE [440 ページ]
複数の異なるデータ分割を単一の結果セットに連結する場合は、GROUPING SETS を使用します。
Window 関数 [445 ページ]
入力ローのセットに対して分析演算を実行できる関数は、Window 関数と呼ばれます。OLAP の機能には、入力ローの処理に合わせて入力ローを上から下に移動するスライドウィンドウの概念が含まれます。
Window 集合関数 [452 ページ]
Window 集合関数は、入力内のローの指定されたセットに対する値を返します。
基本集合関数 [453 ページ]
ローのグループの値を返すために使用できる基本集合関数がいくつかサポートされています。
標準偏差関数と平方偏差関数 [461 ページ]
平方偏差関数と標準偏差関数について、標本バージョンと母集団バージョンという 2 つのバージョンがサポートされています。
相関関数と線形回帰関数 [464 ページ]
さまざまな統計関数がサポートされています。関数の結果は、線形回帰の質を分析するのに役立ちます。
Window ランキング関数 [465 ページ]
Window ランキング関数は、分割内の他のローに関連するローのランクを返します。
ローナンバリング関数 [472 ページ]
ローナンバリング関数は、分割内のローにユニークな番号を付けます。

関連情報

Window ランキング関数 [465 ページ]
Window 集合関数 [452 ページ]

1.3.7.1 OLAP パフォーマンスの向上

OLAP パフォーマンスを向上させるには、optimization_workload データベースオプションを OLAP に設定して、調査する候補にクラスタド GROUP BY ハッシュ演算子を使用することをオプティマイザに指示します

インデックスの定義時に FOR OLAP WORKLOAD オプションを使用して、OLAP 負荷のインデックスを調整することもできます。このオプションを使用すると、データベースサーバは特定の最適化を実行します。この最適化には、同じキー内の 2 つのローの最大ページ距離に関して、クラスタド GROUP BY ハッシュ演算子で使用する統計情報の管理が含まれます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.7.2 GROUP BY 句の拡張

SELECT 文で標準の GROUP BY 句を使用すると、指定したグループ化の式に従って、結果セット内のローをグループ化できます。

たとえば、`GROUP BY columnA, columnB` を指定すると、columnA と columnB のユニークな値の組み合わせでローがグループ化されます。標準の GROUP BY 句では、グループは、指定されたすべての GROUP BY 式の組み合わせの評価を反映します。

ただし、結果セットに別のグループ化またはサブグループ化を指定したい場合も考えられます。たとえば、結果で columnA と columnB のユニークな値でグループ化されたデータを表示してから、columnC の別の値でもう一度グループ化するような状況です。このような結果を得るには、GROUP BY 句で GROUPING SETS 拡張を使用します。

このセクションの内容:

GROUP BY GROUPING SETS [437 ページ]
GROUPING SETS 句を使用すると、複数の SELECT 文を使用しなくても結果を複数の方法でグループ化できます。

1.3.7.2.1 GROUP BY GROUPING SETS

GROUPING SETS 句を使用すると、複数の SELECT 文を使用しなくても結果を複数の方法でグループ化できます。

GROUPING SETS 句は、SELECT 文の GROUP BY 句の拡張です。

たとえば、次の 2 つのクエリ文はセマンティック上同義です。ただし、2 番目のクエリでは GROUP BY GROUPING SETS 句を使用することで、グループ化基準をより効率的に定義します。

複数の SELECT 文を使用した複数のグループ化

```
SELECT NULL, NULL, NULL, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
   UNION ALL
SELECT City, State, NULL, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY City, State
   UNION ALL
SELECT NULL, NULL, CompanyName, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY CompanyName;
```

GROUPING SETS を使用した複数のグループ化

```
SELECT City, State, CompanyName, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY GROUPING SETS( ( City, State ), ( CompanyName ) , ( ) );
```

どちらの方法でも、次に示す同じ結果が生成されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	City	State	CompanyName	Cnt
1	(NULL)	(NULL)	(NULL)	8
2	(NULL)	(NULL)	Cooper Inc.	1
3	(NULL)	(NULL)	Westend Dealers	1
4	(NULL)	(NULL)	Toto's Active Wear	1
5	(NULL)	(NULL)	North Land Trading	1
6	(NULL)	(NULL)	The Ultimate	1
7	(NULL)	(NULL)	Molly's	1
8	(NULL)	(NULL)	Overland Army Navy	1
9	(NULL)	(NULL)	Out of Town Sports	1
10	'Pembroke'	'MB'	(NULL)	4
11	'Petersburg'	'KS'	(NULL)	1
12	'Drayton'	'KS'	(NULL)	3

ロー 2 ～ 9 は、CompanyName ごとにグループ化して生成されたローで、ロー 10 ～ 12 は、City と State の組み合わせでグループ化されたローです。ロー 1 は、一致した括弧 () のペアを使用して指定される空のグループ化セットで表される総合計です。空のグループ化セットは、GROUP BY に対する入力のすべてのローの分割 1 つを表します。

NULL 値は、グループ化セットで使用されない任意の式のプレースホルダとして使用されていることに注意してください。これは、結果セットが結合可能である必要があるためです。たとえば、ロー 2 ～ 9 は、クエリの 2 番目のグループ化セット (CompanyName) から得られます。グループ化セットには式として City または State が含まれないため、ロー 2 ～ 9 では City と State の値にプレースホルダの NULL が含まれますが、CompanyName の値には CompanyName に出現する重複しない値が含まれます。

NULL はプレースホルダとして使用されるため、プレースホルダの NULL とデータ内の実際の NULL を混乱しがちです。プレースホルダの NULL を NULL データと区別するには、GROUPING 関数を使用してください。

例

次の例では、クエリから返される結果を調整するために GROUPING SETS を使用する方法と、結果をわかりやすく整理するために ORDER BY 句を使用する方法を示します。クエリは、各年 (Year) の四半期 (Quarter) ごとの注文の合計数と、年 (Year) ごとの合計数を返します。年 (Year)、四半期 (Quarter) の順で並べることで、結果を理解しやすくなります。

```
SELECT Year( OrderDate ) AS Year,
        Quarter( OrderDate ) AS Quarter,
        COUNT (*) AS Orders
FROM SalesOrders
GROUP BY GROUPING SETS ( ( Year, Quarter ), ( Year ) )
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

	Year	Quarter	Orders
1	2000	(NULL)	380
2	2000	1	87
3	2000	2	77

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

	Year	Quarter	Orders
4	2000	3	91
5	2000	4	125
6	2001	(NULL)	268
7	2001	1	139
8	2001	2	119
9	2001	3	10

ロー 1 と 6 は、それぞれ 2000 年と 2001 年の注文の小計です。ロー 2 ～ 5 とロー 7 ～ 9 は、小計ローのディテールローに当たります。つまり、これらのローは、四半期ごとと年ごとの注文の合計数を示します。

この結果セットには、すべての年のすべての四半期の総合計がありません。総合計が含まれるようにするには、クエリで GROUPING SETS 指定に空のグループ化指定 '()' を含める必要があります。

空のグループ化指定の使用

GROUP BY 句で空の GROUPING SETS 指定 '()' を使用すると、結果で合計されるすべての項目について総合計のローが生成されます。総合計の行では、すべてのグループ化の式に対するすべての値にプレースホルダの NULL が含まれます。GROUPING 関数を使用すると、ローの基本となるデータで値を評価するため、プレースホルダの NULL と実際の NULL を区別できます。

重複したグループ化セットの指定

GROUPING SETS 句では、重複したグループ化指定を使用できます。この場合、SELECT 文の結果に同一のローが含まれます。

次のクエリには、重複したグループ化が含まれます。

```
SELECT City, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY GROUPING SETS( ( City ), ( City ) );
```

このクエリは、次の結果を返します。重複したグループ化の結果として、ロー 1 ～ 3 がロー 4 ～ 6 と同一になります。

	City	Cnt
1	'Drayton'	3
2	'Petersburg'	1
3	'Pembroke'	4
4	'Drayton'	3
5	'Petersburg'	1

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	City	Cnt
6	'Pembroke'	4

適切な形式の実践

GROUP BY GROUPING SETS 句でのグループ化構文の解釈は、単純な GROUP BY 句の場合とは異なります。たとえば、GROUP BY (X, Y) は X と Y の値の異なる組み合わせによってグループ化されます。しかし GROUP BY GROUPING SETS (X, Y) の場合は、2 つの個別のグループ化セットを指定し、その 2 つのグループ化の結果がユニオンされます。つまり、結果は (X) でグループ化されてから、(Y) でグループ化された同じ結果にユニオンされます。

適切な形式を使用し、複雑な式の場合のあいまいさを避けるために、エラーが発生する可能性がある場合は指定内の各グループ化セットを括弧で囲んでください。たとえば、次の両方の文は正しく、セマンティック上同一ですが、2 番目が推奨される形式を反映した文です。

```
SELECT * FROM t GROUP BY GROUPING SETS ( X, Y );
SELECT * FROM t GROUP BY GROUPING SETS( ( X ), ( Y ) );
```

関連情報

GROUPING 関数を使用した NULL の検出 [444 ページ]
GROUPING 関数を使用した NULL の検出 [444 ページ]

1.3.7.3 GROUPING SETS のショートカットとしての ROLLUP と CUBE

複数の異なるデータ分割を単一の結果セットに連結する場合は、GROUPING SETS を使用します。

多くのグループ化を指定する必要があり、かつ小計を含める場合は、ROLLUP 拡張と CUBE 拡張を使用します。

ROLLUP 句と CUBE 句は、事前に定義された GROUPING SETS 指定のショートカットと見なすことができます。

ROLLUP は、空のグループ化セット「()」から始まり、追加する式を前の式に連結させるグループ化セットが次々と続くような、一連のグループ化を指定するのと同等です。たとえば、3 つのグループ化式 a、b、c があり、ROLLUP を指定した場合は、セット ()、(a)、(a, b)、(a, b, c) を使用して GROUPING SETS 句を指定した場合と同じ結果になります。この構成は、階層グループ化と呼ばれることもあります。

CUBE を使用すると、さらに多くのグループ化を実現できます。CUBE を指定することは、すべての可能な GROUPING SETS を指定するのと同等です。たとえば、同様の 3 つのグループ化式 a、b、c があり、CUBE を指定した場合は、セット ()、(a)、(a, b)、(a, c)、(b)、(b, c)、(c)、(a, b, c) を使用して GROUPING SETS 句を指定した場合と同じ結果になります。

ROLLUP または CUBE を指定する場合は、GROUPING 関数を使用して、結果内にあるプレースホルダの NULL を識別してください。プレースホルダの NULL は、ROLLUP または CUBE による結果セット内で暗黙的である小計のローが原因で発生します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

ROLLUP 句 [441 ページ]
ROLLUP 句を使用して、グループ化属性の階層を指定できます。

CUBE 句 [442 ページ]
データキューブとは、GROUP BY 式の可能な組み合わせを使用した入力の n 次元要約で、CUBE 句が使用されます。

GROUPING 関数を使用した NULL の検出 [444 ページ]
ROLLUP と CUBE で作成される合計や小計のローには、グループ化で使用されなかった SELECT リストで指定したあらゆるカラムに、プレースホルダの NULL が含まれます。

1.3.7.3.1 ROLLUP 句

ROLLUP 句を使用して、グループ化属性の階層を指定できます。

多くのアプリケーションに共通の要件は、グループ化属性の小計を左から右へ順番に計算することです。このパターンは、階層として参照されます。小計の計算が追加されることにより、詳細度を上げた追加のローが生成されるためです。

ROLLUP 句を使用したクエリでは、次のようなグループ化セットの階層が生成されます。ROLLUP 句に (X_1,X_2, ... , X_n) という形式で n 個の GROUP BY 式が含まれている場合、ROLLUP 句によって次のように n +1 個のグループ化セットが生成されます。

```
{ (), (X1), (X1,X2), (X1,X2,X3), ... , (X1,X2,X3, ... , Xn) }
```

例

次のクエリは、年ごとと四半期ごとに販売注文を要約し、次の表に示す結果セットを返します。

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY ROLLUP( Year, Quarter )
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

	Quarter	Year	Orders	GQ	GY
1	(NULL)	(NULL)	648	1	1
2	(NULL)	2000	380	1	0
3	1	2000	87	0	0
4	2	2000	77	0	0
5	3	2000	91	0	0
6	4	2000	125	0	0

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

	Quarter	Year	Orders	GQ	GY
7	(NULL)	2001	268	1	0
8	1	2001	139	0	0
9	2	2001	119	0	0
10	3	2001	10	0	0

結果セットの 1 番目のローは、2 年間のすべての四半期について、すべての注文の総合計 (648) を示します。

ロー 2 は 2000 年の注文の合計数 (380) を示し、ロー 3 〜 6 はこの年の四半期ごとの注文の小計を示します。同様に、ロー 7 は 2001 年の注文の合計数 (268) を示し、ロー 8 〜 10 はこの年の四半期ごとの小計を示します。

GROUPING 関数で返される値を使用して、総合計が含まれるローと小計のローを区別できます。ロー 2 と 7 では、四半期カラムは NULL で、GQ (Grouping by Quarter) カラムは値 1 です。これは、このローが年ごとのすべての四半期の注文の合計であることを示します。

同様に、ロー 1 の四半期 (Quarter) カラムと年 (Year) カラムは NULL で、GQ カラムと GY カラムは値 1 です。これは、このローがすべての年のすべての四半期における注文の合計であることを示します。

Transact-SQL WITH ROLLUP 構文のサポート

代わりに Transact-SQL 互換の構文である WITH ROLLUP を使用して、GROUP BY ROLLUP と同じ結果を得ることもできます。ただし、構文が多少異なり、構文に指定できるのは単純な GROUP BY 式リストだけです。

次のクエリは、前述の GROUP BY ROLLUP の例の場合と同等の結果を生成します。

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY Year, Quarter WITH ROLLUP
ORDER BY Year, Quarter;
```

1.3.7.3.2 CUBE 句

データキューブとは、GROUP BY 式の可能な組み合わせを使用した入力の n 次元要約で、CUBE 句が使用されます。

CUBE 句を使用してデータキューブを作成することができます。

CUBE 句の結果は、各値セットの要素のすべての可能な組み合わせを含む積集合になります。複雑なデータ分析で非常に役立ちます。

CUBE 句に (X_1,X_2, ...,X_n) という形式で n 個の GROUPING 式が存在する場合、CUBE によって次のように 2^n 個のグループ化セットが生成されます。

```
{(), (X1), (X1,X2), (X1,X2,X3), ... , (X1,X2,X3, ...,Xn),
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
(X2), (X2,X3), (X2,X3,X4), ... , (X2,X3,X4, ... , Xn), ... , (Xn)}.
```

例

次のクエリは、年別、四半期別、および各年の四半期ごとの販売注文を要約し、次の表に示す結果セットを生成します。

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY CUBE ( Year, Quarter )
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

	Quarter	Year	Orders	GQ	GY
1	(NULL)	(NULL)	648	1	1
2	1	(NULL)	226	0	1
3	2	(NULL)	196	0	1
4	3	(NULL)	101	0	1
5	4	(NULL)	125	0	1
6	(NULL)	2000	380	1	0
7	1	2000	87	0	0
8	2	2000	77	0	0
9	3	2000	91	0	0
10	4	2000	125	0	0
11	(NULL)	2001	268	1	0
12	1	2001	139	0	0
13	2	2001	119	0	0
14	3	2000	10	0	0

結果セットの 1 番目のローは、2000 年と 2001 年を結合した、すべての四半期についての、すべての注文の総合計 (648) を示します。

ロー 2 ～ 5 は、すべての年の暦四半期ごとの販売注文を要約します。

ロー 6 と 11 の Orders は、それぞれ 2000 年と 2001 年の注文の合計を示します。

ロー 7 ～ 10 と 12 ～ 14 は、それぞれ 2000 年と 2001 年の四半期ごとの合計を示します。

GROUPING 関数で返される値を使用して、総合計が含まれるローと小計のローを区別できます。ロー 6 と 11 は、四半期 (Quarter) カラムは NULL で、GQ (Grouping by Quarter) カラムは値 1 です。これは、このローが年ごとにすべての四半期の注文 (Orders) の合計であることを示します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

注記

CUBE は指数関数的な個数のグループ化セットを生成するため、CUBE を使用して生成される結果セットは非常に大きくなることがあります。このため、64 個を超える GROUP BY 式が含まれる GROUP BY 句はサポートされていません。この上限を超える文は、SQLCODE -944 (SQLSTATE 42WA1) で失敗します。

Transact-SQL WITH CUBE 構文のサポート

代わりに Transact-SQL 互換の構文である WITH CUBE を使用して、GROUP BY CUBE と同じ結果を得ることもできます。ただし、構文が多少異なり、構文に指定できるのは単純な GROUP BY 式リストだけです。

次のクエリは、前述の GROUP BY CUBE の例の場合と同等の結果を生成します。

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY Year, Quarter WITH CUBE
ORDER BY Year, Quarter;
```

1.3.7.3.3 GROUPING 関数を使用した NULL の検出

ROLLUP と CUBE で作成される合計や小計のローには、グループ化で使用されなかった SELECT リストで指定したあらゆるカラムに、プレースホルダの NULL が含まれます。

結果を検査しているときは、小計のローにある NULL がプレースホルダの NULL なのか、それともローの基本になるデータの評価による NULL なのかを区別できません。その結果、ディテールロー、小計ロー、総合計ローを区別することも困難になります。

GROUPING 機能を使用すると、プレースホルダの NULL を基本となるデータによる NULL と区別できます。グループ化セット指定から `group-by-expression` を 1 つ使用して GROUPING 関数を指定した場合、この関数は、プレースホルダの NULL の場合は 1 を返し、そのローの基本となるデータに存在する値 (通常は NULL) を反映している場合は 0 を返します。

たとえば、次のクエリは下の表に示される結果セットを返します。

```
SELECT Employees.EmployeeID AS Employee,
   YEAR( OrderDate ) AS Year,
   COUNT( SalesOrders.ID ) AS Orders,
   GROUPING( Employee ) AS GE,
   GROUPING( Year ) AS GY
FROM Employees LEFT OUTER JOIN SalesOrders
   ON Employees.EmployeeID = SalesOrders.SalesRepresentative
WHERE Employees.Sex IN ( 'F' )
   AND Employees.State IN ( 'TX' , 'NY' )
GROUP BY GROUPING SETS ( ( Year, Employee ), ( Year ), ( ) )
ORDER BY Year, Employee;
```

このクエリは、次の結果を返します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	Employees	Year	Orders	GE	GY
1	(NULL)	(NULL)	54	1	1
2	(NULL)	(NULL)	0	1	0
3	102	(NULL)	0	0	0
4	390	(NULL)	0	0	0
5	1062	(NULL)	0	0	0
6	1090	(NULL)	0	0	0
7	1507	(NULL)	0	0	0
8	(NULL)	2000	34	1	0
9	667	2000	34	0	0
10	(NULL)	2001	20	1	0
11	667	2001	20	0	0

この例では、空のグループ化セット「()」が指定されたため、ロー 1 は注文の総合計 (54) を表します。GE と GY の両方に 1 が含まれており、Employees カラムと Year カラムの NULL がそれぞれ Employees と Year のプレースホルダ NULL であることを示しています。

ロー 2 は小計のローです。GE カラムの 1 は、Employees カラムの NULL がプレースホルダ NULL であることを示しています。GY カラムの 0 は、Year カラムの NULL が基本となるデータの評価による結果であり、プレースホルダ NULL ではないことを示します。この場合、このローは注文のない従業員を表します。

ロー 3 ～ 7 は、従業員ごとの、Year が NULL である注文の合計数を示しています。つまり、これらは注文のない Texas と New York に住む女性従業員のものです。これらのローはロー 2 のディテールローです。つまり、ロー 2 はロー 3 ～ 7 の合計です。

ロー 8 は、2000 年のすべての従業員の分を合わせた注文数を示す小計ローです。ロー 9 は、ロー 8 の単一ディテールローです。

ロー 10 は、2001 年のすべての従業員の分を合わせた注文数を示す小計ローです。ロー 11 は、ロー 10 の単一ディテールローです。

1.3.7.4 Window 関数

入力ローのセットに対して分析演算を実行できる関数は、Window 関数と呼ばれます。OLAP の機能には、入力ローの処理に合わせて入力ローを上から下に移動するスライドウィンドウの概念が含まれます。

たとえば、すべてのランキング関数、およびほとんどの集合関数は **Window** 関数です。Window 関数を使用すると、データの追加分析を実行できます。この操作を行うには、入力ローを処理前に分割してソートし、次にサイズを設定可能な入力が進むウィンドウ内でローを処理します。

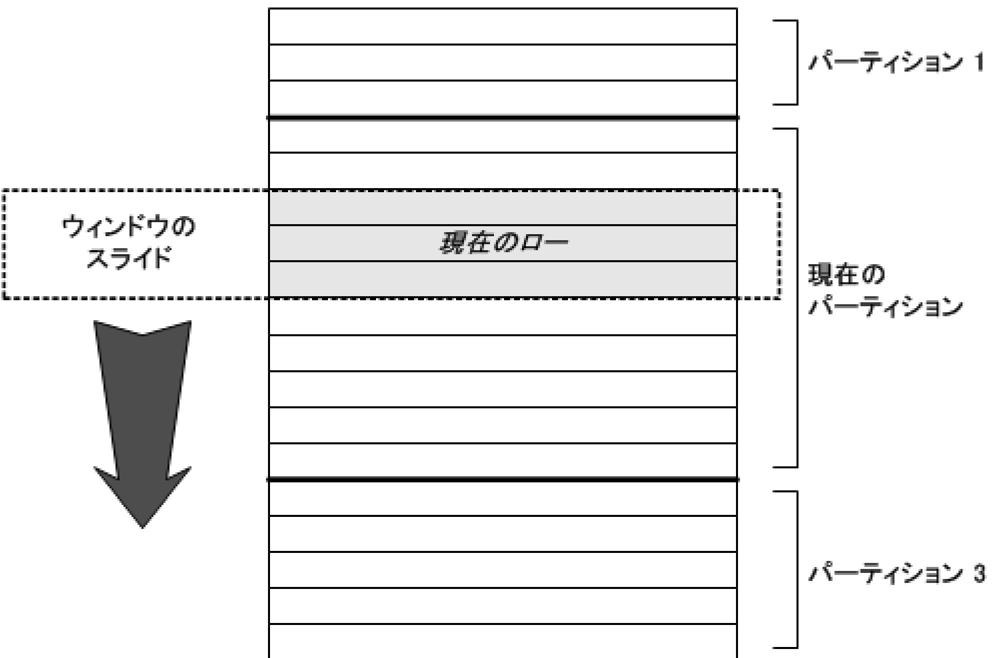
ウィンドウの移動とともにウィンドウ内のデータに対する追加の計算を実行できるため、セマンティック上同等なセルフジョインクエリや相関サブクエリを使用する方法よりも効率的な方法で、詳細な分析を実行できます。

ウィンドウの境界は、データから抽出しようとする情報を基に設定します。ウィンドウには、ウィンドウ定義内のグループ化指定に従って分割された入力データの、1 つ、複数、またはすべてのローが含まれます。ウィンドウは入力データを上から下に移動し、必須の計算を実行するために必要なローを採用します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

Window 関数には、Window 集合関数、Window ランキング関数、ローナンバリング関数の 3 種類があります。

次の図は、入力ローが処理されるのに伴って移動するウィンドウを示しています。データ分割は、ウィンドウ定義に指定された、入力ローのグループ化を反映します。グループ化が指定されていない場合は、すべての入力ローで 1 分割であると見なされます。ウィンドウの長さ (つまりウィンドウに含まれるローの数) と、現在のローと比較したウィンドウのオフセットは、ウィンドウ定義で指定した境界を反映します。



このセクションの内容:

ウィンドウ定義 [446 ページ]
SQL のウィンドウ拡張を使用して、ウィンドウの境界や、入力ローの分割や順序付けを設定できます。

ウィンドウ定義: OVER 句と WINDOW 句を使用したインライン定義 [449 ページ]
OLAP ウィンドウは、OVER 句と WINDOW 句を使用して定義されています。

1.3.7.4.1 ウィンドウ定義

SQL のウィンドウ拡張を使用して、ウィンドウの境界や、入力ローの分割や順序付けを設定できます。

論理的には、GROUP BY 句で定義されたグループが作成された後、最終の SELECT リストの評価とクエリの ORDER BY 句の前に、クエリ仕様の結果を計算するセマンティックの一部として分割が作成されます。SQL 文における句の評価順は次のようになります。

1. FROM
2. WHERE
3. GROUP BY
4. HAVING

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

5. WINDOW
6. DISTINCT
7. ORDER BY

クエリを形成する際には、評価順の影響を考慮する必要があります。たとえば、同じ SELECT クエリブロックにある Window 関数を参照する式を述部にすることはできません。ただし、クエリブロックを派生テーブルに入れることで、派生テーブルを使用して述部を指定できます。次のクエリを実行すると、Window 関数が述部に使用されているという旨のメッセージが表示されて、クエリは失敗します。

```
SELECT DepartmentID, Surname, StartDate, Salary,
       SUM( Salary ) OVER ( PARTITION BY DepartmentID
          ORDER BY StartDate
          RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS "Sum_Salary"
   FROM Employees
   WHERE State IN ( 'CA', 'UT', 'NY', 'AZ' )
      AND DepartmentID IN ( '100', '200' )
   GROUP BY DepartmentID, Surname, StartDate, Salary
   HAVING Salary > 0 AND "Sum_Salary" > 200
   ORDER BY DepartmentID, StartDate;
```

目的の結果を得るためには、派生テーブル (DT) を使用して述部を指定します。

```
SELECT * FROM ( SELECT DepartmentID, Surname, StartDate, Salary,
                       SUM( Salary ) OVER ( PARTITION BY DepartmentID
                         ORDER BY StartDate
                         RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS
"Sum_Salary"
                  FROM Employees
                  WHERE State IN ( 'CA', 'UT', 'NY', 'AZ' )
                     AND DepartmentID IN ( '100', '200' )
                  GROUP BY DepartmentID, Surname, StartDate, Salary
                  HAVING Salary > 0
                  ORDER BY DepartmentID, StartDate ) AS DT
   WHERE DT.Sum_Salary > 200;
```

ウィンドウ分割は GROUP BY 演算子に続くので、分割を実行する計算で、任意の集合関数 (SUM、AVG、VARIANCE など) の結果を利用できます。そのため、クエリの GROUP BY 句や ORDER BY 句だけでなく、ウィンドウを使用することでもグループ化と順序付けの操作を実行できます。

ウィンドウ指定の定義

Window 関数で操作するウィンドウを定義するときは、次の項目を 1 つまたは複数指定します。

分割 (PARTITION BY 句)

PARTITION BY 句により、入力ローのグループ化方法を定義します。省略すると、入力全体が単一の分割として扱われます。指定した内容に応じて、1 つ、複数、またはすべての入力ローから分割を作成できます。2 つの分割からのデータが混合することはありません。つまり、ウィンドウが 2 つの分割の境界に達すると、分割内のデータの処理が終了してから、次の分割内のデータの処理が開始されます。ウィンドウの境界の定義方法に応じて、ウィンドウのサイズが分割の先頭と末尾で変化する可能性があります。

順序付け (ORDER BY 句)

ORDER BY 句により、Window 関数による処理の前に、入力ローの順序を決める方法を定義します。RANGE 句を使用して境界を指定する場合、またはランキング関数がウィンドウを参照する場合にかぎり、ORDER BY 句が必要です。それ

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

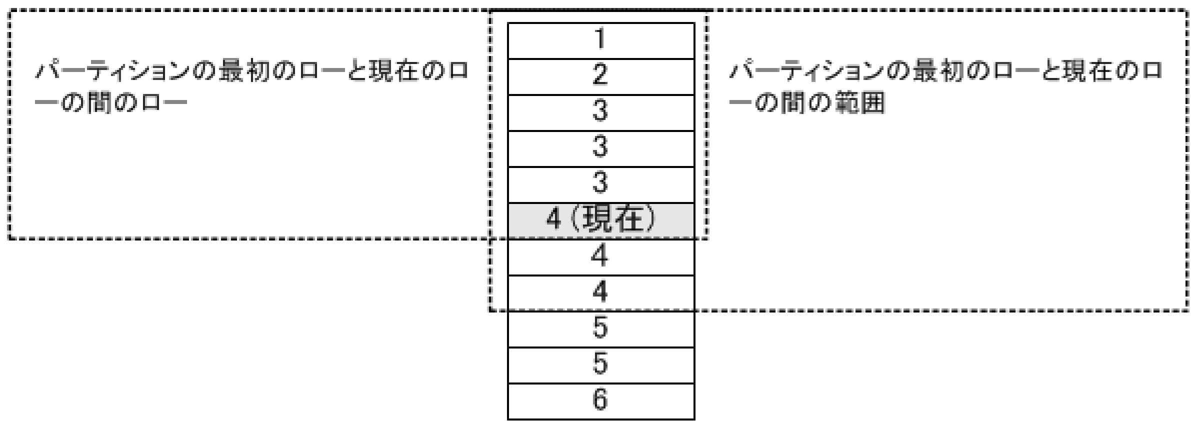
以外の場合、ORDER BY 句はオプションです。省略すると、データベースサーバが最も効率的な順序で入力ローを処理します。

境界 **(RANGE** 句および **ROWS** 句**)**

現在のローは、ウィンドウの開始ローと終了ローを判断するための参照ポイントになります。ウィンドウ定義の RANGE 句と ROWS 句を使用して、境界を設定できます。RANGE 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ことでウィンドウを定義します。範囲を計算するためにはデータが順序付けられている必要があるため、RANGE を指定する場合は ORDER BY 句も指定する必要があります。

ROWS は、現在のローからのオフセットとしてロー数を指定することで、ウィンドウを定義します。

RANGE は、データ値の範囲としてローのセットを定義するため、RANGE ウィンドウに含まれるローは現在のローを越える場合があります。この点は ROWS とは異なります。次の図は、ROWS 句と RANGE 句の違いを示したものです。



ROWS 句と RANGE 句内では、現在のローを基準にしてウィンドウの開始ローと終了ローを (オプションで) 指定できます。これには、PRECEDING 句、BETWEEN 句、および FOLLOWING 句を使用します。これらの句には、式の他に、UNBOUNDED と CURRENT ROW の各キーワードも指定できます。ウィンドウに境界が定義されていない場合、ウィンドウ境界はデフォルトで次のようになります。

- ウィンドウ指定に ORDER BY 句が含まれている場合は、RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW を指定したことと同義になります。
- ウィンドウ指定に ORDER BY 句が含まれていない場合は、ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING を指定したことと同義になります。

次の表は、ウィンドウ境界の例とウィンドウに含まれるローの説明を示したものです。

指定	意味
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW	分割の先頭から開始し、現在のローで終了します。累積の結果 (累積合計など) を計算する場合に使用してください。
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING	分割内のすべてのローを使用します。分割の各ローに対して集合関数の値を同一にする場合に使用してください。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

指定	意味
ROWS BETWEEN x PRECEDING AND y FOLLOWING	現在のローから x の距離にある開始ローと y の距離にある終了ロー (境界値を含む) からなる固定サイズの移動するウィンドウを作成します。移動平均を計算する場合や隣接するロー間の値の差分を計算する場合には、この例を使用してください。 複数のローからなる移動するウィンドウでは、分割の最初のローや最後のローを計算するときに NULL になります。これは、現在のローが分割のまさに最初または最後のローである場合に、計算で使用できる直前または直後のローが存在しないためです。そのため、代わりに NULL 値が使用されます。
ROWS BETWEEN CURRENT ROW AND CURRENT ROW	1 つのロー (現在のロー) からなるウィンドウ。
RANGE BETWEEN 5 PRECEDING AND 5 FOLLOWING	ロー内の値に基づいてウィンドウを作成します。たとえば、現在のローに対して、ORDER BY 句に指定されたカラムに値 10 が含まれているとします。ウィンドウのサイズを RANGE BETWEEN 5 PRECEDING AND 5 FOLLOWING と指定した場合、最初のローにはそのカラムに 5、最後のローにはそのカラムに 15 がそれぞれ含まれるのに必要な大きさとなるよう、ウィンドウのサイズを指定します。ウィンドウが分割を移動すると、範囲仕様を満たすのに必要なサイズに応じて、ウィンドウのサイズが変化します。

ウィンドウ指定はできるかぎり明示的にしてください。明示的に指定しないと、デフォルトが予期した結果を返さない場合があります。

連続でない値の場合は、RANGE 句を使用して、Window 関数の入力の差に起因する問題を回避します。RANGE 句を使用してウィンドウ境界が設定された場合、データベースサーバは隣接するローや重複する値を含むローを自動的に処理します。

RANGE では、符号なし整数値を使用します。ORDER BY 式のドメインと RANGE 句で指定した値のドメインに応じて、範囲式のトランケーションが発生することがあります。

ランキング関数やロー番号付け関数を使用するときは、ウィンドウの境界を指定しないでください。

1.3.7.4.2 ウィンドウ定義: OVER 句と WINDOW 句を使用したインライン定義

OLAP ウィンドウは、OVER 句と WINDOW 句を使用して定義されています。

ウィンドウを定義する方法には次の 3 つがあります。

- インライン (Window 関数の OVER 句内)
- WINDOW 句内
- インラインと WINDOW 句内で部分的に指定

ただし、以降の項に示すとおり、方法によっては制限があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

インライン定義 (Window 関数の OVER 句内)

ウィンドウ定義は、Window 関数の OVER 句に配置できます。このことを、ウィンドウをインラインで定義するといいます。

たとえば、次の文は、2001 年 7 月と 8 月に出荷されたすべての製品と、出荷日ごとの累積出荷数量について、サンプルデータベースに問い合わせます。ウィンドウはインラインで定義されています。

```
SELECT p.ID, p.Description, s.Quantity, s.ShipDate,
   SUM( s.Quantity ) OVER ( PARTITION BY s.ProductID
       ORDER BY s.ShipDate
       ROWS BETWEEN UNBOUNDED PRECEDING
       AND CURRENT ROW ) AS Cumulative_qty
FROM SalesOrderItems s JOIN Products p
   ON ( s.ProductID = p.ID )
WHERE s.ShipDate BETWEEN '2001-07-01' AND '2001-08-31'
ORDER BY p.ID;
```

このクエリは、次の結果を返します。

	ID	Description	Quantity	ShipDate	Cumulative_qty
1	301	V-neck	24	2001-07-16	24
2	302	Crew Neck	60	2001-07-02	60
3	302	Crew Neck	36	2001-07-13	96
4	400	Cotton Cap	48	2001-07-05	48
5	400	Cotton Cap	24	2001-07-19	72
6	401	Wool Cap	48	2001-07-09	48
7	500	Cloth Visor	12	2001-07-22	12
8	501	Plastic Visor	60	2001-07-07	60
9	501	Plastic Visor	12	2001-07-12	72
10	501	Plastic Visor	12	2001-07-22	84
11	601	Zipped Sweatshirt	60	2001-07-19	60
12	700	Cotton Shorts	24	2001-07-26	24

この例では、2 つのテーブルのジョインとクエリの WHERE 句の適用後に、SUM Window 関数の計算が発生します。クエリは次のように処理されます。

1. ProductID の値を基に、入力ローを分割 (グループ化) します。
2. 各分割内で、ShipDate の値を基にローをソートします。
3. 分割内の各ローについて、Quantity の値に対して SUM 関数を評価します。このとき、各分割の最初の (ソートされた) ローからなるスライドウィンドウを使用します。

WINDOW 句の定義

前述のクエリの別の構成として、WINDOW 句を使用して、ウィンドウを使用する関数とは別にウィンドウを指定します。次に、各関数の OVER 句からウィンドウを参照します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

この例で、WINDOW 句はウィンドウ Cumulative を作成し、データを ProductID ごとに分割し、ShipDate で並べ替えます。SUM 関数は、その OVER 句でウィンドウを参照し、ROWS 句を使用してウィンドウのサイズを定義します。

```
SELECT p.ID, p.Description, s.Quantity, s.ShipDate,
    SUM( s.Quantity ) OVER ( Cumulative
    ROWS BETWEEN UNBOUNDED PRECEDING
    AND CURRENT ROW ) AS cumulative_qty
FROM SalesOrderItems s
JOIN Products p ON ( s.ProductID = p.ID )
WHERE s.ShipDate BETWEEN '2001-07-01' AND '2001-08-31'
WINDOW Cumulative AS ( PARTITION BY s.ProductID ORDER BY s.ShipDate )
ORDER BY p.ID;
```

WINDOW 句構文を使用するときは、次の制限があります。

- PARTITION BY 句を指定する場合は、WINDOWS 句内に配置する必要があります。
- ROWS 句または RANGE 句を指定する場合は、参照元関数の OVER 句内に配置する必要があります。
- ウィンドウに ORDER BY 句を指定する場合は、WINDOW 句内か参照元関数の OVER 句内に配置できますが、両方には配置できません。
- WINDOW 句は、SELECT 文の ORDER BY 句に先行する必要があります。

インラインおよび **WINDOW** 句の定義の組み合わせ

ウィンドウ定義の一部をインラインで指定し、残りを WINDOW 句で定義できます。次に例を示します。

```
AVG() OVER ( windowA
             ORDER BY expression )...
...
WINDOW windowA AS ( PARTITION BY expression )
```

この方法でウィンドウ定義を分割すると、次の制限が適用されます。

- Window 関数構文では PARTITION BY 句を使用できません。
- Window 関数構文または WINDOW 句のいずれかで ORDER BY 句を使用することはできますが、両方では使用できません。
- RANGE 句または ROWS 句を WINDOW 句に含めることはできません。

関連情報

Window 集合関数 [452 ページ]
Window ランキング関数 [465 ページ]
ウィンドウ定義 [446 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.7.5 Window 集合関数

Window 集合関数は、入力内のローの指定されたセットに対する値を返します。

たとえば、Window 関数を使用して、指定した期間における会社の販売数の移動平均を計算できます。

Window 集合関数は次の 3 つのカテゴリに分類されます。

基本集合関数

サポートされる基本集合関数のリストを次に示します。

- SUM 関数 [集合]
- AVG 関数 [集合]
- MAX 関数 [集合]
- MIN 関数 [集合]
- MEDIAN 関数 [集合]
- FIRST_VALUE 関数 [集合]
- LAST_VALUE 関数 [集合]
- COUNT 関数 [集合]

標準偏差関数と平方偏差関数

サポートされる標準偏差関数と平方偏差関数のリストを次に示します。

- STDDEV 関数 [集合]
- STDDEV_POP 関数 [集合]
- STDDEV_SAMP 関数 [集合]
- VAR_POP 関数 [集合]
- VAR_SAMP 関数 [集合]
- VARIANCE 関数 [集合]

相関関数と線形回帰関数

サポートされる相関関数と線形回帰関数のリストを次に示します。

- COVAR_POP 関数 [集合]
- COVAR_SAMP 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]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

基本集合関数 [453 ページ]
相関関数と線形回帰関数 [464 ページ]
標準偏差関数と平方偏差関数 [461 ページ]

1.3.7.6 基本集合関数

ローのグループの値を返すために使用できる基本集合関数がいくつかサポートされています。

複雑なデータ分析では、複数レベルの集約が必要になることがあります。GROUP BY 句に加え、またはその代わりに、ウィンドウ分割や並べ替えを使用すると、複雑な SQL クエリを非常に柔軟に構成できます。たとえば、ウィンドウ構成と単純な集合関数を組み合わせると、移動平均、移動合計、移動最小、移動最大、累積合計などの値を計算できます。

サポートしている基本集合関数を次に示します。

SUM 関数

ローグループごとに、指定された式の合計を返します。

AVG 関数

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返します。

MAX 関数

各ローグループで見つかった式の最大値を返します。

MIN 関数

各ローグループで見つかった式の最小値を返します。

MEDIAN 関数

ローのセットの数値式の中央値を返します。

FIRST_VALUE 関数

ウィンドウの最初のローの値を返します。この関数では、ウィンドウを指定する必要があります。

LAST_VALUE 関数

ウィンドウの最後のローの値を返します。この関数では、ウィンドウを指定する必要があります。

COUNT 関数

指定された式の条件を満たすローの数を返します。

このセクションの内容:

SUM 関数の例 [454 ページ]
SUM 関数を使用して、一連のローの値の合計を返すことができます。

AVG 関数の例 [457 ページ]
AVG 関数を使用して、一連のローの値の移動平均を計算することができます。

MAX 関数の例 [458 ページ]
MAX 関数を使用して、一連のローの値の最大値を返すことができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

FIRST_VALUE 関数と LAST_VALUE 関数の例 [458 ページ]
FIRST_VALUE 関数と LAST_VALUE 関数は、ウィンドウの最初と最後のローの値を返します。

関連情報

Window 関数 [445 ページ]

1.3.7.6.1 SUM 関数の例

SUM 関数を使用して、一連のローの値の合計を返すことができます。

次のクエリは、DepartmentID 別にデータを分割した結果セットを返し、在籍経験が長い人から順に従業員の給与の累積合計 (Sum_Salary) を算出します。結果セットには、カリフォルニア州、ユタ州、ニューヨーク州、またはアリゾナ州に住む従業員のみが含まれます。Sum_Salary のカラムは、従業員の給与の累積合計です。

```
SELECT DepartmentID, Surname, StartDate, Salary,
SUM( Salary ) OVER ( PARTITION BY DepartmentID
    ORDER BY StartDate
    RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW )
AS "Sum_Salary"
FROM Employees
WHERE State IN ( 'CA', 'UT', 'NY', 'AZ' )
   AND DepartmentID IN ( '100', '200' )
ORDER BY DepartmentID, StartDate;
```

次のテーブルは、クエリからの結果セットを示します。結果セットは、DepartmentID ごとに分割されています。

	DepartmentID	Surname	StartDate	Salary	Sum_Salary
1	100	Whitney	1984-08-28	45700.00	45700.00
2	100	Cobb	1985-01-01	62000.00	107700.00
3	100	Shishov	1986-06-07	72995.00	180695.00
4	100	Driscoll	1986-07-01	48023.69	228718.69
5	100	Guevara	1986-10-14	42998.00	271716.69
6	100	Wang	1988-09-29	68400.00	340116.69
7	100	Soo	1990-07-31	39075.00	379191.69
8	100	Diaz	1990-08-19	54900.00	434091.69
9	200	Overbey	1987-02-19	39300.00	39300.00
10	200	Martel	1989-10-16	55700.00	95000.00
11	200	Savarino	1989-11-07	72300.00	167300.00
12	200	Clark	1990-07-21	45000.00	212300.00
13	200	Goggin	1990-08-05	37900.00	250200.00

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

DepartmentID 100 の場合、カリフォルニア州、ユタ州、ニューヨーク州、またはアリゾナ州に住む従業員の給与の累積合計は 434,091.69 ドルで、部門 200 の従業員の累積合計は 250,200.00 ドルです。

隣接ローのデルタの計算

2 つのウィンドウ (現在のローと直前のローのそれぞれに対するウィンドウ) を使用すると、隣接ローのデルタ (変化量) を計算できます。たとえば、次のクエリの結果では、ある従業員とその直前の従業員の給与のデルタ (Delta) を計算します。

```
SELECT EmployeeID AS EmployeeNumber,
   Surname AS LastName,
   SUM( Salary ) OVER ( ORDER BY BirthDate
      ROWS BETWEEN CURRENT ROW AND CURRENT ROW )
      AS CurrentRow,
   SUM( Salary ) OVER ( ORDER BY BirthDate
      ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING )
      AS PreviousRow,
   ( CurrentRow - PreviousRow ) AS Delta
FROM Employees
WHERE State IN ( 'NY' );
```

	EmployeeNumber	LastName	CurrentRow	PreviousRow	Delta
1	913	Martel	55700.000	(NULL)	(NULL)
2	1062	Blaikie	54900.000	55700.000	-800.000
3	249	Guevara	42998.000	54900.000	-11902.000
4	390	Davidson	57090.000	42998.000	14092.000
5	102	Whitney	45700.000	57090.000	-11390.000
6	1507	Wetherby	35745.000	45700.000	-9955.000
7	1751	Ahmed	34992.000	35745.000	-753.000
8	1157	Soo	39075.000	34992.000	4083.000

CurrentRow ウィンドウでは、ウィンドウのサイズが `ROWS BETWEEN CURRENT ROW AND CURRENT ROW` に設定されているため、SUM は現在のローのみに対して実行されます。同様に、PreviousRow ウィンドウでは、ウィンドウのサイズが `ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING` に設定されているため、SUM は直前のローのみに対して実行されます。最初のローには先行するローがないため、このローの PreviousRow の値は NULL です。そのため Delta の値も NULL になります。

複雑な分析

次のクエリを考えてみます。このクエリは、データベース内で製品ごとに最上位の営業担当者 (総売り上げで定義) をリストします。

```
SELECT s.ProductID AS Products, o.SalesRepresentative,
     SUM( s.Quantity ) AS total_quantity,
     SUM( s.Quantity * p.UnitPrice ) AS total_sales
  FROM SalesOrders o KEY JOIN SalesOrderItems s
    KEY JOIN Products p
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
  GROUP BY s.ProductID, o.SalesRepresentative
  HAVING total_sales = (
    SELECT First SUM( s2.Quantity * p2.UnitPrice )
        AS sum_sales
      FROM SalesOrders o2 KEY JOIN
        SalesOrderItems s2 KEY JOIN Products p2
      WHERE s2.ProductID = s.ProductID
      GROUP BY o2.SalesRepresentative
      ORDER BY sum_sales DESC )
  ORDER BY s.ProductID;
```

このクエリは、次の結果を返します。

	Products	SalesRepresentative	total_quantity	total_sales
1	300	299	660	5940.00
2	301	299	516	7224.00
3	302	299	336	4704.00
4	400	299	458	4122.00
5	401	902	360	3600.00
6	500	949	360	2520.00
7	501	690	360	2520.00
8	501	949	360	2520.00
9	600	299	612	14688.00
10	601	299	636	15264.00
11	700	299	1008	15120.00

元のクエリは、ProductID をサブクエリの相関外部参照として、ある製品の最高売り上げを判断する相関サブクエリを使用して作成されています。ただし、この場合のようにネストされたクエリを使用すると、コストの高いオプションになることがあります。これは、サブクエリに GROUP BY 句だけでなく、GROUP BY 句内の ORDER BY 句も含まれるからです。そのため、クエリオプティマイザは、同じセマンティックを保持したままになり、このネストされたクエリをジョインとして書き換えることができません。したがって、クエリの実行中は、外部ブロック内で計算される派生ローごとにサブクエリが評価されます。

コストの高い Filter 述部に注意してください。オプティマイザは、クエリの実行コストの 99% がこのプラン演算子に起因するものであると推定します。サブクエリのプランでは、メインブロックのフィルタ演算子のコストが高くなる理由を明確にしています。サブクエリには、ネストループジョインが 2 つと、ハッシュ GROUP BY 演算が 1 つ、ソートが 1 つ含まれます。

ランキング関数を使用した書き換え

ランキング関数を使用した同じクエリの書き換えでは、同じ結果が非常に効率よく計算されます。

```
SELECT v.ProductID, v.SalesRepresentative,
  v.total_quantity, v.total_sales
  FROM ( SELECT o.SalesRepresentative, s.ProductID,
           SUM( s.Quantity ) AS total_quantity,
           SUM( s.Quantity * p.UnitPrice ) AS total_sales,
           RANK() OVER ( PARTITION BY s.ProductID
             ORDER BY SUM( s.Quantity * p.UnitPrice ) DESC )
             AS sales_ranking
          FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
          GROUP BY o.SalesRepresentative, s.ProductID )
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
          AS v
  WHERE sales_ranking = 1
  ORDER BY v.ProductID;
```

GROUP BY 句が処理されてから SELECT リスト項目の評価とクエリの ORDER BY 句の処理が行われるまでに、ウィンドウ演算子が計算されます。3 つのテーブルのジョイン後、ジョインされたローは SalesRepresentative 属性と ProductID 属性の組み合わせでグループ化されます。したがって、total_quantity と total_sales の SUM 集合関数は、SalesRepresentative と ProductID の組み合わせごとに計算できます。

GROUP BY 句の評価に続いて、中間結果のローを total_sales の降順にランク付けするために、ウィンドウを使用して RANK 関数が計算されます。WINDOW 指定には PARTITION BY 句が含まれます。それによって、GROUP BY 句の結果が、今度は ProductID ごとに再分割 (再グループ化) されます。そのため、RANK 関数は、総売り上げの降順で製品ごとにローをランク付けしますが、その製品を販売した営業担当者はすべてまとめられます。このランキングにより、最上位の営業担当者を特定するのに必要なのは、ランクが 1 ではない抽出テーブルのローを拒否するように抽出テーブルの結果を制限するだけです。同順の場合 (結果セットのロー 7 と 8)、RANK は同じ値を返します。したがって、690 と 949 の両方の営業担当者が最終結果に出現します。

1.3.7.6.2 AVG 関数の例

AVG 関数を使用して、一連のローの値の移動平均を計算することができます。

次の例では、2000 年における月別の全製品販売の移動平均を計算するための Window 関数として AVG が使用されています。

WINDOW 指定で RANGE 句を使用します。ROWS 句の場合のように隣接ローの数で計算されるのではなく、RANGE 句により月の値を基にウィンドウ境界が計算されます。ある月に一部またはすべての製品の販売がなかった場合などは、ROWS を使用すると異なる結果が生成されます。

```
SELECT *
  FROM ( SELECT s.ProductID,
         Month( o.OrderDate ) AS julian_month,
         SUM( s.Quantity ) AS sales,
         AVG( SUM( s.Quantity ) )
         OVER ( PARTITION BY s.ProductID
           ORDER BY Month( o.OrderDate ) ASC
           RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING )
         AS average_sales
         FROM SalesOrderItems s KEY JOIN SalesOrders o
         WHERE Year( o.OrderDate ) = 2000
         GROUP BY s.ProductID, Month( o.OrderDate ) )
  AS DT
  ORDER BY 1,2;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.7.6.3 MAX 関数の例

MAX 関数を使用して、一連のローの値の最大値を返すことができます。

相関サブクエリの削除

場合によっては、特定のカラムの値を最大値や最小値と比較する必要があります。

このようなクエリは、相関属性 (外部参照とも呼ばれる) のあるネストされたクエリとして作成されることがよくあります。たとえば、次のクエリでは、その製品の注文 1 回あたりの最大数が製品の在庫数を超えているような製品について、すべての注文を製品情報とともにリストします。

```
SELECT o.ID, o.OrderDate, p.*
FROM SalesOrders o, SalesOrderItems s, Products p
WHERE o.ID = s.ID AND s.ProductID = p.ID
       AND p.Quantity < ( SELECT MAX( s2.Quantity )
                          FROM SalesOrderItems s2
                          WHERE s2.ProductID = p.ID )
ORDER BY p.ID, o.ID;
```

このクエリのグラフィカルなプランは、プランビューアに表示されます。このネストされたクエリが、クエリオプティマイザによってどのように変形されたかを確認してください。クエリは、Products テーブルと、Window 関数を含む派生テーブル (相関名 DT) のある SalesOrders テーブルとのジョインに変形されています。

オプティマイザに依存して相関サブクエリを派生テーブルによるジョインに変形するのではなく (セマンティック分析は複雑なので、この方法は単純な場合にしか使用できない)、Window 関数を使用して、このようなクエリを作成できます。

```
SELECT order_quantity.ID, o.OrderDate, p.*
  FROM ( SELECT s.ID, s.ProductID,
           MAX( s.Quantity ) OVER (
             PARTITION BY s.ProductID
             ORDER BY s.ProductID )
           AS max_quantity
           FROM SalesOrderItems s )
  AS order_quantity, Products p, SalesOrders o
  WHERE p.ID = ProductID
    AND o.ID = order_quantity.ID
    AND p.Quantity < max_quantity
  ORDER BY p.ID, o.ID;
```

1.3.7.6.4 FIRST_VALUE 関数と LAST_VALUE 関数の例

FIRST_VALUE 関数と LAST_VALUE 関数は、ウィンドウの最初と最後のローの値を返します。

これらの関数を使用すると、セルフジョインを使わずにクエリで複数のローの値に一度にアクセスできます。

この 2 つの関数は、ウィンドウに使用する必要があるので、他の Window 集合関数とは異なります。また、これらの関数では IGNORE NULLS 句を使用できる点も、他の Window 集合関数と異なります。IGNORE NULLS を指定すると、式の最初または最後の NULL 以外の値が返されます。指定しなければ、最初または最後の値が返されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

例 1: グループの最初のエントリ

FIRST_VALUE 関数を使用して、一定の順序で並んでいる値グループの最初のエントリを取り出すことができます。次のクエリは、各注文について、注文の最初の品目の ProductID、つまり各注文で LineID が最も小さい品目の ProductID を返します。

クエリでは、DISTINCT キーワードを使用して重複を削除しています。このキーワードを指定しなかった場合、各注文の各品目について重複するローが返されます。

```
SELECT DISTINCT ID,
FIRST_VALUE( ProductID ) OVER ( PARTITION BY ID ORDER BY LineID )
FROM SalesOrderItems
ORDER BY ID;
```

例 2: 最大売り上げに対する割合

FIRST_VALUE 関数の一般的な使用方法として、各ローの値を、現在のグループ内の最大値または最小値と比較できます。次のクエリは、各営業担当者の総売り上げを計算してから、その総売り上げと、同じ製品の最大総売り上げを比較します。結果は、最大総売り上げのパーセントで表されます。

```
SELECT s.ProductID AS prod_id, o.SalesRepresentative AS sales_rep,
    SUM( s.Quantity * p.UnitPrice ) AS total_sales,
    100 * total_sales / ( FIRST_VALUE( SUM( s.Quantity * p.UnitPrice ) )
                          OVER Sales_Window ) AS total_sales_percentage
  FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
  GROUP BY o.SalesRepresentative, s.ProductID
    WINDOW Sales_Window AS ( PARTITION BY s.ProductID
                             ORDER BY SUM( s.Quantity * p.UnitPrice ) DESC )
  ORDER BY s.ProductID;
```

例 3: NULL 値を設定することによるデータの高密度化

FIRST_VALUE 関数と LAST_VALUE 関数は、データの密度を高めた後で、NULL の代わりに値の設定が必要な場合に便利です。たとえば、1 日の総売り上げが最も高い営業担当者が表彰されるとします。次のクエリは、2001 年 4 月の第 1 週にトップ成績を上げた担当者を表示します。

```
SELECT v.OrderDate, v.SalesRepresentative AS rep_of_the_day
FROM ( SELECT o.SalesRepresentative, o.OrderDate,
              RANK() OVER ( PARTITION BY o.OrderDate
                            ORDER BY SUM( s.Quantity *
                              p.UnitPrice ) DESC ) AS sales_ranking
       FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
       GROUP BY o.SalesRepresentative, o.OrderDate ) AS v
WHERE v.sales_ranking = 1
AND v.OrderDate BETWEEN '2001-04-01' AND '2001-04-07'
ORDER BY v.OrderDate;
```

このクエリは、次の結果を返します。

OrderDate	rep_of_the_day
2001-04-01	949
2001-04-02	856
2001-04-05	902
2001-04-06	467

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

OrderDate	rep_of_the_day
2001-04-07	299

ただし、売り上げがなかった日については結果は返されません。次のクエリは、データの密度を高め、売り上げがなかった日のレコードも作成されるようにします。また、LAST_VALUE 関数を使用して、表彰がなかった日には、成績順位に入れ替わりがあるまで最後にトップ成績を獲得した者の ID を NULL 値の代わりに rep_of_the_day に表示させています。

```
SELECT d.dense_order_date,
                   LAST_VALUE( v.SalesRepresentative IGNORE NULLS )
                       OVER ( ORDER BY d.dense_order_date )
                       AS rep_of_the_day
FROM ( SELECT o.SalesRepresentative, o.OrderDate,
              RANK() OVER ( PARTITION BY o.OrderDate
                            ORDER BY SUM( s.Quantity *
                             p.UnitPrice ) DESC ) AS sales_ranking
       FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
       GROUP BY o.SalesRepresentative, o.OrderDate ) AS v
RIGHT OUTER JOIN ( SELECT DATEADD( day, row_num, '2001-04-01' )
                        AS dense_order_date
                   FROM sa_rowgenerator( 0, 6 )) AS d
ON v.OrderDate = d.dense_order_date AND sales_ranking = 1
ORDER BY d.dense_order_date;
```

このクエリは、次の結果を返します。

dense_order_date	rep_of_the_day
2001-04-01	949
2001-04-02	856
2001-04-03	856
2001-04-04	856
2001-04-05	902
2001-04-06	467
2001-04-07	299

前のクエリからの派生テーブル v を、対象日をすべて含む派生テーブル d にジョインすると、1 日ごとにローができます。ただし、この外部ジョインでは、売り上げがなかった日の SalesRepresentative カラムには NULL が含まれます。LAST_VALUE 関数を使用すると、この問題を解決できます。特定のローの rep_of_the_day を、対応する日までの SalesRepresentative の最後の NULL 以外の値と定義します。

関連情報

Window 関数 [445 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.7.7 標準偏差関数と平方偏差関数

平方偏差関数と標準偏差関数について、標本バージョンと母集団バージョンという 2 つのバージョンがサポートされています。

どちらのバージョンを選択するかは、その関数が使用される統計上のコンテキストによって変わります。

すべての平方偏差関数と標準偏差関数は、クエリの GROUP BY 句で決定されるローの分割について値を計算できるという点で、真の集合関数であるといえます。MAX や MIN などのその他の基本集合関数と同様に、入力の NULL 値は無視されます。

パフォーマンスを向上させるために、データベースサーバは平均と平均からの偏差を同時に計算します。そのためデータを 1 パスするだけですみます。

また、分析対象の式のドメインに関係なく、すべての平方偏差と標準偏差は IEEE 倍精度浮動小数点を使用して計算されます。平方偏差関数や標準偏差関数の入力が空のセットである場合、各関数は結果で NULL を返します。単一ローに対して VAR_SAMP が計算されると NULL が返されます。VAR_POP の場合は値 0 が返されます。

サポートされる標準偏差関数と平方偏差関数を次に示します。

- STDDEV 関数
- STDDEV_POP 関数
- STDDEV_SAMP 関数
- VARIANCE 関数
- VAR_POP 関数
- VAR_SAMP 関数

STDDEV 関数

この関数は、STDDEV_SAMP 関数のエイリアスです。

STDDEV_POP 関数

この関数は、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。

例

例 1

次のクエリは、部門の平均給与に標準偏差を加えたものよりも多い給与を得ている従業員を示す結果セットを返します。標準偏差は、データがどれだけ平均からばらつきがあるかを計るものです。

```
SELECT *
FROM ( SELECT
    Surname AS Employee,
    DepartmentID AS Department,
    CAST( Salary as DECIMAL( 10, 2 ) )
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
        AS Salary,
    CAST( AVG( Salary )
        OVER ( PARTITION BY DepartmentID ) AS DECIMAL ( 10, 2 ) )
        AS Average,
    CAST( STDDEV_POP( Salary )
        OVER ( PARTITION BY DepartmentID ) AS DECIMAL ( 10, 2 ) )
        AS StandardDeviation
    FROM Employees
    GROUP BY Department, Employee, Salary )
    AS DerivedTable
WHERE Salary > Average + StandardDeviation
ORDER BY Department, Salary, Employee;
```

次のテーブルは、クエリからの結果セットを示します。すべての部門で、少なくとも 1 人は平均から著しく外れた従業員がいます。

	Employee	Department	Salary	Average	StandardDeviation
1	Lull	100	87900.00	58736.28	16829.60
2	Scheffield	100	87900.00	58736.28	16829.60
3	Scott	100	96300.00	58736.28	16829.60
4	Sterling	200	64900.00	48390.95	13869.60
5	Savarino	200	72300.00	48390.95	13869.60
6	Kelly	200	87500.00	48390.95	13869.60
7	Shea	300	138948.00	59500.00	30752.40
8	Blaikie	400	54900.00	43640.67	11194.02
9	Morris	400	61300.00	43640.67	11194.02
10	Evans	400	68940.00	43640.67	11194.02
11	Martinez	500	55500.00	33752.20	9084.50

従業員 Scott は 96,300.00 ドルを得ていますが、部門の平均給与は 58,736.28 ドルです。この部門の標準偏差は 16,829.00 ドルです。つまり、75,565.88 ドル (`58736.28 + 16829.60 = 75565.88`) に満たない給与は、平均から標準偏差 1 個分の範囲内にあるということになります。従業員 Scott の給与は 96,300.00 ドルで、この値を超えています。

この例では、Surname と Salary が従業員ごとにユニークであることを想定していますが、ユニークである必要はありません。ユニーク性を保証するには、EmployeeID を GROUP BY 句に追加します。

例 2

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 SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Year	Quarter	Average	Variance
2000	1	25.775148	14.2794...
2000	2	27.050847	15.0270...
...	...	...	...

STDDEV_SAMP 関数

この関数は、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。たとえば、次の文は、異なる四半期における注文ごとの項目数で平均と平方偏差を返します。

```
SELECT YEAR( ShipDate ) AS Year,
    QUARTER( ShipDate ) AS Quarter,
    AVG( Quantity ) AS Average,
    STDDEV_SAMP( Quantity ) AS Variance
FROM SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

Year	Quarter	Average	Variance
2000	1	25.775148	14.3218...
2000	2	27.050847	15.0696...
...	...	...	...

VARIANCE 関数

この関数は、VAR_SAMP 関数のエイリアスです。

VAR_POP 関数

この関数は、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。たとえば、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 SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Year	Quarter	Average	Variance
2000	1	25.775148	203.9021...
2000	2	27.050847	225.8109...
...	...	...	...

単一ローに対して VAR_POP が計算されると値 0 が返されます。

VAR_SAMP 関数

この関数は、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。

たとえば、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 SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

このクエリは、次の結果を返します。

Year	Quarter	Average	Variance
2000	1	25.775148	205.1158...
2000	2	27.050847	227.0939...
...	...	...	...

単一ローに対して VAR_SAMP が計算されると値 NULL が返されます。

関連情報

集合関数に対応する数式 [473 ページ]

1.3.7.8 相関関数と線形回帰関数

さまざまな統計関数がサポートされています。関数の結果は、線形回帰の質を分析するのに役立ちます。

各関数の最初の引数は従属式 (Y で示される)、2 番目の引数は独立式 (X で示される) です。

COVAR_SAMP 関数

COVAR_SAMP 関数は、(Y, X) ペアのセットの標本共平方偏差を返します。

COVAR_POP 関数

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

COVAR_POP 関数は、(Y, X) ペアのセットの母共平方偏差を返します。

CORR 関数

CORR 関数は、(Y, X) ペアのセットの相関係数を返します。

REGR_AVGX 関数

REGR_AVGX 関数は、(Y, X) 値の NULL 以外のすべてのペアから x 値の平均を返します。

REGR_AVGY 関数

REGR_AVGY 関数は、(Y, X) 値の NULL 以外のすべてのペアから y 値の平均を返します。

REGR_SLOPE 関数

REGR_SLOPE 関数は、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_INTERCEPT 関数

REGR_INTERCEPT 関数は、従属変数と独立変数に最も適切な線形回帰直線の y 切片を計算します。

REGR_R2 関数

REGR_R2 関数は、回帰直線の決定係数 (**R-squared** または適合度とも呼ぶ) を計算します。

REGR_COUNT 関数

REGR_COUNT 関数は、入力で (Y, X) 値の NULL 以外のペアの数を返します。当該ペアの X と Y の両方が NULL 以外である場合にかぎり、線形回帰の計算で観測が使用されます。

REGR_SXX 関数

この関数は、(Y, X) ペアの x 値のセットの平方和を返します。

この関数の式は、標本平方偏差式や母平方偏差式の分子に相当します。その他の線形回帰関数と同様に、REGR_SXX は、入力で X と Y のどちらかが NULL であるような (Y, X) 値のペアを無視します。

REGR_SYY 関数

この関数は、(Y, X) ペアの Y 値のセットの平方和を返します。

REGR_SXY 関数

この関数は、(Y, X) ペアのセットに対して 2 つの積和の差を返します。

1.3.7.9 Window ランキング関数

Window ランキング関数は、分割内の他のローに関連するローのランクを返します。

サポートされているランキング関数は次のとおりです。

- CUME_DIST
- DENSE_RANK
- PERCENT_RANK
- RANK

ランキング関数は、SUM 集合関数などと同様の方法で複数の入力ローからの結果を計算しないため、集合関数とは見なされません。これらの関数は、特定の式の値に基づいて、分割内のローのランク (相対的な順序) を計算します。分割内のローの各セットは個別にランク付けされます。そのため OVER 句に PARTITION BY 句が含まれない場合は、入力全体が単一の分割として扱われます。このため、ランキング関数で使用されるウィンドウに対して、ROWS 句または RANGE 句は指定でき

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ません。複数のランキング関数を含むクエリを作成し、それぞれの関数が入力ローを異なる状態に分割またはソートするようにできます。

すべてのランキング関数では、ランキング関数が依存する入力ローのソート順序を指定するために ORDER BY 句が必要です。ORDER BY 句に複数の式が含まれる場合は、最初の式によって隣接ローで同じ値になるときに同順の発生を避けるために、2 番目以降の式が使用されます。NULL 値はその他の値よりも前にソートされます (昇順の場合)。

このセクションの内容:

Window ランキング関数: RANK 関数 [466 ページ]
RANK 関数は、その他のローの値と比較した現在のローの値のランクを返します。

Window ランキング関数: DENSE_RANK 関数 [468 ページ]
DENSE_RANK 関数は、その他のローの値と比較した現在のローの値のランクを返します。

Window ランキング関数: CUME_DIST 関数 [470 ページ]
累積分布関数 CUME_DIST は、百分位数の逆数として定義される場合があります。

Window ランキング関数: PERCENT_RANK 関数 [471 ページ]
PERCENT_RANK 関数は、ウィンドウの ORDER BY 句で指定されたカラムの値についてランクを返します。ただし、ランクは 0 ～ 1 の小数として表され、(RANK - 1)/(- 1) として計算されます。

1.3.7.9.1 Window ランキング関数: RANK 関数

RANK 関数は、その他のローの値と比較した現在のローの値のランクを返します。

値のランクは、値のリストがソートされた場合の順序を反映しています。

RANK 関数を使用すると、ウィンドウの ORDER BY 句で指定された式でランクが計算されます。ORDER BY 句に複数の式が含まれる場合は、最初の式によって隣接ローで同じ値になるときに同順の発生を避けるために、2 番目以降の式が使用されます。NULL 値はその他の値よりも前にソートされます (昇順の場合)。

例

例 1

次のクエリは、データベースで最もコストが高い製品 3 つを特定します。ウィンドウでは降順のソート順序が指定されるため、最もコストの高い製品はランクが最も低くなります。つまり、ランク付けは 1 から開始します。

```
SELECT Top 3 *
       FROM ( SELECT Description, Quantity, UnitPrice,
              RANK() OVER ( ORDER BY UnitPrice DESC ) AS Rank
              FROM Products ) AS DT
ORDER BY Rank;
```

このクエリは、次の結果を返します。

	Description	Quantity	UnitPrice	Rank
1	Zipped Sweatshirt	32	24.00	1
2	Hooded Sweatshirt	39	24.00	1

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	Description	Quantity	UnitPrice	Rank
3	Cotton Shorts	80	15.00	3

ロー 1 と 2 は、UnitPrice の値が同じであるため、ランクも同じになります。これを「同順」と呼びます。

RANK 関数では、同順の後はランクの値がジャンプします。たとえば、ロー 3 のランク値は 2 ではなく 3 にジャンプします。この動作は、同順の後でジャンプが発生しない DENSE_RANK 関数と異なります。

例 2

次の SQL クエリは、ユタ州の男性および女性従業員を検索し、給与が多い順にランクします。

```
SELECT Surname, Salary, Sex,
     RANK() OVER ( ORDER BY Salary DESC ) "Rank"
     FROM Employees
WHERE State IN ( 'UT' );
```

次のテーブルは、クエリからの結果セットを示します。

	Surname	Salary	Sex	Rank
1	Shishov	72995.00	F	1
2	Wang	68400.00	M	2
3	Cobb	62000.00	M	3
4	Morris	61300.00	M	4
5	Diaz	54900.00	M	5
6	Driscoll	48023.69	M	6
7	Hildebrand	45829.00	F	7
8	Goggin	37900.00	M	8
9	Rebeiro	34576.00	M	9
10	Bigelow	31200.00	F	10
11	Lynch	24903.00	M	11

例 3

データを分割して異なる結果になるように計算できます。例 2 のクエリを使用して、性別で分割することでデータを変更できます。次の例は、従業員を性別ごとに給与の多い順でランクしたものです。

```
SELECT Surname, Salary, Sex,
     RANK ( ) OVER ( PARTITION BY Sex
     ORDER BY Salary DESC ) "Rank"
     FROM Employees
WHERE State IN ( 'UT' );
```

次のテーブルは、クエリからの結果セットを示します。

	Surname	Salary	Sex	Rank
1	Wang	68400.00	M	1
2	Cobb	62000.00	M	2

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	Surname	Salary	Sex	Rank
3	Morris	61300.00	M	3
4	Diaz	54900.00	M	4
5	Driscoll	48023.69	M	5
6	Goggin	37900.00	M	6
7	Rebeiro	34576.00	M	7
8	Lynch	24903.00	M	8
9	Shishov	72995.00	F	1
10	Hildebrand	45829.00	F	2
11	Bigelow	31200.00	F	3

関連情報

Window ランキング関数: DENSE_RANK 関数 [468 ページ]

1.3.7.9.2 Window ランキング関数: DENSE_RANK 関数

DENSE_RANK 関数は、その他のローの値と比較した現在のローの値のランクを返します。

値のランクは、値のリストがソートされた場合の順序を反映しています。ランクは、ウィンドウの ORDER BY 句で指定された式で計算されます。

DENSE_RANK 関数は、ランク値にギャップ (ジャンプ) がなく単調に増加し続ける一連のランクを返します。RANK 値とは異なり、ランク値にジャンプがないことから DENSE (密) という語が使われます。

ウィンドウが入力ローを移動するのに合わせて、ウィンドウの ORDER BY 句で指定された式でランクが計算されます。ORDER BY 句に複数の式が含まれる場合は、最初の式によって隣接ローで同じ値になるときに同順の発生を避けるために、2 番目以降の式が使用されます。NULL 値はその他の値よりも前にソートされます (昇順の場合)。

例

例 1

次のクエリは、データベースで最もコストが高い製品 3 つを特定します。ウィンドウでは降順のソート順序が指定されるため、最もコストの高い製品はランクが最も低くなります。つまり、ランク付けは 1 から開始します。

```
SELECT Top 3 *
  FROM ( SELECT Description, Quantity, UnitPrice,
    DENSE_RANK( ) OVER ( ORDER BY UnitPrice DESC ) AS Rank
    FROM Products ) AS DT
  ORDER BY Rank;
```

このクエリは、次の結果を返します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	Description	Quantity	UnitPrice	Rank
1	Hooded Sweatshirt	39	24.00	1
2	Zipped Sweatshirt	32	24.00	1
3	Cotton Shorts	80	15.00	2

ロー 1 と 2 は、UnitPrice の値が同じであるため、ランクも同じになります。これを「同順」と呼びます。

DENSE_RANK 関数を使用した場合は、同順の後のランク値はジャンプしません。たとえば、ロー 3 のランク値は 2 です。この動作は、同順の後でランク値がジャンプする RANK 関数と異なります。

例 2

ウィンドウはクエリの GROUP BY 句の後に評価されるため、集合関数の値を基にランキングを判断するような複雑な要求を指定できます。

次のクエリは、地域ごとに総売り上げ上位 3 人の営業担当者と、地域ごとの総売り上げを生成します。

```
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 Products 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;
```

このクエリは、次の結果を返します。

	SalesRepresentative	Region	total_sales	sales_rank
1	299	Canada	9312.00	1
2	(NULL)	Canada	24768.00	1
3	1596	Canada	3564.00	2
4	856	Canada	2724.00	3
5	299	Central	32592.00	1
6	(NULL)	Central	134568.00	1
7	856	Central	14652.00	2
8	467	Central	14352.00	3
9	299	Eastern	21678.00	1
10	(NULL)	Eastern	142038.00	1
11	902	Eastern	15096.00	2
12	690	Eastern	14808.00	3
13	1142	South	6912.00	1
14	(NULL)	South	45262.00	1

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

	SalesRepresentative	Region	total_sales	sales_rank
15	667	South	6480.00	2
16	949	South	5782.00	3
17	299	Western	5640.00	1
18	(NULL)	Western	37632.00	1
19	1596	Western	5076.00	2
20	667	Western	4068.00	3

このクエリは、GROUPING SETS を使用して複数のグループ化を結合します。そのため、ウィンドウの WINDOW PARTITION 句では GROUPING 関数を使用して、特定の営業担当者を表すディテールローと、地域全体の総売り上げをリストする小計ローとを区別します。地域ごとの小計のローは、SalesRepresentative 属性に値 NULL がありますが、入力の分割ごとに結果のランキング順序が付けられるため、それぞれの小計ローのランキング値は 1 になります。これにより、ディテールローは 1 から適切にランク付けされます。

この例では、DENSE_RANK 関数により、総売り上げの集約について入力がランク付けされています。WINDOW ORDER 句では、エイリアスの設定された SELECT リスト項目が省略形として使用されます。

関連情報

Window ランキング関数: RANK 関数 [466 ページ]

1.3.7.9.3 Window ランキング関数: CUME_DIST 関数

累積分布関数 CUME_DIST は、百分位数の逆数として定義される場合があります。

CUME_DIST は、ウィンドウ内の値のセットに関して、特定値の正規化された位置を計算します。関数の範囲は 0 ～ 1 です。

ウィンドウが入力ローを移動するのに合わせて、ウィンドウの ORDER BY 句で指定された式で累積分布が計算されます。ORDER BY 句に複数の式が含まれる場合は、最初の式によって隣接ローで同じ値になるときに同順の発生を避けるために、2 番目以降の式が使用されます。NULL 値はその他の値よりも前にソートされます (昇順の場合)。

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 Employees
  WHERE State IN ( 'CA' );
```

このクエリは、次の結果を返します。

DepartmentID	Surname	Salary	Rank
200	Savarino	72300.00	0.333333333333333

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

DepartmentID	Surname	Salary	Rank
200	Clark	45000.00	0.666666666666667
200	Overbey	39300.00	1

1.3.7.9.4 Window ランキング関数: PERCENT_RANK 関数

PERCENT_RANK 関数は、ウィンドウの ORDER BY 句で指定されたカラムの値についてランクを返します。ただし、ランクは 0 ～ 1 の小数として表され、(RANK - 1)/(- 1) として計算されます。

ウィンドウが入力ローを移動するのに合わせて、ウィンドウの ORDER BY 句で指定された式でランクが計算されます。ORDER BY 句に複数の式が含まれる場合は、最初の式によって隣接ローで同じ値になるときに同順の発生を避けるために、2 番目以降の式が使用されます。NULL 値はその他の値よりも前にソートされます (昇順の場合)。

例

例 1

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返します。結果セットは、小数のパーセンテージを使用して降順にランキングされ、性別ごとに分けられます。

```
SELECT DepartmentID, Surname, Salary, Sex,
    PERCENT_RANK( ) OVER ( PARTITION BY Sex
      ORDER BY Salary DESC ) AS PctRank
  FROM Employees
  WHERE State IN ( 'NY' );
```

このクエリは、次の結果を返します。

	DepartmentID	Surname	Salary	Sex	PctRank
1	200	Martel	55700.000	M	0.0
2	100	Guevara	42998.000	M	0.333333333
3	100	Soo	39075.000	M	0.666666667
4	400	Ahmed	34992.000	M	1.0
5	300	Davidson	57090.000	F	0.0
6	400	Blaikie	54900.000	F	0.333333333
7	100	Whitney	45700.000	F	0.666666667
8	400	Wetherby	35745.000	F	1.0

入力は性別 (Sex) で分割されるため、PERCENT_RANK は男性と女性で個別に評価されます。

例 2

次の例は、ユタ州とアリゾナ州の女性従業員のリストを返し、給与の多い順にランクしたものです。PERCENT_RANK 関数は、降順で累積合計を計算するのに使用します。

```
SELECT Surname, Salary,
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
      PERCENT_RANK ( ) OVER ( ORDER BY Salary DESC ) "Rank"
      FROM Employees
WHERE State IN ( 'UT', 'AZ' ) AND Sex IN ( 'F' );
```

このクエリは、次の結果を返します。

	Surname	Salary	Rank
1	Shishov	72995.00	0
2	Jordan	51432.00	0.25
3	Hildebrand	45829.00	0.5
4	Bigelow	31200.00	0.75
5	Bertrand	29800.00	1

PERCENT_RANK を使用した最上位と最下位の百分位数の検索

PERCENT_RANK 関数を使用して、データセット内の最上位または最下位の百分位数を検索できます。次の例では、クエリは給与額についてデータセット内で上位 5 % の男性従業員を返します。

```
SELECT *
FROM ( SELECT Surname, Salary,
       PERCENT_RANK ( ) OVER ( ORDER BY Salary DESC ) "Rank"
       FROM Employees
       WHERE Sex IN ( 'M' )  )
       AS DerivedTable ( Surname, Salary, Percent )
WHERE Percent < 0.05;
```

このクエリは、次の結果を返します。

	Surname	Salary	Percent
1	Scott	96300.00	0
2	Sheffield	87900.00	0.025
3	Lull	87900.00	0.025

1.3.7.10 ローナンバリング関数

ローナンバリング関数は、分割内のローにユニークな番号を付けます。

NUMBER と ROW_NUMBER の 2 つのローナンバリング関数がサポートされています。ANSI 標準準拠の関数であり、NUMBER(*) 関数と同等の機能の多くを使用できるため、ROW_NUMBER 関数を使用してください。どちらの関数も同様のタスクを実行しますが、NUMBER 関数には、ROW_NUMBER 関数にはない制限がいくつかあります。

このセクションの内容:

ROW_NUMBER 関数の使用法 [473 ページ]
ROW_NUMBER 関数は、結果セットのローにユニークな番号を付けます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

[集合関数に対応する数式 [473 ページ]](#)
集合関数に使用される数式について学習します。

1.3.7.10.1 ROW_NUMBER 関数の使用法

ROW_NUMBER 関数は、結果セットのローにユニークな番号を付けます。

この関数はランキング関数ではありませんが、ランキング関数を使用できるどのような状況でも使うことができ、動作はランキング関数に似ています。

たとえば、派生テーブルで ROW_NUMBER を使用して、ROW_NUMBER の値について、ジョインであっても制限を追加できます。

```
SELECT *
FROM ( SELECT Description, Quantity,
       ROW_NUMBER( ) OVER ( ORDER BY ID ASC ) AS RowNum
FROM Products ) AS DT
WHERE RowNum <= 3
ORDER BY RowNum;
```

このクエリは、次の結果を返します。

Description	Quantity	RowNum
Tank Top	28	1
V-neck	54	2
Crew Neck	75	3

ランキング関数の場合と同様に、ROW_NUMBER には ORDER BY 句が必要です。

ウィンドウの ORDER BY 句がユニークでない式で構成される場合は、ROW_NUMBER は非決定的な結果を返すことがあり、同順が発生したときのローの順序は予測できなくなります。

ROW_NUMBER は、分割全体に対して機能するように設計されているため、ROWS 句や RANGE 句を ROW_NUMBER 関数とともに指定することはできません。

1.3.7.10.2 集合関数に対応する数式

集合関数に使用される数式について学習します。

単純な集合関数

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Function	Symbol	Formula
SUM(X)		$\sum_{i=1}^{n} x_i$
MAX(X)		$x_i : x_i \geq x_j, i \neq j \; \forall i, j \in n$
MIN(X)		$x_i : x_i \leq x_j, i \neq j \; \forall i, j \in n$
AVG(X)	$\bar{x}$	$\frac{\sum x_i}{n}$
COUNT(*)		n
VAR_SAMP(X)	s_x^2	$\frac{\sum (x_i - \bar{x})^2}{(n-1)}$
VAR_POP(X)	σ_x^2	$\frac{\sum (x_i - \bar{x})^2}{n}$
VARIANCE(X)		identical to VAR_SAMP(X)
STDDEV_SAMP(X)	s_x	$\sqrt{\frac{\sum (x_i - \bar{x})^2}{(n-1)}}$
STDDEV_POP(X)	σ_x	$\sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$
STDDEV(X)		identical to STDDEV_SAMP(X)

統計集合関数

COVAR_SAMP(Y,X)	*Co-variance*	$s_{xy} = \frac{\sum xy - \frac{(\sum x)(\sum y)}{n}}{(n-1)}$
COVAR_POP(Y,X)	*Co-variance*	$\sigma_{xy} = \frac{\sum xy - \frac{(\sum x)(\sum y)}{n}}{n}$
CORR(Y,X)	*Correlation Coefficient*	$r = \frac{\sum xy - \frac{1}{n}(\sum x)(\sum y)}{(n-1)s_x s_y}$
REGR_AVGX(Y,X)	*Independent mean*	$\bar{x}$
REGR_AVGY(Y,X)	*Dependent mean*	$\bar{y}$
REGR_SLOPE(Y,X)	*Regression Slope*	$b = r\frac{s_y}{s_x}$
REGR_INTERCEPT(Y,X)	*Regression Intercept*	$a = \bar{y} - b\bar{x}$
REGR_R2(Y,X)	*'Goodness-of-fit'*	r^2
REGR_COUNT(Y,X)	*Sample size*	n (non-null (Y, X) pairs)
REGR_SXX(Y,X)	*Sum of squares* (x)	$\sum x^2 - \frac{(\sum x)^2}{n}$
REGR_SYY(Y,X)	*Sum of squares* (y)	$\sum y^2 - \frac{(\sum y)^2}{n}$
REGR_SXY(Y,X)	*Sum of products*	$\sum xy - \frac{(\sum y)(\sum x)}{n}$

1.3.8 サブクエリの使用

リレーショナルデータベースを使用すると、複数のテーブルに関連するデータを保存できます。ジョインを使用して関連するテーブルからデータを抽出できるほか、サブクエリを使用しても抽出できます。

サブクエリとは、親の SQL 文の SELECT 句、WHERE 句、HAVING 句内にネストされた SELECT 文です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

サブクエリを使用すると、一部のクエリをジョインより簡単に記述できます。また、サブクエリを使用しないと記述できないクエリがあります。

サブクエリは、次のようなさまざまな方法に分類されます。

- 1 つ以上のローを返すかどうか (単一ローと複数ローのサブクエリ)
- 相関サブクエリか、非相関サブクエリか
- 別のサブクエリ内でネストしているかどうか

このセクションの内容:

単一ローのサブクエリと複数ローのサブクエリ [475 ページ]
外部の文に 1 つまたは 0 個のローを返すサブクエリを、単一ローのサブクエリと呼びます。

相関サブクエリと非相関サブクエリ [478 ページ]
サブクエリには、親の文に定義されたオブジェクトへの参照を含めることができます。このことは外部参照と呼ばれます。

ネストされたサブクエリ [479 ページ]
ネストされたサブクエリとは、別のサブクエリの中にネストされたサブクエリです。

ジョインに代わるサブクエリの使用 [480 ページ]
他のテーブルから要求されるカラムが 1 つだけである場合は、ジョインの代わりにサブクエリを使用できます。

WHERE 句でのサブクエリ [481 ページ]
WHERE 句内のサブクエリは、ロー選択のプロセスの一部として機能します。

HAVING 句でのサブクエリ [482 ページ]
サブクエリは通常は WHERE 句内で探索条件として使用しますが、クエリの HAVING 句で使用することもできます。

サブクエリを使用した述部 [483 ページ]
サブクエリでサポートされている検索条件は多数あります。

オプティマイザによるサブクエリからジョインへの自動変換 [491 ページ]
クエリオプティマイザは、サブクエリを利用するクエリの多くをジョインとして自動的に書き換えます。

1.3.8.1 単一ローのサブクエリと複数ローのサブクエリ

外部の文に 1 つまたは 0 個のローを返すサブクエリを、単一ローのサブクエリと呼びます。

比較演算子の有無にかかわらず、単一ローのサブクエリは SQL 文のどこでも使用できます。

たとえば、単一ローのサブクエリは、SELECT 句の式で使用できます。

```
SELECT (select FIRST T.x FROM T) + 1 as ITEM_1, 2 as ITEM_2,...
```

また、単一ローのサブクエリは、比較演算子とともに SELECT 句の式で使用できます。

次に例を示します。

```
SELECT IF (select FIRST T.x FROM T) >= 10 THEN 1 ELSE 0 ENDIF as ITEM_1, 2 as
ITEM_2,...
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

外部の文に複数のロー (ただしカラムは 1 つだけ) を返すサブクエリを、複数ローのサブクエリと呼びます。複数ローのサブクエリは、IN 句、ANY 句、ALL 句または EXISTS 句で使用されるサブクエリです。

例

例 1: 単一ローのサブクエリ

テーブル Products に製品だけの情報を、別のテーブル SalesOrdersItems には注文関連の情報を保存します。Products テーブルにはさまざまな製品の情報が入っています。SalesOrdersItems テーブルには、顧客の注文についての情報が入っています。在庫数が 50 未満になったときに製品を再注文する場合、次のクエリで「在庫数が少ない製品は何か」という問い合わせに対する回答を得ることができます。

```
SELECT ID, Name, Description, Quantity
FROM Products
WHERE Quantity < 50;
```

ただし、ほとんど注文されない製品がわずかしかないことよりも頻繁に購入される製品が少ないことの方が関心が高いため、製品が注文される頻度を考慮すると便利です。

サブクエリを使用してある顧客が注文する品目の平均数を決定し、その平均をメインクエリに使用して在庫数が少ない製品を検索します。次に示すクエリは、顧客が注文した各タイプの平均品目数の 2 倍未満の製品名とその説明を検索します。

```
SELECT Name, Description
FROM Products WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems
   );
```

WHERE 句では、クエリ結果に含まれる、FROM 句にリストされるテーブルからローを選択するのにサブクエリが役立ちます。HAVING 句では、クエリ結果に含まれる、メインクエリの GROUP BY 句で指定されるローのグループを選択するのに役立ちます。

例 2: 単一ローのサブクエリ

次の単一ローのサブクエリの例では、Products テーブルの製品の平均価格を計算します。そしてその平均は外部クエリの WHERE 句に渡されます。外部クエリは、平均より低い価格のすべての製品の ID、Name、UnitPrice を返します。

```
SELECT ID, Name, UnitPrice
FROM Products
WHERE UnitPrice <
  ( SELECT AVG( UnitPrice ) FROM Products )
ORDER BY UnitPrice DESC;
```

ID	Name	UnitPrice
401	Baseball Cap	10.00
300	Tee Shirt	9.00
400	Baseball Cap	9.00
500	Visor	7.00
501	Visor	7.00

例 3: IN を使用した単純な複数ローのサブクエリ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

在庫数が少ない品目を識別し、一方でそれらの品目に対する注文も識別したいとします。次のように、WHERE 句にサブクエリを含む SELECT 文を実行します。

```
SELECT *
FROM SalesOrderItems
WHERE ProductID IN
   (  SELECT ID
       FROM Products
       WHERE Quantity < 20 )
ORDER BY ShipDate DESC;
```

この例では、サブクエリは Products テーブル内の ID カラムにおいて WHERE 句の探索条件を満たすすべての値のリストを作成します。そして一連のローが返されますが、返されるカラムは 1 つだけです。IN キーワードは、それぞれの値をセットのメンバーとして扱い、メインクエリ内の各ローがセットのメンバーかどうかをテストします。

例 4: IN、ANY、ALL の使用を比較する複数ローのサブクエリ

サンプルデータベースには、経理に関するデータを格納するテーブルが 2 つあります。FinancialCodes テーブルは、経理データとこれらの意味についてのさまざまなコードが入っているテーブルです。FinancialData テーブルから歳入項目をリストするには、次のクエリを実行します。

```
SELECT *
FROM FinancialData
WHERE Code IN
    ( SELECT Code
        FROM FinancialCodes
        WHERE type = 'revenue' );
```

Year	Quarter	Code	Amount
1999	Q1	r1	1023
1999	Q2	r1	2033
1999	Q3	r1	2998
1999	Q4	r1	3014
2000	Q1	r1	3114
...	...	...	...

ANY キーワードと ALL キーワードも同様の方法で使用できます。たとえば、次のクエリは前述のクエリと同じ結果を返しますが、ANY キーワードを使用しています。

```
SELECT *
FROM FinancialData
WHERE FinancialData.Code = ANY
   (  SELECT FinancialCodes.Code
       FROM FinancialCodes
       WHERE type = 'revenue' );
```

=ANY 条件は IN 条件とまったく同じですが、ANY を < や > などの不等号とともに使用すると、サブクエリをより柔軟に使用できます。

ALL キーワードは ANY に似ています。たとえば、次のクエリは歳入以外の経理データをリストします。

```
SELECT *
FROM FinancialData
WHERE FinancialData.Code <> ALL
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    (  SELECT FinancialCodes.Code
        FROM FinancialCodes
       WHERE type = 'revenue' );
```

このクエリは、NOT IN を使用した場合の次の文と同じです。

```
SELECT *
FROM FinancialData
WHERE FinancialData.Code NOT IN
    (  SELECT FinancialCodes.Code
       FROM FinancialCodes
       WHERE type = 'revenue' );
```

1.3.8.2 相関サブクエリと非相関サブクエリ

サブクエリには、親の文に定義されたオブジェクトへの参照を含めることができます。このことは外部参照と呼ばれます。

外部参照があるサブクエリは相関サブクエリと呼ばれます。相関サブクエリは、外部クエリとは別に評価することはできません。これは、サブクエリが親の文の値を使用するためです。つまり、サブクエリは親の文のローごとに実行されます。したがって、サブクエリの結果は、親の文で評価されるアクティブなローに依存します。

たとえば、次の文のサブクエリは、Products テーブル内のアクティブなローに依存する値を返します。

```
SELECT Name, Description
FROM Products
WHERE Quantity < 2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems
   WHERE Products.ID=SalesOrderItems.ProductID );
```

この例では、このサブクエリの Products.ID カラムは外部参照です。このクエリは、在庫数が平均注文数の 2 倍より少ない製品、具体的には、メインクエリの WHERE 句によってテストされている製品の、名前と説明を抽出します。サブクエリは SalesOrderItems テーブルをスキャンしてこれを実行します。しかし、サブクエリの WHERE 句にある Products.ID カラムは、サブクエリではなく、メインクエリの FROM 句に指定されているテーブルのカラムを参照します。データベースサーバは Products テーブルの各ローの間を移動して、サブクエリの WHERE 句を評価するときに、現在のローの ID 値を使用します。

サブクエリで参照されるカラムが、サブクエリの FROM 句で参照されるテーブルになくても、外部クエリの FROM 句で参照されるテーブルにあれば、クエリはエラーなく実行されます。データベースサーバでは、サブクエリのカラムが外部クエリのテーブル名で暗黙的に修飾されます。

親の文にオブジェクトへの参照を含まないサブクエリを、非相関サブクエリと呼びます。次の例では、サブクエリは正確に 1 つの値、SalesOrderItems テーブルの平均数を計算します。クエリを評価するときに、データベースサーバはこの値を一度計算し、その値を Products テーブルの Quantity フィールドにあるそれぞれの値と比較して、対応するローを選択するかどうかを決定します。

```
SELECT Name, Description
FROM Products
WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.8.3 ネストされたサブクエリ

ネストされたサブクエリとは、別のサブクエリの中にネストされたサブクエリです。

定義できるサブクエリのネストのレベルに制限はありませんが、3 つ以上のレベルのクエリは、それ以下のレベルのクエリに比べて、実行にかなりの時間がかかります。

次の例では、ネストされたサブクエリを使用して、Fees 部の任意の品目が注文された日に出荷された注文の注文 ID とライン ID を決定します。

```
SELECT ID, LineID
FROM SalesOrderItems
WHERE ShipDate = ANY (
   SELECT OrderDate
   FROM SalesOrders
   WHERE FinancialCode IN (
      SELECT Code
      FROM FinancialCodes
      WHERE ( Description = 'Fees' ) ) );
```

ID	LineID
2001	1
2001	2
2001	3
2002	1
...	...

この例では、最も内側のサブクエリが、"Fees" という説明がある経理コードのカラムを生成します。

```
SELECT Code
FROM FinancialCodes
WHERE ( Description = 'Fees' );
```

次のサブクエリは、最も内側のサブクエリが選択したコードに一致するコードを持つ品目の注文日を検索します。

```
SELECT OrderDate
FROM SalesOrders
WHERE FinancialCode
IN ( subquery-expression );
```

最後に、一番外側のクエリが、サブクエリの検索した日付のいずれかに出荷された注文の注文 ID とライン ID を検索します。

```
SELECT ID, LineID
FROM SalesOrderItems
WHERE ShipDate = ANY ( subquery-expression );
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.8.4 ジョインに代わるサブクエリの使用

他のテーブルから要求されるカラムが 1 つだけである場合は、ジョインの代わりにサブクエリを使用できます。

注文とその発注先の会社を時系列にリストする必要があり、顧客 ID の代わりに会社名を使いたいとします。この結果を得るには、ジョインを使用します。

ジョインの使用

2001 年 1 月 1 日以降の受注 ID、日付、各注文を行った会社名をリストするには、次のクエリを実行します。

```
SELECT SalesOrders.ID,
            SalesOrders.OrderDate,
            Customers.CompanyName
FROM SalesOrders
   KEY JOIN Customers
WHERE OrderDate > '2001/01/01'
ORDER BY OrderDate;
```

サブクエリの使用方法

次の SQL 文は、ジョインではなくサブクエリを使用して同じ結果を得ます。

```
SELECT SalesOrders.ID,
   SalesOrders.OrderDate,
   (  SELECT CompanyName FROM Customers
       WHERE Customers.ID = SalesOrders.CustomerID )
FROM SalesOrders
WHERE OrderDate > '2001/01/01'
ORDER BY OrderDate;
```

SalesOrders テーブルがサブクエリの一部でない場合でも、サブクエリは SalesOrders テーブル内の CustomerID カラムを参照します。一方、SalesOrders.CustomerID カラムは SQL 文の本文にある SalesOrders テーブルを参照します。

この例では CompanyName カラムだけを必要としていたので、ジョインをサブクエリに変更することができました。

外部ジョインの使用

ワシントン州在住の顧客名すべてとその顧客の最も最近の受注 ID をリストするには、次のクエリを実行します。

```
SELECT  CompanyName, State,
   ( SELECT MAX( ID )
       FROM SalesOrders
      WHERE SalesOrders.CustomerID = Customers.ID )
FROM Customers
WHERE State = 'WA';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

CompanyName	State	MAX(SalesOrders.ID)
Custom Designs	WA	2547
It's a Hit!	WA	(NULL)

It's a Hit! という会社は何も注文しなかったので、サブクエリはこの顧客については NULL を返します。内部ジョインを使用した場合、発注しなかった会社はリストされません。

外部ジョインを明示的に指定することもできます。その場合は、次のように GROUP BY 句も指定する必要があります。

```
SELECT CompanyName, State,
   MAX( SalesOrders.ID )
FROM Customers
   KEY LEFT OUTER JOIN SalesOrders
WHERE State = 'WA'
GROUP BY CompanyName, State;
```

1.3.8.5 WHERE 句でのサブクエリ

WHERE 句内のサブクエリは、ロー選択のプロセスの一部として機能します。

ローの選択に使用する基準が別のテーブルの結果に依存するときに、WHERE 句内にサブクエリを使用します。

例

在庫数が平均注文数の 2 倍よりも少ない製品を検索します。

```
SELECT Name, Description
FROM Products WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems );
```

このクエリは 2 段階で実行されます。まず、注文ごとに要求される品目の平均数を検索します。次に、どの製品の在庫数がその数の 2 倍より少ないかを検索します。

2 段階のクエリ

要求される品目の数は、品目のタイプ、顧客、注文ごとに、SalesOrderItems テーブルの Quantity カラムに格納されます。サブクエリは次のようになります。

```
SELECT AVG( Quantity )
FROM SalesOrderItems;
```

これによって SalesOrderItems テーブルの品目の平均数、25.851413 が返されます。

次のクエリは、前述のクエリで抽出した値の 2 倍よりも少ない在庫数の品目の名前とその説明を返します。

```
SELECT Name, Description
FROM Products
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
WHERE Quantity < 2*25.851413;
```

サブクエリを使用すると、この 2 つの手順を 1 つのオペレーションにまとめることができます。

WHERE 句でのサブクエリの目的

WHERE 句内でのサブクエリは、探索条件の一部です。

関連情報

クエリ [160 ページ]

1.3.8.6 HAVING 句でのサブクエリ

サブクエリは通常は WHERE 句内で探索条件として使用しますが、クエリの HAVING 句で使用することもできます。

HAVING 句内のサブクエリは、ローグループの選択の一部として使用されます。

"どの製品の平均在庫数が、顧客ごとの各品目の平均注文数の 2 倍以上あるのか" という要求は、HAVING 句内にサブクエリを持つクエリになります。

例

```
SELECT Name, AVG( Quantity )
FROM Products
GROUP BY Name
HAVING AVG( Quantity ) > 2* (
   SELECT AVG( Quantity )
   FROM SalesOrderItems
 );
```

name	AVG(Products.Quantity)
Baseball Cap	62.000000
Shorts	80.000000
Tee Shirt	52.333333

クエリは次のように実行します。

- サブクエリは SalesOrderItems テーブルにある品目の平均数を計算します。
- メインクエリは Products テーブルを調べて、製品ごとの平均数を計算し、製品名でグループ化します。
- HAVING 句は、各平均数がサブクエリで検索された数量の 2 倍を超えるかどうかを確認します。超える場合、メインクエリはそのローグループを返します。超えない場合は返しません。
- SELECT 句は、グループごとに 1 つの計算ローを生成し、各製品の名前と在庫の平均数を示します。

次の例で示すように、HAVING 句には外部参照も使用できます。この例は、前述の例を若干変更したものです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

この例では、平均注文数が在庫数の半分よりも多い製品の ID 番号とライン ID 番号を検索します。

```
SELECT ProductID, LineID
FROM SalesOrderItems
GROUP BY ProductID, LineID
HAVING 2* AVG( Quantity ) > (
   SELECT Quantity
   FROM Products
   WHERE Products.ID = SalesOrderItems.ProductID );
```

ProductID	LineID
601	3
601	2
601	1
600	2
...	...

この例では、サブクエリは、HAVING 句にテストされるローグループに対応する製品の在庫数を生成します。サブクエリは外部参照 SalesOrderItems.ProductID を使用して、その特定製品のレコードを選択します。

比較演算子を持つサブクエリは 1 つの値を返す

このクエリは比較演算子 > を使用するので、サブクエリは 1 つの値を返します。この場合は、1 つの値を返します。Products テーブルの ID フィールドがプライマリキーなので、特定の製品 ID に対応する Products テーブルのレコードは 1 つだけになります。

1.3.8.7 サブクエリを使用した述部

サブクエリでサポートされている検索条件は多数あります。

サブクエリを使用した述部の比較

メインクエリのテーブルにある各レコードについて、サブクエリが生成した 1 つの値と式の値を比較します。比較テストでは、サブクエリで提供される演算子 (=、<>、<、<=、>、<=) を使用します。

限定比較テスト

サブクエリが生成した値のそれぞれのセットと式の値を比較します。

サブクエリセットメンバーシップテスト

サブクエリが生成した値のセットのいずれかと、式の値が一致するかどうかを調べます。

存在テスト

サブクエリがローを生成するかどうかを調べます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

サブクエリ比較テスト [484 ページ]
サブクエリの比較テスト (=、<>、<、<=、>、>=) は、単純な比較テストを変更したものです。

サブクエリと IN テスト [485 ページ]
サブクエリセットメンバーシップテストを使用して、メインクエリからの値をサブクエリの複数の値と比較できます。

サブクエリと ANY テスト [487 ページ]
ANY テストは、SQL 比較演算子 (=、>、<、>=、<=、!=、<>、!>、!<) のいずれかと組み合せて使用して、1 つの値をサブクエリが生成するデータ値のカラムと比較します。

サブクエリと ALL テスト [488 ページ]
ALL テストは、SQL 比較演算子 (=、>、<、>=、<=、!=、<>、!>、!<) のいずれかと組み合せて使用して、1 つの値をサブクエリが生成するデータ値と比較します。

サブクエリと EXISTS テスト [489 ページ]
サブクエリ比較テストとセットメンバーシップテストに使用されるサブクエリは、いずれもサブクエリテーブルからデータ値を返します。

関連情報

クエリ [160 ページ]

1.3.8.7.1 サブクエリ比較テスト

サブクエリの比較テスト (=、<>、<、<=、>、>=) は、単純な比較テストを変更したものです。

サブクエリの比較テストでは、演算子の後ろに来る式がサブクエリになる点だけが異なります。このテストを使用して、メインクエリのローからの値を、サブクエリが生成する *1* つの値と比較します。

例

このクエリにはサブクエリ比較テストの例が含まれています。

```
SELECT Name, Description, Quantity
FROM Products
WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems );
```

name	Description	Quantity
Tee Shirt	Tank Top	28
Baseball Cap	Wool cap	12
Visor	Cloth Visor	36
Visor	Plastic Visor	28

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

name	Description	Quantity
...	...	...

次のサブクエリは単一の値、つまり各顧客が発注したタイプ別平均品目数を、SalesOrderItems テーブルから取り出します。

```
SELECT AVG( Quantity )
FROM SalesOrderItems;
```

メインクエリは、各品目の在庫数をその値と比較します。

比較テストのサブクエリは 1 つの値を返す

比較テストのサブクエリは 1 つの値を返します。次の例では、SalesOrderItems テーブルから 2 つのカラムを抽出するサブクエリを持つクエリを考えてみます。

```
SELECT Name, Description, Quantity
FROM Products
WHERE Quantity <  2 * (
   SELECT AVG( Quantity ), MAX( Quantity )
   FROM SalesOrderItems);
```

エラーが返されます。

1.3.8.7.2 サブクエリと IN テスト

サブクエリセットメンバーシップテストを使用して、メインクエリからの値をサブクエリの複数の値と比較できます。

サブクエリセットメンバーシップテストは、メインクエリの各ローの 1 つのデータ値を、サブクエリが生成したデータ値の 1 つのカラムと比較します。メインクエリのデータ値がカラムのデータ値のいずれかと一致する場合、サブクエリは TRUE を返します。

例

Shipping 部または Finance 部の部長である従業員の名前を選択します。

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID IN (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
           DepartmentName = 'Shipping' ) );
```

GivenName	Surname
Mary Anne	Shea

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

GivenName	Surname
Jose	Martinez

この例のサブクエリは、Shipping 部と Finance 部の部長に対応する ID 番号を、Departments テーブルから抽出します。次にメインクエリが、サブクエリによって検索された 2 つの値のいずれかに一致する ID 番号を持つ従業員の名前を返します。

```
SELECT DepartmentHeadID
FROM Departments
WHERE ( DepartmentName='Finance' OR
       DepartmentName = 'Shipping' );
```

セットメンバーシップテストは =ANY テストと同等

サブクエリセットメンバーシップテストは =ANY テストと同等です。次のクエリは前述の例のクエリと同等です。

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID = ANY (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );
```

セットメンバーシップテストの否定

サブクエリセットメンバーシップテストは、サブクエリによって生成される値に一致しないカラム値を持つローを抽出する場合にも使用できます。セットメンバーシップテストを否定するには、キーワード IN の前に NOT を挿入します。

例

このクエリのサブクエリは、Finance 部または Shipping 部の部長でない従業員の姓と名前を返します。

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID NOT IN (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.8.7.3 サブクエリと ANY テスト

ANY テストは、SQL 比較演算子 (=、>、<、>=、<=、!=、<>、!>、!<) のいずれかと組み合せて使用して、1 つの値をサブクエリが生成するデータ値のカラムと比較します。

テストを実行するには、SQL は指定された比較演算子を使用して、テスト値をカラムのデータ値のそれぞれと比較します。いずれかの比較の結果が TRUE になる場合、ANY テストは TRUE を返します。

ANY を使用するサブクエリは 1 つのカラムを返します。

例

注文番号 2005 の最初の製品が出荷された後に受けた注文の注文 ID と顧客 ID を検索します。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ANY (
   SELECT ShipDate
   FROM SalesOrderItems
   WHERE ID=2005 );
```

ID	CustomerID
2006	105
2007	106
2008	107
2009	108
...	...

このクエリを実行すると、メインクエリは、注文番号 2005 のすべての製品の出荷日に対して、各注文の注文日をテストします。注文日が注文番号 2005 の *1* つの出荷の出荷日より後であれば、SalesOrders テーブルの注文 ID と顧客 ID が結果セットに示されます。このように ANY テストは OR 演算子に似ています。前述のクエリは、"この注文は注文番号 2005 の最初の製品が出荷された後に受けたものか、または注文番号 2005 の 2 番目の製品が出荷された後に受けたものか、または ..." というように解釈できます。

ANY 演算子の知識

ANY 演算子はやや複雑な場合があります。このクエリは、「注文番号 2005 の任意の製品が出荷された後に受けた注文を返す」と解釈してしまいがちです。しかし、それでは注文番号 2005 のすべての製品が出荷された後に受けた注文の注文 ID と顧客 ID を返すことになり、クエリの動作と異なります。

そうではなく、"注文番号 2005 の少なくとも *1* つの製品が出荷された後に受けた注文の注文 ID と顧客 ID を返す" というようにクエリを解釈してみます。キーワード SOME を使用すると、もう少し直感的な方法でクエリを表現できます。次のクエリは前述のクエリと同等です。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > SOME (
   SELECT ShipDate
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    FROM SalesOrderItems
    WHERE ID=2005 );
```

キーワード SOME はキーワード ANY と同等です。

ANY 演算子についての注意

ANY テストには、このほかに 2 つの重要な特徴があります。

空のサブクエリの結果セット

サブクエリが空の結果セットを生成する場合、ANY テストは FALSE を返します。結果がない場合、少なくとも 1 つの結果が比較テストを満たしているというのは真ではないので、これは理にかなっています。

サブクエリの結果セットの **NULL** 値

サブクエリの結果セットには少なくとも 1 つの NULL 値があることが前提です。結果セットの NULL 以外のすべてのデータ値に対して比較テストが FALSE の場合、ANY は UNKNOWN を返します。これは、比較テストが保持するサブクエリの値があるかどうか、この状況では確定できないためです。値があるかどうかは、結果セットの NULL データの正確な値によって異なります。

1.3.8.7.4 サブクエリと ALL テスト

ALL テストは、SQL 比較演算子 (=、>、<、>=、<=、!=、<>、!>、!<) のいずれかと組み合せて使用して、1 つの値をサブクエリが生成するデータ値と比較します。

テストを実行するには、SQL は指定された比較演算子を使用して、テスト値を結果セットのデータ値のそれぞれと比較します。すべての比較の結果が TRUE になる場合、ALL テストは TRUE を返します。

例

次の例は、注文番号 2001 のすべての製品が出荷された後に受けた注文の注文 ID と顧客 ID を検索します。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ALL (
   SELECT ShipDate
   FROM SalesOrderItems
   WHERE ID=2001 );
```

ID	CustomerID
2002	102
2003	103
2004	104
2005	101
...	...

このクエリを実行すると、メインクエリは、注文番号 2001 のすべての製品の出荷日に対して、各注文の注文日をテストします。注文日が注文番号 2001 のすべての出荷の出荷日より後であれば、SalesOrders テーブルの注文 ID と顧客 ID が

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果セットに示されます。このように ALL テストは AND 演算子に似ています。前述のクエリは、"この注文は注文番号 2001 の最初の製品が出荷される前に受けたもので、なおかつ注文番号 2001 の 2 番目の製品が出荷される前に受けたもので、なおかつ ..." というように解釈できます。

ALL 演算子についての注意

ALL テストには、このほかに 3 つの重要な特徴があります。

空のサブクエリの結果セット

サブクエリが空の結果セットを生成した場合、ALL テストは TRUE を返します。結果がない場合、比較テストが結果セットのどの値に対しても適用しているというのは真なので、これは理にかなっています。

サブクエリの結果セットの **NULL** 値

結果セットのいずれかの値に対する比較テストが FALSE の場合、ALL は FALSE を返します。すべての値が TRUE の場合は TRUE を返します。それ以外の場合は、UNKNOWN を返します。たとえば、サブクエリの結果セットに NULL 値があっても、NULL 以外のすべての値の探索条件が TRUE の場合などです。

ALL テストの否定

次の 2 つの式は同じではありません。

```
NOT a = ALL (subquery)
a <> ALL (subquery)
```

関連情報

ANY、ALL、または SOME に続くサブクエリ [493 ページ]

1.3.8.7.5 サブクエリと EXISTS テスト

サブクエリ比較テストとセットメンバーシップテストに使用されるサブクエリは、いずれもサブクエリテーブルからデータ値を返します。

しかし、場合によっては、どの結果をサブクエリが返すのかではなく、サブクエリが何らかの結果を返すのかどうかが重要であることがあります。存在テスト (EXISTS) は、サブクエリがクエリ結果のローを生成するかどうかを調べます。サブクエリが 1 つ以上の結果のローを返す場合、EXISTS テストは TRUE を返します。結果のローを返さない場合は、FALSE を返します。

例

ここでは、"2001 年 7 月 13 日よりも後に発注したのはどの顧客か" という要求を、サブクエリを使って表現してみます。

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
   SELECT *
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    FROM SalesOrders
    WHERE ( OrderDate > '2001-07-13' ) AND
          ( Customers.ID = SalesOrders.CustomerID ) );
```

GivenName	Surname
Almen	de Joie
Grover	Pendelton
Ling Ling	Andrews
Bubba	Murphy

存在テストの説明

この例では、サブクエリが、Customers テーブルのローごとに、その顧客 ID が 2001 年 7 月 13 日より後に発注した顧客 ID に対応するかどうかを調べます。対応していれば、クエリはその顧客の姓と名前をメインテーブルから抽出します。

EXISTS テストはサブクエリの結果を使用しません。単にサブクエリがローを生成するかどうかを調べるだけです。このため、次の 2 つのサブクエリに適用した存在テストでも同じ結果が返されます。これらはサブクエリですから、それ自体では処理できません。サブクエリが参照する Customers テーブルは、メインクエリの一部であってサブクエリの一部ではないからです。

```
SELECT *
FROM Customers, SalesOrders
WHERE ( OrderDate > '2001-07-13' ) AND
      ( Customers.ID = SalesOrders.CustomerID )
SELECT OrderDate
FROM Customers, SalesOrders
WHERE ( OrderDate > '2001-07-13' ) AND
      ( Customers.ID = SalesOrders.CustomerID );
```

便宜上、"SELECT *" という表記を使用していますが、SalesOrders テーブルのどのカラムが SELECT 文に指定されるかどうかは問題ではありません。

存在テストの否定

EXISTS テストの論理は、NOT EXISTS という形式で否定できます。この場合、テストはサブクエリがローを返さない場合に TRUE を、ローを返す場合に FALSE を返します。

相関サブクエリ

サブクエリには Customers テーブルからの ID カラムへの参照が含まれています。メインテーブル内のカラムや式への参照は、外部参照と呼ばれます。また、そのサブクエリは相関です。概念的には、SQL は Customers テーブルを調べ、顧客ごとにサブクエリを実行して、前述のクエリを処理します。SalesOrders テーブルの注文日が 2001 年 7 月 13 日より後で、Customers テーブルと SalesOrders テーブルの顧客 ID が一致していれば、Customers テーブルからの姓と名前が表示さ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

れます。サブクエリはメインクエリを参照するので、上述のサブクエリは、前項のサブクエリとは異なり、サブクエリをそれだけで実行しようとするとエラーが返されます。

関連情報

相関サブクエリと非相関サブクエリ [478 ページ]

1.3.8.8 オプティマイザによるサブクエリからジョインへの自動変換

クエリオプティマイザは、サブクエリを利用するクエリの多くをジョインとして自動的に書き換えます。

変換はユーザによるアクションを必要とすることなく実行されます。データベースでのクエリのパフォーマンスを理解できるように、いくつかのサブクエリをジョインに変換することができます。

マルチレベルのクエリをジョインに書き換えるために満たす必要がある基準は、演算子のタイプ、クエリの構造、サブクエリの構造によって異なります。サブクエリが WHERE 句内にある場合は、次のフォームになることに注意してください。

```
SELECT select-list
FROM table
WHERE
[NOT] expression comparison-operator ( subquery-expression )
| [NOT] expression comparison-operator { ANY | SOME } ( subquery-expression )
| [NOT] expression comparison-operator ALL ( subquery-expression )
| [NOT] expression IN ( subquery-expression )
| [NOT] EXISTS ( subquery-expression )
GROUP BY group-by-expression
HAVING search-condition
```

たとえば、「Mrs. Clarke と Suresh がいつ注文し、どの担当者が注文を受けたか」という要求を考えてみます。これは次のクエリを使用して回答できます。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID IN (
   SELECT ID
   FROM Customers
   WHERE Surname = 'Clarke' OR GivenName = 'Suresh' );
```

OrderDate	SalesRepresentative
2001-01-05	1596
2000-01-27	667
2000-11-11	467
2001-02-04	195
...	...

サブクエリは WHERE 句に名前がリストされている 2 人の顧客に対応する顧客 ID のリストを生成します。メインクエリはこの 2 人の注文に対応する注文日と担当者を検索します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

同じ問い合わせをジョインを使用して応答できます。このクエリの、2 つのテーブルのジョインを使用した代替形式を次に示します。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders, Customers
WHERE CustomerID=Customers.ID AND
   ( Surname = 'Clarke' OR GivenName = 'Suresh' );
```

この形式のクエリは SalesOrders テーブルを Customers テーブルにジョインして各顧客の注文を検索し、Suresh と Clarke のレコードだけを返します。

サブクエリは有効でもジョインが有効でない場合

サブクエリは有効でも、ジョインが有効でない場合があります。次に例を示します。

```
SELECT Name, Description, Quantity
FROM Products
WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems );
```

name	Description	Quantity
Tee Shirt	Tank Top	28
Baseball Cap	Wool cap	12
Visor	Cloth Visor	36
...	...	...

この場合、内部クエリは集計クエリで外部クエリは集計クエリではないので、2 つのクエリを簡単なジョインで組み合わせることはできません。

このセクションの内容:

比較演算子に続くサブクエリ [493 ページ]
比較演算子に続くサブクエリ (=、>、<、>=、<=、!=、<>、!>、!<) は、比較と呼ばれます。

ANY、ALL、または SOME に続くサブクエリ [493 ページ]
キーワード ALL、ANY、または SOME に続くサブクエリは、限定比較と呼ばれます。

IN に続くサブクエリ [496 ページ]
一定の基準を満たす場合に、オプティマイザは、IN キーワードが続くサブクエリのみを変換します。

EXISTS に続くサブクエリ [497 ページ]
一定の基準を満たす場合に、オプティマイザは、EXISTS キーワードに続くサブクエリを変換します。

関連情報

ジョイン: 複数テーブルからのデータ検索 [376 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.8.8.1 比較演算子に続くサブクエリ

比較演算子に続くサブクエリ (=、>、<、>=、<=、!=、<>、!>、!<) は、比較と呼ばれます。

オプティマイザは、サブクエリが次のような場合に、これらのサブクエリをジョインに変換します。

- メインクエリのローごとに値を 1 つずつ返す
- GROUP BY 句を含んでいない
- キーワード DISTINCT を含んでいない
- UNION クエリではない
- 集計クエリではない

例

"Suresh の製品がいつ注文され、どの担当者が注文を受けたか" という要求をサブクエリで表現したとします。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID = (
   SELECT ID
   FROM Customers
   WHERE GivenName = 'Suresh' );
```

このクエリは基準を満たすので、ジョインを使用するクエリに変換できます。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders, Customers
WHERE CustomerID=Customers.ID AND
  ( Surname = 'Clarke' OR GivenName = 'Suresh' );
```

しかし、「在庫数が平均注文数の 2 倍よりも少ない製品を検索する」という要求はジョインに変換できません。これは、サブクエリに集計関数 AVG が含まれているためです。

```
SELECT Name, Description
FROM Products
WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems );
```

1.3.8.8.2 ANY、ALL、または SOME に続くサブクエリ

キーワード ALL、ANY、または SOME に続くサブクエリは、限定比較と呼ばれます。

オプティマイザは、次のような場合にこれらのサブクエリをジョインに変換します。

- メインクエリが GROUP BY 句を含んでおらず、集計クエリではありません。または、サブクエリが 1 つの値を返します。
- サブクエリが GROUP BY 句を含んでいません。
- サブクエリがキーワード DISTINCT を含んでいません。
- サブクエリが UNION クエリではありません。
- サブクエリが集計クエリではありません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 以下の部分が否定されていません。

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

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

最初の 4 つの条件は、比較的簡単です。

例

"Mrs. Clarke と Suresh がいつ注文し、どの担当者が注文を受けたか" という要求は、サブクエリの形式で処理できます。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID = ANY (
   SELECT ID
   FROM Customers
   WHERE Surname = 'Clarke' OR GivenName = 'Suresh' );
```

または、ジョインの形式で表現できます。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders, Customers
WHERE CustomerID=Customers.ID AND
  ( Surname = 'Clarke' OR GivenName = 'Suresh' );
```

しかし、「Mrs. Clarke、Suresh、および顧客でもある従業員が、注文したのはいつか」という要求は UNION クエリとして表現されるので、ジョインには変換できません。

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID = ANY (
   SELECT ID
   FROM Customers
   WHERE Surname = 'Clarke' OR GivenName = 'Suresh'
   UNION
   SELECT EmployeeID
   FROM Employees );
```

同様に、「すべての製品の最初の出荷日の後に出荷されていない注文の注文 ID と顧客 ID を検索する」という要求は、集計クエリで表現されるため、ジョインに変換できません。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE NOT OrderDate > ALL (
   SELECT FIRST ( ShipDate )
   FROM SalesOrderItems
   ORDER BY ShipDate );
```

ANY と ALL 演算子を使用するサブクエリの否定

5 つ目の条件はやや複雑です。次の形式のクエリがジョインに変換されます。

```
SELECT select-list
FROM table
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
WHERE NOT expression comparison-operator ALL( subquery-expression )
```

```
SELECT select-list
FROM table
WHERE expression comparison-operator ANY( subquery-expression )
```

ただし、次のクエリはジョインに変換されません。

```
SELECT select-list
FROM table
WHERE expression comparison-operator ALL( subquery-expression )
```

```
SELECT select-list
FROM table
WHERE NOT expression comparison-operator ANY( subquery-expression )
```

最初の 2 つのクエリも、後の 2 つのクエリも、それぞれ同等です。すでに説明したように、ANY 演算子は OR 演算子と似ていますが、引数の数が異なります。同様に、ALL 演算子は AND 演算子に似ています。たとえば、次の 2 つの式は同等です。

```
NOT ( ( X > A ) AND ( X > B ) )
( X <= A ) OR ( X <= B )
```

次の 2 つの式も同等です。

```
WHERE NOT OrderDate > ALL (
   SELECT FIRST ( ShipDate )
   FROM SalesOrderItems
   ORDER BY ShipDate )
```

```
WHERE OrderDate <= ANY (
   SELECT FIRST ( ShipDate )
   FROM SalesOrderItems
   ORDER BY ShipDate )
```

ANY と ALL の否定

一般に、次の 2 つの式は同等です。

```
NOT column-name operator ANY( subquery-expression )
```

```
column-name inverse-operator ALL( subquery-expression )
```

次の式も、一般に同等です。

```
NOT column-name operator ALL( subquery-expression )
```

```
column-name inverse-operator ANY( subquery-expression )
```

inverse-operator は、次の表に示すように、operator を否定することによって取得されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

operator	inverse-operator
=	<>
<	=>
>	=<
=<	>
=>	<
<>	=

1.3.8.8.3 IN に続くサブクエリ

一定の基準を満たす場合に、オプティマイザは、IN キーワードが続くサブクエリのみを変換します。

- メインクエリが GROUP BY 句を含んでおらず、集計クエリではありません。または、サブクエリが 1 つの値を返します。
- サブクエリが GROUP BY 句を含んでいません。
- サブクエリがキーワード DISTINCT を含んでいません。
- サブクエリが UNION クエリではありません。
- サブクエリが集計クエリではありません。
- 'expression IN (subquery-expression)' の部分が否定されていません。

例

"部長でもある従業員の名前を検索する" という要求は、次のクエリで表現されますが、この要求は条件を満たすため、ジョインされたクエリに変換されます。

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID IN (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName ='Finance' OR
          DepartmentName = 'Shipping' ) );
```

しかし、「部長か顧客のいずれかである従業員の名前を検索する」という要求は、UNION クエリで表現されているとジョインに変換されません。

IN 演算子に続く UNION クエリは変換できない

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID IN (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' )
   UNION
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    SELECT CustomerID
    FROM SalesOrders);
```

同様に、「部長ではない従業員の名前を検索する」という要求は、次に示す否定のサブクエリで表現されますが、変換されません。

```
SELECT GivenName, Surname
FROM Employees
  WHERE NOT EmployeeID IN (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );
```

IN サブクエリまたは ANY サブクエリがジョインに変換されるために必要な条件は、同じです。これは、2 つの式が論理的には同等であるためです。

ANY 演算子を使用するクエリに変換される、IN 演算子を使用するクエリ

データベースサーバは、IN 演算子を使用するクエリを ANY 演算子を使用するクエリに変換し、サブクエリをジョインに変換するかどうかを決定することがあります。たとえば、次の 2 つの式は同等です。

```
WHERE column-name IN( subquery-expression )
```

```
WHERE column-name = ANY( subquery-expression )
```

同様に、次の 2 つのクエリは同等です。

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID IN (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );
```

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID = ANY (
   SELECT DepartmentHeadID
   FROM Departments
   WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );
```

1.3.8.8.4 EXISTS に続くサブクエリ

一定の基準を満たす場合に、オプティマイザは、EXISTS キーワードに続くサブクエリを変換します。

- メインクエリが GROUP BY 句を含んでおらず、集計クエリではありません。または、サブクエリが 1 つの値を返します。
- 'EXISTS (subquery)' の部分が否定されていません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- サブクエリが相関です。つまり、外部参照を含んでいます。

例

"どの顧客が 2001 年 7 月 13 日以降に発注したか" という要求は、外部参照 Customers.ID = SalesOrders.CustomerID を含む否定されていないサブクエリを持つクエリで表現できるので、次のジョインで表すことができます。

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
   SELECT *
   FROM SalesOrders
   WHERE ( OrderDate > '2001-07-13' ) AND
         ( Customers.ID = SalesOrders.CustomerID ) );
```

EXISTS キーワードは、空の結果セットをチェックするようデータベースサーバに通知するものです。内部ジョインが使用されていると、データベースサーバは、FROM 句内のすべてのテーブルからのデータがあるローのみを自動的に表示します。つまり、次のクエリは、サブクエリを持つクエリが返すものと同じローを返します。

```
SELECT DISTINCT GivenName, Surname
FROM Customers, SalesOrders
WHERE ( SalesOrders.OrderDate > '2001-07-13' ) AND
      ( Customers.ID = SalesOrders.CustomerID );
```

1.3.9 データ操作文

データの追加、変更、削除に使用する文はデータ操作文と呼ばれ、ANSI SQL のデータ操作言語 (DML) 文部分のサブセットとなっています。

主要な DML 文を次に示します。

INSERT 文

テーブルまたはビューに新規のローを追加します。

UPDATE 文

一連のテーブルまたはビューのローを変更します。

DELETE 文

一連のテーブルまたはビューのローを削除します。

MERGE 文

テーブルまたはビューの特定のローを追加、変更、削除します。

前述の文に加えて、LOAD TABLE 文と TRUNCATE TABLE 文は、データのバルクロードや削除を行う場合に便利です。

このセクションの内容:

データ操作の権限 [499 ページ]
データ操作文は、修正するデータベーステーブルに対する適切な権限を持っている場合のみ実行できます。

トランザクションとデータ操作 [499 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データを修正すると、各データ操作文によって影響を受ける各ローの新旧の状態がコピーされて、ロールバックログに格納されます。

永続的なデータ変更 [500 ページ]
COMMIT 文は、まとまった意味を持つ文のグループの後で使用します。COMMIT 文は、データベースの変更を永続的なものにします。

変更のキャンセル [500 ページ]
コミットされていない変更はすべてキャンセルできます。

トランザクションとデータリカバリ [500 ページ]
システム障害や停電が発生時に、データベースの整合性が保護されます。

INSERT を使用したデータの追加 [501 ページ]
INSERT 文を使用してデータベースにローを追加することができます。

UPDATE によるデータの変更 [508 ページ]
UPDATE 文では、変更するローと、それらのローで特定のカラムの新しい値として使用する式を指定します。

INSERT によるデータの変更 [512 ページ]
INSERT 文の ON EXISTING 句を使用して、テーブル内の既存のローを (プライマリキールックアップに基づいて) 新しい値で更新できます。

DELETE によるデータの削除 [513 ページ]
DELETE 文を使用して、データをデータベースから永続的に削除することができます。

1.3.9.1 データ操作の権限

データ操作文は、修正するデータベーステーブルに対する適切な権限を持っている場合のみ実行できます。

データベースの管理者とデータベースオブジェクトの所有者は、GRANT 文と REVOKE 文を使用して、だれがどのデータ操作機能にアクセスできるのかを決定します。

権限は個々のユーザ、ロール、ユーザ拡張ロールに付与できます。

1.3.9.2 トランザクションとデータ操作

データを修正すると、各データ操作文によって影響を受ける各ローの新旧の状態がコピーされて、ロールバックログに格納されます。

トランザクションを開始し、間違いに気づいてトランザクションをロールバックすると、データベースを前の状態に回復できます。

関連情報

トランザクションと独立性レベル [757 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.9.3 永続的なデータ変更

COMMIT 文は、まとまった意味を持つ文のグループの後で使用します。COMMIT 文は、データベースの変更を永続的なものにします。

たとえば、ある顧客の口座から別の顧客の口座に金銭を振り込む場合、振り込み前と後の合計金額が同一である必要があるため、振り込まれる側の口座にその金額を加え、その後で振り込む側の口座からその金額を削除して、最後にコミットします。

auto_commit オプションを On に設定すると、Interactive SQL に対して変更を自動的にコミットするように指定できます。これは Interactive SQL のオプションです。auto_commit を On に設定すると、INSERT 文、UPDATE 文、DELTE 文を実行するたびに Interactive SQL が COMMIT 文を発行します。このため、パフォーマンスが大幅に低下することがあります。このような場合には、auto_commit オプションを Off に設定することをお奨めします。

注記

このチュートリアルにある例を試してみる場合、データベースの変更を確定しても問題がないと確信するまでは、どのような変更に対してもコミットしないように注意してください。

1.3.9.4 変更のキャンセル

コミットされていない変更はすべてキャンセルできます。

SQL では、ROLLBACK 文を使用することによって最後のコミット以降に加えた変更をすべて取り消せます。この文は、最後に変更を確定した後にデータベースに対して行われたすべての変更を取り消します。

1.3.9.5 トランザクションとデータリカバリ

システム障害や停電が発生時に、データベースの整合性が保護されます。

データベースサーバをリストアするためのオプションが複数用意されています。たとえば、データベースサーバによって別のドライブに保存されたログファイルを使用して、データをリストアできます。リカバリのためにトランザクションログファイルを使用する場合、データベースサーバはデータベースを頻繁に更新する必要がないため、データベースサーバのパフォーマンスは向上します。

トランザクション処理により、データベースサーバはデータが一貫性を保っていることを識別できます。トランザクション処理は、なんらかの理由でトランザクションが正常に完了しなかった場合に、トランザクション全体が取り消されるか、ロールバックしたことを確認します。トランザクションが失敗しても、データベースには影響ありません。

SQL Anywhere のトランザクション処理は、トランザクションの途中でシステムがダウンした場合でも、トランザクションの内容は確実に処理されることを保証します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.9.6 INSERT を使用したデータの追加

INSERT 文を使用してデータベースにローを追加することができます。

INSERT 文には 2 つの形式があります。VALUES キーワードまたは SELECT 文を使用できます。

値を使う INSERT

VALUES キーワードで新しいロー内の一部、またはすべてのカラムの値を指定します。VALUES キーワードを使った INSERT 文の簡略バージョンの構文は、次のとおりです。

```
INSERT [ INTO ] table-name [ ( column-name, ... ) ]
VALUES ( expression, ... )
```

SELECT * によるクエリを実行した結果に表示される順で、テーブルの各カラムに値を入力すると、カラム名のリストを省略できます。

SELECT からの INSERT

INSERT 文に SELECT 文を使用して、1 つ以上のテーブルから値を引き出せます。データを挿入するテーブルに多数のカラムがある場合は、WITH AUTO NAME を使用して構文を簡単にすることもできます。WITH AUTO NAME を使用する場合、カラム名を指定する必要があるのは、INSERT 文と SELECT 文の両方ではなく、SELECT 文のみです。SELECT 文の名前には、カラム参照かエイリアスの式を指定してください。

SELECT 文を使用した INSERT 文の簡略バージョンの構文は、次のとおりです。

```
INSERT [ INTO ] table-name
[ WITH AUTO NAME ] select-statement
```

このセクションの内容:

ローの全カラムへの値の挿入 [502 ページ]
INSERT 文を使用して、ローの全カラムに値を挿入します。

指定カラムへの値の挿入 [503 ページ]
値は INSERT 文で指定された内容に従ってカラムに挿入されます。

SELECT を使用した新しいローの追加 [505 ページ]
1 つ以上のテーブルから他のテーブルへ値を引き出すには、INSERT 文に SELECT 句を使用します。

ドキュメントとイメージの挿入 [506 ページ]
ドキュメントまたはイメージをデータベースに格納するには、ファイルの内容を変数に読み込んで、その変数を INSERT 文の値として指定するアプリケーションを記述します。

高度: 挿入されたローに対するディスク割り付け [507 ページ]
挿入されたローに対するディスク割り付けが連続しているかどうか、または、任意の順序でローを挿入できるかどうかを制御することができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.9.6.1 ローの全カラムへの値の挿入

INSERT 文を使用して、ローの全カラムに値を挿入します。

前提条件

テーブルの INSERT オブジェクトレベル権限が必要です。ON EXISTING UPDATE 句が指定された場合は、そのテーブルに対する UPDATE 権限も必要になります。

元の CREATE TABLE 文にあるカラム名と同じ順序で値を入力します。

値をカッコで囲みます。

すべての文字データを一重引用符で囲みます。

追加する各ローには、別の INSERT 文を使用します。

手順

各カラムの値を含む INSERT 文を実行します。

結果

指定された値は、新規ローの各カラムに挿入されます。

例

次の INSERT 文は、Departments テーブルに新規のローを追加して、そのローのすべてのカラムに値を指定します。

```
INSERT INTO GROUPO.Departments
VALUES ( 702, 'Eastern Sales', 902 );
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.9.6.2 指定カラムへの値の挿入

値は INSERT 文で指定された内容に従ってカラムに挿入されます。

指定されたカラムと指定されていないカラムに挿入される値

値は INSERT 文で指定された内容に従ってローに挿入されます。カラムに値が入力されていない場合、挿入される値はカラム設定 (NULL 値やデフォルト値を挿入するなど) によって異なります。挿入操作が失敗して、エラーが返される場合もあります。次の表は、挿入される値 (該当する場合) とカラム設定に基づいた結果を示しています。

挿入される値	NULL 入力可	NULL 入力不可	NULL 入力可 (DEFAULT 指定)	NULL 入力不可 (DEFAULT 指定)	NULL 入力不可 (DEFAULT AUTOINCREMENT または DEFAULT [UTC] TIMESTAMP 指定)
<なし>	NULL	SQL エラー	デフォルト値	デフォルト値	デフォルト値
NULL	NULL	SQL エラー	NULL	SQL エラー	デフォルト値
指定された値	指定された値	指定された値	指定された値	指定された値	指定された値

デフォルトでは、テーブルの作成時にカラム定義で NOT NULL と明示的に記述しないかぎり、カラムに NULL を使用できます。allow_nulls_by_default オプションを使用して、デフォルトを変更できます。また、ALTER TABLE 文を使用して、特定のカラムに NULL 値を許可するかどうかを変更できます。

制約を使用したカラムデータの制限

カラムまたはドメインに対する制約を作成できます。制約によって、追加可能または追加不可能なデータの種類が決定されます。

NULL の明示的挿入

NULL を入力すると、カラムに NULL を明示的に挿入できます。NULL を引用符で囲まないでください。囲むと文字列として扱われます。たとえば、次の文は DepartmentHeadID カラムに NULL を明示的に挿入します。

```
INSERT INTO Departments
VALUES ( 703, 'Western Sales', NULL );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトを使用した値の指定

カラムが値を受け取らなくても、ローを挿入したら常にデフォルト値が自動的に挿入されるように、カラムを定義できます。これを設定するには、カラムにデフォルト値を設定します。

このセクションの内容:

[指定カラムへの値の挿入 [504 ページ]]
カラムとその値のみを指定して、ローにあるカラムにデータを追加します。

関連情報

テーブルとカラム制約 [738 ページ]
カラムデフォルト [730 ページ]

1.3.9.6.2.1 指定カラムへの値の挿入

カラムとその値のみを指定して、ローにあるカラムにデータを追加します。

前提条件

テーブルの INSERT オブジェクトレベル権限が必要です。ON EXISTING UPDATE 句が指定された場合は、そのテーブルに対する UPDATE 権限も必要になります。

コンテキスト

指定するカラムの順序はテーブル内のカラムの順序と一致させる必要はなく、挿入する値を指定する順序と一致させてください。

そのカラムリストにない他のすべてのカラムは、NULL 入力可、またはデフォルト値を持つように定義します。デフォルト値の入ったカラムを省略すると、そのカラムにはデフォルトが挿入されます。

手順

INSERT INTO 文を実行して、指定カラムにデータを追加します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば次の文は、DepartmentID と DepartmentName の 2 つのカラムだけにデータを追加します。

```
INSERT INTO GROUPO.Departments ( DepartmentID, DepartmentName )
VALUES ( 703, 'Western Sales' );
```

DepartmentHeadID にはデフォルトの値はありませんが、NULL. は受け入れます。そのため、NULL は自動的にそのカラムに割り当てられます。

結果

データが、指定されたカラムに挿入されます。

1.3.9.6.3 SELECT を使用した新しいローの追加

1 つ以上のテーブルから他のテーブルへ値を引き出すには、INSERT 文に SELECT 句を使用します。

SELECT 句により、ローにあるカラムの一部またはすべてに値を挿入できます。

一部のカラムだけに対する値の挿入は、既存のテーブルから値を取得する場合に便利です。その場合、UPDATE 文を使用して他のカラムの値を追加できます。

テーブルにある一部のカラムにのみ値を挿入するには、その前にデフォルト値が存在するか、または値を挿入しないカラムに NULL を指定しているか確認します。こうしないと、エラーが表示されます。

あるテーブルから他のテーブルにローを挿入する場合、2 つのテーブルは互換性のある構造にします。すなわち、一致するカラムを、同じデータ型またはデータベースサーバが自動的に変換できるデータ型にします。

一部のカラムへのデータ挿入

SELECT 文を使用して、VALUES 句を使用する場合と同様に、ローにある一部のカラムにのみデータを追加できます。INSERT 句でデータを追加するカラムを指定するだけです。

同じテーブルからのデータの挿入

特定のテーブルに対して、同じテーブルにある他のデータに基づくデータを挿入できます。本質的には、これはローの全部または一部をコピーすることを意味します。

たとえば、Products テーブルに既存の製品に基づく新しい製品を挿入できます。次の文は Products テーブルに新しい項目の Extra Large Tee Shirt (Tank Top、V-neck、Crew Neck の各種) を追加します。ID 番号は既存サイズのシャツ番号に 30 を加えます。

```
INSERT INTO Products
SELECT ID + 30, Name, Description,
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    'Extra large', Color, 50, UnitPrice, NULL
FROM Products
WHERE Name = 'Tee Shirt';
```

例

2 つのテーブルでカラムの順序が同じである場合、どちらのテーブルのカラム名も指定する必要はありません。たとえば、NewProducts というテーブルのスキーマが Products テーブルのスキーマと同じであり、NewProducts テーブルには Products テーブルに追加する製品情報の一部のローが含まれているとします。この場合、次の文を実行できます。

```
INSERT Products
SELECT *
FROM NewProducts;
```

1.3.9.6.4 ドキュメントとイメージの挿入

ドキュメントまたはイメージをデータベースに格納するには、ファイルの内容を変数に読み込んで、その変数を INSERT 文の値として指定するアプリケーションを記述します。

テーブルへのファイル内容の挿入には、xp_read_file システムプロシージャも使用できます。ファイルの内容を Interactive SQL から挿入する場合や、完全なプログラミング言語を提供しない他の環境から挿入する場合に、このプロシージャを使用すると便利です。

例

この例では、テーブルを作成してテーブルのカラムにイメージを挿入します。これらの手順は Interactive SQL から実行します。

1. イメージを保持するテーブルを作成します。

```
CREATE TABLE Pictures
( C1 INT DEFAULT AUTOINCREMENT PRIMARY KEY,
   Filename VARCHAR(254),
   Picture LONG BINARY );
```

2. データベースサーバの現在の作業ディレクトリにある `portrait.gif` の内容をテーブルに挿入します。

```
INSERT INTO Pictures ( Filename, Picture )
VALUES ( 'portrait.gif',
   xp_read_file( 'portrait.gif' ) );
```

関連情報

OPENXML 演算子を使用した XML のインポート [554 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.3.9.6.5 高度: 挿入されたローに対するディスク割り付け

挿入されたローに対するディスク割り付けが連続しているかどうか、または、任意の順序でローを挿入できるかどうかを制御することができます。

SQL Anywhere では、可能な場合ローを連続して格納する

新しいローは、データベースファイルのページサイズよりも小さい場合、常に単一のページに保管されます。現在のページに新しいローを保存する十分な空き領域がない場合、データベースサーバはローを新しいページに書き込みます。たとえば、新しいローが 600 バイトの領域を必要とするときに、ページの一部が埋まっていて 500 バイトしか使用できない場合、データベースサーバは新しいページにローを配置します。

ディスク上のテーブルページがさらに連続するように、データベースサーバはテーブルページを 8 ページのブロック単位で割り付けます。たとえば、1 ページの割り付けが必要な場合は、8 ページを割り付け、必要な 1 ページをブロックに挿入してから、ブロックの残りの 7 ページを埋めます。また、空きページビットマップを使用して、DB 領域内で連続するページブロックを検索します。次に、64 KB のグループを読み込み、ビットマップを使用して関連ページを検索し、逐次スキャンを実行します。このため、逐次スキャンの効率が高まります。

SQL Anywhere ではローをどのような順序でも保存できる

データベースサーバはページの領域を検索し、受け取った順序でローを挿入します。それぞれのローを 1 ページに割り当てますが、テーブル内で選択したロケーションは、ローが挿入された順序と一致しない場合があります。たとえば、データベースサーバは、長いローを隣接して保管するためにページを新しくする必要があることがあります。次のローが短い場合、そのローは前のページの空いているロケーションに配置されます。

すべてのテーブルのローには順序が付いていません。ローを受け取ったり処理したりする順序が重要である場合、SELECT 文で ORDER BY 句を使用し、結果に順序を付けます。テーブル内のローの順序に依存するアプリケーションは、警告なしに失敗することがあります。

テーブルのローを特定の順序にすることが頻繁に必要になる場合は、クエリの ORDER BY 句で指定したカラムにインデックスを作成することを検討してください。

NULL カラム用の領域は予約されない

デフォルトでは、データベースサーバは、ローを挿入する場合は必ず、ローを作成時の値で格納するために必要な領域だけを予約します。NULL 値、またはテキスト文字列などの拡張する可能性のあるフィールドを格納するための追加領域は予約しません。

データベースサーバに対して、テーブルの作成時に PCTFREE オプションを使用して領域を予約するよう強制できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

一度挿入されたローの識別子は不変

一度ページ上にホーム位置が割り当てられると、ローは決してそのページから移動しません。更新によりローのいずれかの値が変更され、割り当てられているページに適合しなくなると、ローは分割され、追加情報が別のページに挿入されます。

この特性には注意が必要です。特に、ローの挿入時にデータベースサーバは追加領域を許可しないためです。たとえば、大量の空のローを 1 つのテーブルに挿入して、UPDATE 文を使用して一度に 1 カラムずつ値を入力するとします。この結果、1 つのローにあるほとんどすべての値は別々のページに保存されます。1 つのローからすべての値を取り出すために、データベースサーバは複数のディスクページを読み込まなければならない場合があります。この簡単な操作にかなりの時間がかかることになります。

新しいローへのデータ配置を挿入時に行うことを検討してください。ローは一度挿入されれば、データを保持するのに十分な領域を確保します。

データベースファイルは縮小しない

データベースにローを挿入してから削除すると、データベースサーバはローが使用していた領域を自動的に再利用します。したがって、データベースサーバは別のローが以前に使用していた領域に新しいローを挿入します。

データベースサーバは、各ページの空き領域の量の記録を保持しています。新しいローを挿入するよう要求されると、まず既存のページの領域の記録を検索します。既存のページで十分な領域を見つけると、新しいローをそのページに配置し、必要であればそのページの内容を再編成します。見つけられない場合には、新しいページを開始します。

いくつかのローを削除し、空き領域を使用できる程度の小さなローを新しく挿入しなかった場合、時間の経過とともにデータベース内の情報がまばらになることがあります。その場合は、テーブルを再ロードするか、REORGANIZE TABLE 文を使用してテーブルの断片化を解除できます。

1.3.9.7 UPDATE によるデータの変更

UPDATE 文では、変更するローと、それらのローで特定のカラムの新しい値として使用する式を指定します。

UPDATE 文を使用して、テーブルにある単一のロー、ローのグループ、またはすべてのローを変更できます。他のデータ操作文 (INSERT、MERGE、DELETE) と異なり、UPDATE 文では複数のテーブル内のローを同時に修正することもできます。すべての場合で、UPDATE 文の実行はアトミックです。すべてのローがエラーなく修正されるか、すべて修正されません。たとえば、修正される値の 1 つが誤ったデータ型であったり、新しい値によって CHECK 制約違反が発生したりした場合、UPDATE は失敗し、操作全体がロールバックされます。

UPDATE 構文

UPDATE 文の構文の簡略バージョンは次のとおりです。

```
UPDATE table-name
SET column_name = expression
WHERE search-condition
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

会社 Newton Ent.(SQL Anywhere サンプルデータベースの Customers テーブル内の会社) が Einstein, Inc. に吸収される場合は、次のような文を使用して会社名を更新できます。

```
UPDATE Customers
SET CompanyName = 'Einstein, Inc.'
WHERE CompanyName = 'Newton Ent.';
```

WHERE 句で任意の式を使用できます。入力された会社名のスペルがわからなければ、次のような文を使用して Newton という会社名を更新してみます。

```
UPDATE Customers
SET CompanyName = 'Einstein, Inc.'
WHERE CompanyName LIKE 'Newton%';
```

探索条件は更新されるカラムを参照する必要はありません。Newton Entertainments の会社 ID は 109 です。ID 値はテーブルのプライマリキーなので、次の文を使用して正しいローを確実に更新できます。

```
UPDATE Customers
SET CompanyName = 'Einstein, Inc.'
WHERE ID = 109;
```

➡ ヒント

また、Interactive SQL で結果セットからのローを修正することもできます。

SET 句

SET 句は、更新されるカラムとその新しい値を指定します。WHERE 句は、更新する必要があるローを決定します。WHERE 句がない場合、指定されたすべてのローのカラムが SET 句の値によって更新されます。

SET 句で指定できる式は、定数リテラル、ホストまたは SQL 変数、サブクエリ、CURRENT TIMESTAMP などの特別値、別のテーブルから引き出された式の値、またはこれらの組み合わせです。また、SET 句で DEFAULT を指定すると、そのベーステーブルのカラムのデフォルト値を指定できます。式のデータ型が修正するカラムのデータ型と異なる場合は、可能であれば、データベースサーバによって式はカラムの型に自動的に変換されます。変換できない場合は、データ例外が発生し、UPDATE 文は失敗します。

SET 句は、カラム値の修正以外に、変数の値を設定するために使用できます。次の例は、テーブル T の更新以外に、変数 @var に値を代入します。

```
UPDATE T
SET @var = expression1, col1 = expression2
WHERE...;
```

これは、SELECT 文とそれに続く UPDATE を連続して実行することとほぼ同じです。

```
SELECT @var = expression1
FROM T
WHERE... ;
UPDATE T SET col1 = expression2
WHERE...;
```

UPDATE 文での変数の代入の利点は、書き込みロックを保持したまま、変数の値を文の実行の中で設定できる点です。これにより、他の接続から同時に実行される更新アクティビティによる予期しない値の代入を防ぎます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

WHERE 句

WHERE 句は、UPDATE 文で指定したテーブルまたはテーブル式の直積に search-condition を適用することによって、更新されるローを指定します。たとえば、次の文は "One Size Fits All" を "Extra Large Tee Shirt" に書き換えます。

```
UPDATE Products
SET Size  = 'Extra Large'
WHERE Name = 'Tee Shirt'
   AND Size = 'One Size Fits All';
```

複雑な UPDATE 文

より複雑な形式の UPDATE 文では、ジョインによる更新やその他のタイプのテーブル式が可能です。

UPDATE 文の次の構文を考えてみます。

```
UPDATE [ row-limitation ] table-name
SET set-item[, ...]
FROM table-expression [, ...] ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ] , ...]
[ OPTION( query-hint, ... ) ]
```

この形式の UPDATE 文のセマンティックは、最初に各 table-expression からのローのすべての組み合わせで構成される結果セットを計算し、次に WHERE 句の search-condition を適用してから、結果ローを ORDER BY 句を使用して順序付けします。この計算の結果は、修正されるローのセットになります。各 table-expression は、ベーステーブル、ビュー、派生テーブルのジョインで構成できます。この構文では、他のテーブルにあるカラムの値を持つ 1 つまたは複数のテーブルを更新できます。クエリオプティマイザによって操作が並べ替えられ、UPDATE 文のより効率的な実行方式が作成される場合があります。

ベーステーブルのローが、修正されるローのセット内に複数回現れる場合、ローの新しい値が各操作の試行で異なっていれば、そのローは複数回更新されます。BEFORE ROW UPDATE トリガが存在する場合は、トリガの UPDATE OF column-list 句に応じて、個々のロー操作のたびに BEFORE ROW UPDATE トリガが起動されます。AFTER ROW UPDATE トリガも、トリガの UPDATE OF column-list 句に応じて各ロー操作とともに起動されますが、ローの値が実際に変更される場合のみです。

トリガは、トリガのタイプと各トリガ定義の ORDER 句の値に基づいて、更新されるテーブルごとに起動されます。ただし、UPDATE 文によって複数のテーブルが修正される場合、テーブルが更新される順序は保証されません。

次の例では、Products テーブルに BEFORE ROW UPDATE トリガおよび AFTER STATEMENT UPDATE トリガを作成します。各トリガによって、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出力されます。

```
CREATE OR REPLACE TRIGGER trigger0
BEFORE UPDATE
ON Products
REFERENCING OLD AS old_product NEW AS new_product
FOR EACH ROW
BEGIN
   PRINT ('BEFORE row: PK value: ' || old_product.ID || ' New Price: ' ||
new_product.UnitPrice );
END;
CREATE OR REPLACE TRIGGER trigger1
AFTER UPDATE
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ON Products
REFERENCING NEW AS new_product
FOR EACH STATEMENT
BEGIN
  DECLARE @pk INTEGER;
  DECLARE @newUnitPrice DECIMAL(12,2);
  DECLARE @err_notfound EXCEPTION FOR SQLSTATE VALUE '02000';
  DECLARE new_curs CURSOR FOR
   SELECT ID, UnitPrice FROM new_product;
  OPEN new_curs;
  LoopGetRow:
    LOOP
      FETCH NEXT new_curs INTO @pk, @newUnitPrice;
      IF SQLSTATE = @err_notfound THEN
         LEAVE LoopGetRow
      END IF;
      PRINT ('AFTER stmt: PK value: ' || @pk || ' Unit price: ' || @newUnitPrice );
  END LOOP LoopGetRow;
  CLOSE new_curs
END;
```

次に、Products テーブルと SalesOrderItems テーブルのジョインに UPDATE 文を実行し、2001 年 4 月 1 日以降に出荷された、大きい注文が少なくとも 1 つある製品を 5% 割引するとします。

```
UPDATE Products p JOIN SalesOrderItems s ON (p.ID = s.ProductID)
SET p.UnitPrice = p.UnitPrice * 0.95
WHERE s.ShipDate > '2001-04-01' AND s.Quantity >= 72;
```

データベースサーバメッセージウィンドウには、次の情報が表示されます。

```
BEFORE row: PK value: 700 New Price: 14.25
BEFORE row: PK value: 302 New Price: 13.30
BEFORE row: PK value: 700 New Price: 13.54
AFTER stmt: PK value: 700 Unit price: 14.25
AFTER stmt: PK value: 302 Unit price: 13.30
AFTER stmt: PK value: 700 Unit price: 13.54
```

メッセージは、Product 700 が 2 回更新されたことを示しています。これは、Product 700 が UPDATE 文の検索条件と一致する 2 つの異なる注文に含まれていたためです。更新の重複は、BEFORE ROW トリガと AFTER STATEMENT トリガの両方からわかります。各ロー操作により、各トリガ呼び出しの **OLD** 値および **NEW** 値はそれに応じて変更されます。AFTER STATEMENT トリガでは、REFERENCING 句によって形成されるテンポラリテーブル内のローの順序は、ローが修正された順序と一致しない場合があり、それらのローの正確な順序は保証されません。

更新の重複のために、Product 700 の UnitPrice は 2 回割引され、最初の $15.00 から、$14.25 ではなく $13.54 (9.75% の割引が発生) に下がりました。この意図しない結果を回避するには、ジョインではなく EXISTS サブクエリを使用して、各 Product ローの修正が多くても 1 回になることを保証するように UPDATE 文を作成します。書き換えられた UPDATE 文では、EXISTS サブクエリと、FROM 句を許可する代替 UPDATE 文の構文の両方が使用されます。

```
UPDATE Products AS p
SET p.UnitPrice = p.UnitPrice * 0.95
FROM Products AS p
WHERE EXISTS(
    SELECT *
    FROM SalesOrderItems s
    WHERE p.ID = s.ProductID
       AND s.ShipDate > '2001-04-01'
       AND s.Quantity >= 72);
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

UPDATE と制約違反

実行中に UPDATE 文が参照整合性制約に違反した場合、文の動作は wait_for_commit オプションの設定によって制御されます。wait_for_commit オプションが Off に設定されていて、参照制約違反が発生した場合、UPDATE 文の結果は直ちに自動的にロールバックされ、エラーメッセージが表示されます。wait_for_commit オプションが On に設定されている場合、UPDATE 文によって発生した参照整合性制約違反は一時的に無視され、接続によって COMMIT が実行されるときにチェックされます。

修正されるベーステーブルに、プライマリキー、UNIQUE 制約、またはユニークインデックスがある場合、UPDATE 文のローごとの実行によって一意性制約違反が発生する場合があります。たとえば、テーブル T のプライマリキーカラムの値すべてを増分する次のような UPDATE 文を発行する場合があります。

```
UPDATE T SET PKcol = PKcol + 1;
```

UPDATE 文の実行中に一意性違反が発生した場合、データベースサーバによって次の処理が自動的に行われます。

1. 修正されるローの新しい値と古い値が、修正されるベーステーブルと同じスキーマを持つテンポラリテーブルにコピーされます。
2. 元のローがベーステーブルから削除されます。この削除操作の結果として DELETE トリガは起動されません。

UPDATE 文の実行中に、正常に更新されるローと一時的に削除されるローは、評価の順序によって異なり、保証されません。弱い独立性レベル (独立性レベル 0、1、または 2) で実行されている他の接続からの SQL 要求の動作は、これらの一時的に削除されるローの影響を受ける場合があります。修正されるテーブルの BEFORE または AFTER ROW トリガには、トリガの REFERENCING 句に従って各ローの古い値と新しい値が渡されますが、修正されるテーブルに対して ROW トリガによって個別の SQL 文が発行される場合、テンポラリテーブルに保持されているローは失われます。

UPDATE 文による各ローの修正が完了した後、テンポラリテーブルに保持されているローはベーステーブルに戻されます。一意性違反がまだ発生する場合は、UPDATE 文全体がロールバックされます。テンポラリテーブルに保持されているすべてのローがベーステーブルに正常に再挿入された場合にのみ、AFTER STATEMENT トリガが起動されます。

修正されるベーステーブルが ON DELETE CASCADE、ON DELETE SET NULL、ON DELETE DEFAULT、ON UPDATE CASCADE、ON UPDATE SET NULL、ON UPDATE DEFAULT などの参照整合性制約アクションのターゲットである場合、ローを一時的に格納するためにデータベースサーバによって保持テーブルは使用されません。

関連情報

DELETE 文または UPDATE 文の整合性検査 [754 ページ]
更新時のロック [795 ページ]

1.3.9.8 INSERT によるデータの変更

INSERT 文の ON EXISTING 句を使用して、テーブル内の既存のローを (プライマリキールックアップに基づいて) 新しい値で更新できます。

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、またはプロキシテーブルでこの句を使用すると、構文エラーになります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ON EXISTING 句を指定すると、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。対応するローが存在しない場合は、新しいローが挿入されます。すでにテーブルに存在するローに対しては、次の操作を選択できます。

- 重複するキー値に対してエラーを生成します。ON EXISTING 句を指定しない場合は、これがデフォルトの動作です。
- 入力ローを無視して、エラーを生成しません。
- 既存のローを入力ロー内の値で更新します。

1.3.9.9 DELETE によるデータの削除

DELETE 文を使用して、データをデータベースから永続的に削除することができます。

単純な DELETE 文は、次のような形式になっています。

```
DELETE [ FROM ] table-name
WHERE column-name = expression
```

次のような、より複雑な形式も使用できます。

```
DELETE [ FROM ] table-name
FROM table-list
WHERE search-condition
```

WHERE 句

WHERE 句を使用して削除するローを指定します。WHERE 句がないと、DELETE 文によってテーブルのすべてのローが削除されます。

FROM 句

DELETE 文の 2 番目に位置する FROM 句には、テーブルからデータを選択し、最初に指定されたテーブルから一致するデータを削除する、特別な働きがあります。FROM 句で選択したローに、削除の条件を指定します。

例

この例では、SQL Anywhere のサンプルデータベースを使用します。この文を実行するには、wait_for_commit オプションを On に設定してください。次の文は現在の接続に関してのみ、この操作を実行します。

```
SET TEMPORARY OPTION wait_for_commit = 'On';
```

これによって、外部キーに参照されるプライマリキーが含まれるローでも削除できますが、対応する外部キーも削除しないかぎり、COMMIT は許可されません。

次のビューでは、製品と販売した製品の値を表示します。

```
CREATE VIEW ProductPopularity as
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
SELECT  Products.ID,
   SUM( Products.UnitPrice * SalesOrderItems.Quantity )
   AS "Value Sold"
FROM  Products JOIN SalesOrderItems
ON Products.ID = SalesOrderItems.ProductID
GROUP BY Products.ID;
```

このビューを使用して、売り上げが 20,000 ドル未満の製品を Products テーブルから削除できます。

```
DELETE
FROM Products
FROM Products NATURAL JOIN ProductPopularity
WHERE "Value Sold" < 20000;
```

次の ROLLBACK 文を実行して、データベースのこれらの変更をキャンセルしてください。

```
ROLLBACK;
```

➜ ヒント

また、InteractiveSQL の結果セットから、データベーステーブルのローを削除することもできます。

このセクションの内容:

テーブルからのすべてのローの削除 [514 ページ]
TRUNCATE TABLE 文を使用すると、テーブルにあるすべてのローを簡単に削除できます。

1.3.9.9.1 テーブルからのすべてのローの削除

TRUNCATE TABLE 文を使用すると、テーブルにあるすべてのローを簡単に削除できます。

この方法は、条件を指定しない DELETE 文よりもすばやく処理できます。これは、DELETE 文が各変更のログを取るのに対し、TRUNCATE 文では個々のローの削除が記録されないためです。

DROP TABLE 文を実行しないかぎり、TRUNCATE TABLE 文によって空になったテーブルの定義は、インデックスや他の関連オブジェクトとともにデータベースに残されます。

他のテーブルに参照整合性制約で参照するローがあると、TRUNCATE TABLE 文を使用できません。外部テーブルからローを削除するか、外部テーブルをトランケートしてからプライマリテーブルをトランケートします。

ベーステーブルのトランケート操作またはバルクロードの実行操作を行うことにより、インデックス (通常のインデックスまたはテキストインデックス) 内と従属マテリアライズドビュー内のデータが古くなります。最初にインデックスや従属したマテリアライズドビューのデータをトランケートし、INPUT 文を実行してから、インデックスとマテリアライズドビューを再構築または再表示してください。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

TRUNCATE TABLE 構文

TRUNCATE TABLE 文の構文は次のとおりです。

```
TRUNCATE TABLE table-name
```

たとえば、SalesOrders テーブルのすべてのデータを削除するには、次のように入力します。

```
TRUNCATE TABLE SalesOrders;
```

TRUNCATE TABLE 文はテーブルで定義されたトリガを呼び出しません。

次の ROLLBACK 文を実行して、データベースのこれらの変更をキャンセルしてください。

```
ROLLBACK;
```

1.4 SQL のダイアレクトと互換性

準拠に関する情報については、ソフトウェアの各機能のリファレンスマニュアルを参照してください。

SQL Anywhere は、SQL-92 を中心とする米国連邦情報処理規格刊行物 (FIPS PUB) 127 に準拠しています。若干の例外がありますが、SQL Anywhere は、ISO/IEC JTC 1/SC 32 9075-2008 の 第 9 部に記載されているように、ISO/ANSI SQL/2008 の中核となる仕様にも準拠しています。SQL Anywhere。

このセクションの内容:

SQL Flagger を使用した SQL 準拠のテスト [516 ページ]
データベースサーバと SQL プリプロセッサ (sqlpp) によって、ベンダー拡張であるか、特定の ISO/ANSI SQL 標準に準拠していないか、Ultra Light でサポートされていない SQL 文を特定できます。

他の SQL ソフトウェアとは異なる機能 [518 ページ]
他の SQL ソフトウェアとは異なる SQL の機能がいくつかあります。

Watcom SQL [522 ページ]
SQL Anywhere がサポートしている SQL ダイアレクトは Watcom SQL と呼ばれています。

Transact-SQL との互換性 [523 ページ]
SQL Anywhere は、SAP Adaptive Server Enterprise がサポートする SQL ダイアレクトである Transact-SQL の大部分のサブセットをサポートしています。

SQL Anywhere と Adaptive Server Enterprise の比較 [525 ページ]
Adaptive Server Enterprise と SQL Anywhere はそれぞれの異なる目的に合わせたアーキテクチャを持つ、相互に補完的な製品です。

Transact-SQL と互換性のあるデータベース [530 ページ]
データベースを作成するとき、または既存のデータベースを再構築するときに適切なオプションを選択すると、SQL Anywhere と Adaptive Server Enterprise の相違点のいくつかを解除できます。

Transact-SQL 互換性のある SQL 文 [538 ページ]
Transact-SQL で動作する SQL 文を記述する際に、いくつか考慮すべき事項があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Transact-SQL のプロシージャ言語 [544 ページ]
SQL Anywhere は、ISO/ANSI SQL 標準に基づく Watcom SQL ダイアレクトに加えて、Transact-SQL ストアドプロシージャ言語の大部分をサポートします。

ストアドプロシージャの自動変換 [546 ページ]
SQL Anywhere は、Watcom SQL と Transact-SQL の間で文を変換する手助けをします。

Transact-SQL プロシージャから返される結果セット [547 ページ]
SQL Anywhere 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 プロシージャの中の変数 [548 ページ]
SQL Anywhere は SET 文を使用して、プロシージャ内の変数に値を割り当てます。

Transact-SQL プロシージャでのエラー処理 [549 ページ]
デフォルトでのプロシージャのエラー処理は、Watcom SQL と Transact-SQL とでは違います。

1.4.1 SQL Flagger を使用した SQL 準拠のテスト

データベースサーバと SQL プリプロセッサ (sqlpp) によって、ベンダー拡張であるか、特定の ISO/ANSI SQL 標準に準拠していないか、Ultra Light でサポートされていない SQL 文を特定できます。

この機能は、SQL Flagger と呼ばれます。ISO/ANSI 9075-1999 SQL 標準のオプションの SQL 言語機能 F812 として最初に導入されました。SQL Flagger によって、アプリケーション開発者は SQL 言語の特定のサブセットに違反する SQL 言語要素を特定できます。SQL Flagger を使用すると、SQL 標準のコア機能への準拠、またはコア機能とオプション機能の組み合わせへの準拠も確認できます。また、SQL Flagger は、SQL Anywhere で Ultra Light アプリケーションのプロトタイプを作成するときに、使用している SQL が Ultra Light でサポートされていることを確認するためにも使用できます。

空間データサポートは SQL/MM 標準の第 3 部 (ISO/IEC 13249-3) として標準化されているため、標準にない場合は、空間機能、操作、構文は SQL Flagger ではサポートされません。

SQL Flagger では、SQL 文の構文とセマンティックの両要素が分析の対象ですが、準拠はコンパイル時に静的にチェックされます。構文の準拠のテスト例として、INSERT 文にオプションの INTO キーワードがない場合を考えます (たとえば、`INSERT Products VALUES( ... )`)。これは、SQL 言語の文法上の拡張です。ANSI ANSI/ISO SQL 標準では INTO キーワードの使用が必須と定められているので、INTO キーワードがない INSERT 文は通知されます。ただし、Ultra Light アプリケーションでは INTO キーワードはオプションです。

キーのジョインもベンダー拡張として通知されます。ON 句を使用しないで、キーワード JOIN を使用する場合、キージョインがデフォルトで使用されます。キージョインは、既存の外部キー関係を使用して、テーブルをジョインします。キージョインは Ultra Light ではサポートされていません。たとえば、次のクエリは、Products テーブルと SalesOrderItems テーブル間の暗黙的なジョイン条件を指定しています。このクエリは、SQL Flagger でベンダー拡張として通知されます。

```
SELECT * FROM Products JOIN SalesOrderItems;
```

SQL Flagger 機能は、SQL 文の実行に依存しません。すべての通知論理は単に静的なコンパイル時の処理として行われます。

このセクションの内容:

SQL Flagger の起動 [517 ページ]
SQL Flagger を使用して、SQL 文や、SQL 標準に準拠している SQL 文のバッチをチェックすることができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

標準と互換性 [518 ページ]

データベースサーバと SQL プリプロセッサで使用される通知機能は、ANSI/ISO SQL 標準の第 1 部 (フレームワーク) で定義されている SQL Flagger 機能に従っています。

関連情報

キージョイン [407 ページ]

1.4.1.1 SQL Flagger の起動

SQL Flagger を使用して、SQL 文や、SQL 標準に準拠している SQL 文のバッチをチェックすることができます。

SQLFLAGGER 関数

SQLFLAGGER 関数は、文字列引数として渡された単一の SQL 文またはバッチが特定の SQL 標準に準拠しているかどうかを分析します。単一の文またはバッチは解析されますが、実行はされません。

sa_ansi_standard_packages システムプロシージャ

sa_ansi_standard_packages システムプロシージャは、単一の文またはバッチで、ANSI SQL/2008、SQL/2003、または SQL/1999 国際標準のオプションの SQL 言語機能またはパッケージが使用されていないかどうかを分析します。単一の文またはバッチは解析されますが、実行はされません。

sql_flagger_error_level オプションと sql_flagger_warning_level オプション

sql_flagger_error_level オプションと sql_flagger_warning_level オプションは、文が接続に対して準備または実行されると、SQL Flagger を起動します。文がオプション設定 (特定の ANSI 標準または Ultra Light) に準拠しない場合、文はエラー (SQLSTATE 0AW03) で終了するか、警告 (SQLSTATE 01W07) を返します。どちらの処理を行うかはオプション設定で決まります。文が準拠する場合は、文の実行が正常に続行されます。

SQL プリプロセッサ (sqlpp)

SQL プリプロセッサ (sqlpp) には、Embedded SQL アプリケーション内の静的 SQL 文をコンパイル時に通知する機能があります。この機能は、Ultra Light アプリケーションを開発するときに特に便利です。この機能で、SQL 文に Ultra Light との互換性があることを確認できます。

関連情報

バッチ [119 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.1.2 標準と互換性

データベースサーバと SQL プリプロセッサで使用される通知機能は、ANSI/ISO SQL 標準の第 1 部 (フレームワーク) で定義されている SQL Flagger 機能に従っています。

SQL Flagger では、SQL 言語構成の準拠を決定するときに、次の ANSI SQL 標準がサポートされます。

- SQL/1992 の Entry レベル、Intermediate レベル、Full レベル
- SQL/1999 のコアと SQL/1999 のオプションパッケージ
- SQL/2003 のコアと SQL/2003 のオプションパッケージ
- SQL/2008 のコアと SQL/2008 のオプションパッケージ

> **注記**
>
> SQL Flagger では、SQL/1992 (全レベル) はサポートされなくなりました。

SQL Flagger では、Ultra Light SQL に準拠しない文も特定できます。たとえば、Ultra Light では、スキーマオブジェクトの CREATE と ALTER 機能が限られています。

SQL 文はすべて SQL Flagger で分析できます。ただし、スキーマオブジェクトを作成または変更するほとんどの文 (テーブル、インデックス、マテリアライズドビュー、パブリケーション、サブスクリプション、プロキシテーブルを作成する文を含む) は、ANSI SQL 標準のベンダー拡張なので、準拠しないと通知されます。

SET OPTION 文とそのオプション要素は、どの SQL 標準に対しても、Ultra Light との互換性についても、準拠しないと通知されません。

1.4.2 他の SQL ソフトウェアとは異なる機能

他の SQL ソフトウェアとは異なる SQL の機能がいくつかあります。

豊富な SQL 機能が用意されていますが、それらの機能には、ローごとのトリガ、文ごとのトリガ、INSTEAD OF トリガ、SQL ストアドプロシージャ、ユーザ定義関数、RECURSIVE UNION クエリ、共通テーブル式、テーブル関数、LATERAL 派生テーブル、統合全文検索、Window 集合関数、正規表現検索、XML サポート、マテリアライズドビュー、スナップショットアイソレーション、参照整合性などがあります。

日付

日付、時刻、タイムスタンプのデータ型が提供されており、年、月、日、時、分、秒、小数点以下の秒が含まれます。日付フィールドへの挿入または更新、および日付フィールド間の比較については、フリーフォーマットの日付がサポートされています。

また、日付に関しては以下の演算が可能です。

日付 + 整数

指定された値の日数を日付に加えます。

日付 - 整数

指定された値の日数を日付から引きます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

date - date

2 つの日付間の日数の差を計算します。

date + time

日付と時刻からタイムスタンプを作成します。

INTERVAL 日付タイプはサポートされていません。これは ANSI/ISO SQL 標準の SQL 言語機能 F052 です。ただし、DATEADD などの日付と時刻の演算に使用できる数多くの関数があります。

エンティティ整合性と参照整合性

CREATE TABLE 文および ALTER TABLE 文の PRIMARY KEY 句と FOREIGN KEY 句によってエンティティ整合性と参照整合性の両方がサポートされています。

```
PRIMARY KEY [ CLUSTERED ] ( column-name [ ASC | DESC ], ... )
[NOT NULL] FOREIGN KEY [role-name]
           [(column-name [ ASC | DESC ], ...) ]
        REFERENCES table-name [(column-name, ...) ]
           [ MATCH [ UNIQUE | SIMPLE | FULL ] ]
           [ ON UPDATE [ CASCADE | RESTRICT | SET DEFAULT | SET NULL ] ]
           [ ON DELETE [ CASCADE | RESTRICT | SET DEFAULT | SET NULL ] ]
           [ CHECK ON COMMIT ] [ CLUSTERED ]
```

PRIMARY KEY 句では、テーブルのプライマリキーを宣言します。データベースサーバはプライマリキーカラムにユニークなインデックスを作成して、宣言されたプライマリキーがユニークであるよう管理します。このインデックスは、次の 2 つの文法上の拡張によってカスタマイズできます。

CLUSTERED

CLUSTERED キーワードは、プライマリキーインデックスはクラスタドインデックスであり、したがって、インデックス内の連続したインデックスエントリはテーブルの物理的に隣接するローを示すことを表します。

ASC | DESC

プライマリキーインデックスの各インデックスカラムのソートの程度 (昇順または降順) をカスタマイズできます。このカスタマイズを使用して、プライマリキーインデックスのソートの程度が、特定の SQL クエリに必要な、そのクエリの文の ORDER BY 句で指定されたソートの程度に一致するようにできます。

FOREIGN KEY 句は、2 つのテーブルの関係を定義します。その関係は、このテーブルのカラムが他のテーブルのプライマリキーの値を保有することによって確立しています。データベースサーバは、参照整合性制約を確保するように定義された FOREIGN KEY ごとに自動的にインデックスを構築します。制約のセマンティックおよびこのインデックスの物理特性は、次のようにカスタマイズできます。

CLUSTERED

CLUSTERED キーワードは、外部キーインデックスはクラスタドインデックスであり、したがって、インデックス内の連続したインデックスエントリは外部テーブルの物理的に隣接するローを示すことを表します。

ASC | DESC

外部キーインデックスの各インデックスカラムのソートの程度 (昇順または降順) をカスタマイズできます。外部キーインデックスのソートの程度は、プライマリキーインデックスのソートの程度と異なる場合があります。ソートの程度のカスタマイズを使用して、外部キーインデックスのソートの程度が、アプリケーションの特定の SQL クエリに必要な、そのクエリの文の ORDER BY 句で指定されたソートの程度に一致するようにできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

MATCH 句

ANSI/ISO SQL 標準の SQL 言語機能 F741 である MATCH 句がサポートされています。また、MATCH UNIQUE もサポートされており、これは、UNIQUE インデックスを追加する必要なしにプライマリテーブルと外部テーブル間の 1 対 1 の関係を確保します。

ユニークインデックス

NULL 入力可能のカラムへのユニークインデックス (ユニークセカンダリインデックスとも呼ばれる) の作成がサポートされています。デフォルトで、各インデックスキーはユニークであるか、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 つのインデックスエントリ ('a'、NULL) と ('a'、NULL) はそれぞれユニークインデックス値と見なされます。また、ユニークセカンダリインデックスもサポートされており、このインデックスでは、NULL 値が各ドメインで特別値として扱われます。これは、WITH NULLS NOT DISTINCT 句を使用して行われます。このようなインデックスでは、2 つの値のペア ('a'、NULL) と ('a'、NULL) は重複と見なされます。

ジョイン

INNER ジョイン、LEFT OUTER ジョイン、RIGHT OUTER ジョイン、FULL OUTER ジョインを使用することができます。明示的ジョイン述部に加えて、NATURAL ジョインとベンダー拡張 (KEY ジョインとも呼ばれる) を使用することもできます。ベンダー拡張では、テーブルの外部キー関係に基づいて暗黙的ジョイン述部を指定します。

CHAR、NCHAR、BINARY データ型

データベースサーバでは、固定長文字列型と可変長文字列型 (CHAR、NCHAR、または BINARY) は区別されません。また、このような値がデータベースに挿入されるときに、データベースサーバでは、後続ブランクは文字列タイプからトランケートされません。データベースサーバでは、NULL 値と空の文字列は区別されます。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照合を使用して、大文字と小文字を区別しない文字列比較をサポートしています。固定長文字列タイプにブランクが埋め込まれることはありません。ブランク埋め込みセマンティックは、各文字列比較の実行中にシミュレートされます。これらのセマンティックは、他の SQL ソフトウェアの文字列比較とは微妙に異なる場合があります。

UPDATE 文

SQL Anywhere は、オプションの ANSI/ISO SQL 言語機能 T111 を部分的にサポートしています。この機能では、UPDATE 文によってジョインがあるビューを参照できます。また、UPDATE 文と UPDATE WHERE CURRENT OF 文によって、文の SET 句で複数のテーブルを参照できます。さらに、UPDATE 文の FROM 句をジョインおよび派生テーブルを含む任意のテーブル式で構成できます。

SQL Anywhere では、UPDATE、INSERT、MERGE、DELETE の各文を別の SQL 文に派生テーブルとして埋め込むこともできます。このサポートの利点の 1 つは、UPDATE 文によって変更されたローのセットを返すクエリを簡単な方法で構築できることです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブル関数

SQL Anywhere では、ストアドプロシージャの結果セットを文の FROM 句内のテーブルとして参照できます。この機能は、一般にテーブル関数と呼ばれています。テーブル関数は、ANSI/ISO SQL 標準の SQL 言語機能 T326 です。この標準では、テーブル関数は TABLE キーワードを使用して指定されます。SQL Anywhere では、TABLE キーワードを使用する必要はありません。ストアドプロシージャは FROM 句で直接参照され、オプションで相関名と結果セットのスキーマの指定がプロシージャによって返されます。

次の例では、ストアドプロシージャの結果 ShowCustomerProducts をベーステーブル Products にジョインします。ストアドプロシージャの参照には、WITH 句を使用して、プロシージャの結果のスキーマの明示的な宣言が付随しています。

```
SELECT sp.ident, sp.quantity, Products.name
FROM ShowCustomerProducts( 149 )
      WITH ( ident INT, description CHAR(20), quantity INT ) sp
    JOIN Products ON sp.ident = Products.ID
```

マテリアライズドビュー

SQL Anywhere はマテリアライズドビューをサポートしています。これは、SQL クエリから直接的または間接的に参照できる、事前に計算された結果セットです。SQL Anywhere では、CREATE MATERIALIZED VIEW 文を使用して、即時保存ビューと手動保存ビューの両方を作成できます。他のデータベース製品では、異なる語を使用してこの機能を記述する場合があります。

カーソル

SQL Anywhere では、ANSI/ISO SQL 標準のオプションの SQL 言語機能 F431 をサポートしています。SQL Anywhere では、明示的に FORWARD ONLY を宣言されないかぎり、すべてのカーソルは双方向にスクロール可能です。アプリケーションでは、FETCH 文によって相対位置または絶対位置を使用するか、ODBC などの他のアプリケーションプログラミングインタフェースによって同等のものを使用して、カーソルによってスクロールできます。

SQL Anywhere は、value-sensitive カーソルとローメンバーシップ sensitive カーソルをサポートしています。INSENSITIVE、KEYSET-DRIVEN、SENSITIVE カーソルなどの一般的にサポートされているカーソルタイプがサポートされます。Embedded SQL を使用する場合は、FETCH 文でカーソル位置を任意に移動できます。カーソルは現在位置に対して前後させるか、カーソルの最初または最後からレコード数を指定して移動できます。

デフォルトで、Embedded SQL および SQL プロシージャ、ユーザ定義関数、トリガのカーソルは更新可能です。これらのカーソルは、FOR UPDATE 句を使用して明示的に更新可能にできます。ただし、FOR UPDATE 句を単独で指定しても、カーソルの結果セットのローに対するロックは取得されません。結果セットのローが他のトランザクションによって変更されないようにするには、次のいずれかを指定します。

FOR UPDATE BY LOCK

データベースサーバは、この句によって、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。

FOR UPDATE BY { VALUES | TIMESTAMP }

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere 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。

エイリアスの参照

SQL Anywhere では、クエリの SELECT リスト内のエイリアスの式を、クエリの他の部分で参照できます。他のほとんどの SQL システムと ANSI/ISO SQL 標準にはこの機能はありません。たとえば、次のような SQL クエリを指定できます。

```
SELECT column-or-expression AS alias-name
FROM table-reference
WHERE alias-name = expression
```

エイリアスは、SELECT ブロックの任意の場所に使用できます。エイリアスは他の SELECT リスト式にも使用できます。この場合、追加のエイリアスが定義されます。循環的なエイリアスの参照は許可されません。式に指定されたエイリアスが SELECT ブロックのネームスペースにある変数名またはカラム名と同じ場合は、エイリアス定義によってそのカラムまたは変数が遮断されます。ただし、そのような場合は、テーブルによってカラム名を明示的に修飾できます。

スナップショットアイソレーション

SQL Anywhere は、マルチバージョン同時実行性制御 (MVCC) とも呼ばれる、スナップショットアイソレーションをサポートしています。スナップショットアイソレーションをサポートする他の SQL 実装では、書き込み競合、つまり 2 つ以上のトランザクションによる同じローへの同時更新は、COMMIT 時にのみ明らかになります。この場合、通常は最初の COMMIT が優先され、競合に関与するその他のトランザクションはアボートされる必要があります。

SQL Anywhere では、スナップショットトランザクションが ANSI アイソレーションレベルでトランザクション実行と共存できるように、ローへの書き込み操作によってローの書き込みロックが取得されます。このため、SQL Anywhere での書き込み競合ではブロックが発生しますが、正確な動作は BLOCKING と BLOCKING_TIMEOUT の接続オプションによって制御できます。

関連情報

キージョイン [407 ページ]
マテリアライズドビュー [64 ページ]
スナップショットアイソレーション [766 ページ]

1.4.3 Watcom SQL

SQL Anywhere がサポートしている SQL ダイアレクトは Watcom SQL と呼ばれています。

SQL Anywhere は、最初にリリースされた 1992 年当時には Watcom SQL と呼ばれていました。SQL Anywhere がサポートする SQL ダイアレクトについては、現在でも Watcom SQL という用語が使用されています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

また、SQL Anywhere は、SAP Adaptive Server Enterprise がサポートする SQL ダイアレクトである Transact-SQL の大部分のサブセットもサポートしています。

関連情報

Transact-SQL との互換性 [523 ページ]

1.4.4 Transact-SQL との互換性

SQL Anywhere は、SAP Adaptive Server Enterprise がサポートする SQL ダイアレクトである Transact-SQL の大部分のサブセットをサポートしています。

目的

次に、SQL Anywhere の Transact-SQL に対するサポートの目的を示します。

アプリケーションの移植性

アプリケーション、ストアドプロシージャ、バッチファイルの多くは、Adaptive Server Enterprise と SQL Anywhere の両方のデータベースで使用できるように作成できます。

データの移植性

SQL Anywhere データベースと Adaptive Server Enterprise データベースの間では、最小限の作業でデータの交換とレプリケートができます。

目的は、Adaptive Server Enterprise と SQL Anywhere の両方で動作するアプリケーションを作成することです。既存の Adaptive Server Enterprise アプリケーションの多くは、SQL Anywhere データベースで実行するには多少の変更が必要です。

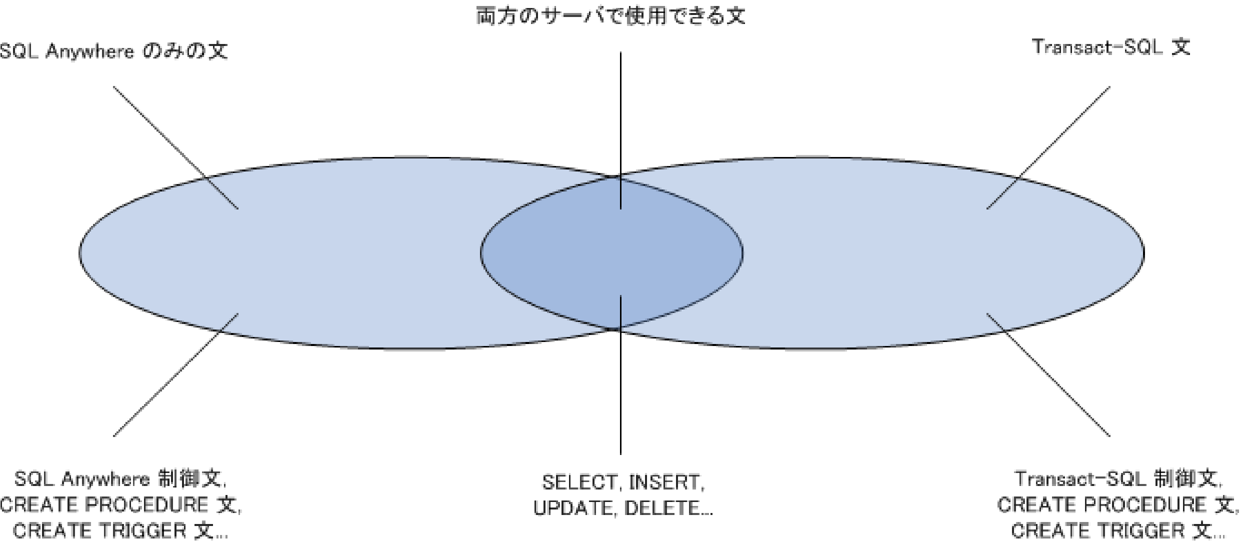
Transact-SQL のサポート方法

次に、SQL Anywhere での Transact-SQL に対するサポート方法を示します。

- SQL 文の多くは、SQL Anywhere と Adaptive Server Enterprise 間で互換性があります。
- 特にプロシージャ、トリガ、バッチで使用されるプロシージャ言語で書かれた文のいくつかでは、これまでの SQL Anywhere のバージョンでサポートされる構文とともに、別の Transact-SQL 文がサポートされます。このような文については、SQL Anywhere では 2 種類の SQL ダイアレクトがサポートされています。これらのダイアレクトは Transact-SQL (Adaptive Server Enterprise のダイアレクト) および Watcom-SQL (SQL Anywhere のダイアレクト) と呼ばれています。
- プロシージャ、トリガ、バッチは Transact-SQL または Watcom-SQL ダイアレクトのどちらでも実行されます。バッチまたはプロシージャ内では、1 つのダイアレクトの制御文だけを通して使用します。たとえば、ダイアレクトごとに異なったフロー制御文があります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

2 つのダイアレクトが重なり合う状況を、次の図に示します。



類似点と相違点

SQL Anywhere は、既存のデータの処理に使用する、Transact-SQL 言語の要素、関数、文の多くをサポートします。たとえば SQL Anywhere は、すべての数値関数、集合関数、日付と時刻関数をサポートし、1 つを除くすべての文字列関数をサポートしています。別の例として、ジョインを使う拡張された DELETE 文と UPDATE 文が SQL Anywhere でサポートされます。

さらに、SQL Anywhere では、Transact-SQL のストアドプロシージャ言語 (CREATE PROCEDURE 文、CREATE TRIGGER 文、制御文など) と、多くの Transact-SQL データのデータ定義言語文がサポートされます。

それぞれの製品にサポートされる構造と設定については設計上の相違点があります。デバイス管理、ユーザ管理、バックアップなどの管理タスクの多くはシステム固有のものです。SQL Anywhere は Transact-SQL のシステムテーブルをビューとして提供しますが、SQL Anywhere にとって意味を持たないテーブルにはローがありません。また、SQL Anywhere は、一般的な管理タスクの一部として一連のシステムプロシージャを提供します。

Transact-SQL のみ

SQL Anywhere がサポートする SQL 文には、一方のダイアレクトの一部になっていて、もう一方のダイアレクトの一部になっていないものがあります。このような 2 つのダイアレクトは、1 つのプロシージャ、トリガ、またはバッチ内で混在させることはできません。たとえば、次の文は SQL Anywhere ではサポートされますが、それは Transact-SQL ダイアレクトのみに属するものとしてです。

- Transact-SQL 制御文の IF と WHILE
- Transact-SQL の EXECUTE 文
- Transact-SQL の CREATE PROCEDURE 文と CREATE TRIGGER 文

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- Transact-SQL の BEGIN TRANSACTION 文
- セミコロンで区切られていない SQL 文は、Transact-SQL のプロシージャまたはバッチに属しています。

SQL Anywhere のみ

Adaptive Server Enterprise では、次の文をサポートしません。

- LOOP 制御文および FOR 制御文
- IF と WHILE の SQL Anywhere バージョン
- CALL 文
- SIGNAL 文
- CREATE PROCEDURE 文、CREATE FUNCTION 文、CREATE TRIGGER 文の SQL Anywhere バージョン

注記

1 つのプロシージャ、トリガ、またはバッチ内では、2 つのダイアレクトを混在させることはできません。

- Transact-SQL のみの文を、両方のダイアレクトに属する文とともにバッチ、プロシージャ、またはトリガ内に含めることができます。
- Adaptive Server Enterprise によってサポートされない文を、両サーバによってサポートされる文とともに、バッチ、プロシージャ、またはトリガ内に含めることができます。
- Transact-SQL のみの文を、SQL Anywhere のみの文とともにバッチ、プロシージャ、またはトリガに含めることはできません。

1.4.5 SQL Anywhere と Adaptive Server Enterprise の比較

Adaptive Server Enterprise と SQL Anywhere はそれぞれの異なる目的に合わせたアーキテクチャを持つ、相互に補完的な製品です。

SQL Anywhere には、互換性のあるデータベース管理向けに Adaptive Server Enterprise に似たツールが含まれています。

このセクションの内容:

Adaptive Server Enterprise のサーバとデータベース [526 ページ]
サーバとデータベースの関係は、Adaptive Server Enterprise と SQL Anywhere では異なります。

Adaptive Server Enterprise のデバイス管理 [527 ページ]
SQL Anywhere と Adaptive Server Enterprise は、用途の違いを反映して、異なったモデルを使用してデバイスとディスク領域を管理します。

Adaptive Server Enterprise のデフォルトとルール [527 ページ]
SQL Anywhere は、Transact-SQL の CREATE DEFAULT 文や CREATE RULE 文はサポートしません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Adaptive Server Enterprise のシステムテーブル [527 ページ]
SQL Anywhere には、独自のシステムテーブルのほかに、Adaptive Server Enterprise のシステムテーブルに対応する箇所をシミュレートするシステムビューがあります。

Adaptive Server Enterprise における管理者の役割 [528 ページ]
Adaptive Server Enterprise は SQL Anywhere より管理上の役割が充実しています。

Adaptive Server Enterprise のユーザとグループ [529 ページ]
SQL Anywhere では、ユーザとグループを管理する Adaptive Server Enterprise のシステムプロシージャがいくつかサポートされています。

1.4.5.1 Adaptive Server Enterprise のサーバとデータベース

サーバとデータベースの関係は、Adaptive Server Enterprise と SQL Anywhere では異なります。

Adaptive Server Enterprise では各データベースはサーバ内に存在し、各サーバは複数のデータベースを持つことができます。ユーザはサーバに対するログイン権限を持っている場合、サーバに接続できます。サーバに接続すると、ユーザはサーバ上のパーミッションを持つ各データベースを使用できます。master データベースに保持されている、システム全体に適用されるシステムテーブルは、サーバ上のすべてのデータベースに共通な情報を含んでいます。

SQL Anywhere に マスタデータベースがない

SQL Anywhere では、Adaptive Server Enterprise のマスタデータベースに対応するレベルはありません。その代わり、それぞれのデータベースが独立したエンティティであり、専用のシステムテーブルを持っています。ユーザは、サーバに対してではなく 1 つのデータベースに対する接続権限を持ちます。ユーザが接続する場合、個々のデータベースに対する接続となります。master データベースレベルで維持されているシステム全体にわたる一連のシステムテーブルはありません。SQL Anywhere の各データベースサーバは、複数のデータベースを動的にロード、アンロードできます。ユーザはそれぞれのデータベースに対する独立した接続を維持できます。

SQL Anywhere は、Transact-SQL のサポートと Open Server のサポートに対して、Adaptive Server Enterprise に似た方法でタスクを実行するツールを提供します。たとえば、SQL Anywhere は、Adaptive Server Enterprise の sp_addlogin システムプロシージャの実装を提供し、同等の処理、つまりデータベースへのユーザの追加を実行します。

ファイル操作文

SQL Anywhere では、Transact-SQL 文 DUMP DATABASE と LOAD DATABASE を使用したバックアップとリストアはサポートされていません。代わりに SQL Anywhere には、構文が異なる BACKUP DATABASE 文と RESTORE DATABASE 文があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.5.2 Adaptive Server Enterprise のデバイス管理

SQL Anywhere と Adaptive Server Enterprise は、用途の違いを反映して、異なったモデルを使用してデバイスとディスク領域を管理します。

Adaptive Server Enterprise は多種多様な Transact-SQL 文を使用する包括的なリソース管理スキームを作成しますが、SQL Anywhere は独自のリソースを自動的に管理し、そのデータベースは通常のオペレーティングシステムファイルです。

SQL Anywhere は、DISK INIT、DISK MIRROR、DISK REFIT、DISK REINIT、DISK REMIRROR、DISK UNMIRROR などの Transact-SQL DISK 文をサポートしません。

1.4.5.3 Adaptive Server Enterprise のデフォルトとルール

SQL Anywhere は、Transact-SQL の CREATE DEFAULT 文や CREATE RULE 文はサポートしません。

CREATE DOMAIN 文を使用すると、デフォルトとルール (いわゆる、CHECK 条件) がドメインの定義に組み込まれるので、Transact-SQL の CREATE DEFAULT 文と CREATE RULE 文に対して、類似した機能性が与えられます。

SQL Anywhere では、ドメインはデフォルト値とそれに対応した CHECK 条件を持つことができ、その CHECK 条件はそのデータ型に基づいて定義されたすべてのカラムに適用されます。ドメインを作成するには、CREATE DOMAIN 文を使用します。

デフォルト値とルール、または CHECK 条件は、個々のカラムについて、CREATE TABLE 文または ALTER TABLE 文を使用して定義できます。

Adaptive Server Enterprise では、CREATE DEFAULT 文は名前付きデフォルトを作成します。このデフォルトは、特定のカラムにバインドすることによって、カラムのデフォルト値として使用できます。また、sp_bindefault システムプロシージャを使用してドメインのカラムにバインドすると、ドメインのすべてのカラムのデフォルト値として使用できます。CREATE RULE 文は、名前のついたルールを作成します。このルールは、特定のカラムにバインドすることによって、さまざまなカラムのドメイン定義に使用でき、データ型にバインドすると、そのドメインのすべてのカラムのルールとして使用できます。ルールをデータ型やカラムにバインドするには、sp_bindrule システムプロシージャを使用します。

1.4.5.4 Adaptive Server Enterprise のシステムテーブル

SQL Anywhere には、独自のシステムテーブルのほかに、Adaptive Server Enterprise のシステムテーブルに対応する箇所をシミュレートするシステムビューがあります。

SQL Anywhere のシステムテーブルは各データベース内に格納されていますが、Adaptive Server Enterprise のシステムテーブルの場合、一部は各データベース内、一部はマスタデータベース内に格納されています。SQL Anywhere のアーキテクチャはマスタデータベースを含みません。

Adaptive Server Enterprise では、データベース所有者 (ユーザ dbo) がシステムテーブルを所有します。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。ユーザ dbo は、SQL Anywhere が提供する Adaptive Server Enterprise と互換性のあるシステムビューを所有します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.5.5 Adaptive Server Enterprise における管理者の役割

Adaptive Server Enterprise は SQL Anywhere より管理上の役割が充実しています。

Adaptive Server Enterprise では、複数のログインアカウントに役割を付与でき、1 つのアカウントが複数の役割を処理できますが、独特の役割のセットもあります。

Adaptive Server Enterprise の役割

Adaptive Server Enterprise の独特の役割には次のものがあります。

システム管理者

特定のアプリケーションに関連していない一般的な管理タスクを行い、あらゆるデータベースオブジェクトにアクセスできます。

システムセキュリティ担当者

Adaptive Server Enterprise のセキュリティが問題となるタスクを行いますが、データベースオブジェクトには特別のパーミッションを持ちません。

データベース所有者

自分が所有者であるデータベース内のオブジェクトに対して、完全な権限を持ちます。また、データベースにユーザを追加したり、データベース内でオブジェクトの作成や文の実行を行うのに必要な権限を他のユーザに付与したりできます。

データ定義文

CREATE TABLE や CREATE VIEW などの特定のデータ定義文に対する権限をユーザに与え、ユーザがデータベースオブジェクトを作成できるようにします。

オブジェクト所有者

各データベースオブジェクトには所有者があり、そのオブジェクトにアクセスする権限を他のユーザに与えることができます。オブジェクトの所有者は、自動的にそのオブジェクトに対するすべての権限を持ちます。

- データベース管理者は、Adaptive Server Enterprise のデータベース所有者のように、所有するデータベース内のすべてのオブジェクト (SYS が所有するオブジェクトは除く) に対して完全な権限を持ち、他のユーザにデータベース内でのオブジェクト作成と文実行に必要な権限を付与できます。デフォルトのデータベース管理者は、ユーザ DBA です。
- リソースロールは、ユーザがデータベース内でオブジェクトを作成するのを許可します。これは Adaptive Server Enterprise で個々の CREATE 文にパーミッションを付与する代わりのものです。
- SQL Anywhere のオブジェクト所有者は、Adaptive Server Enterprise の所有者と同じです。オブジェクト所有者は権限を付与することも含め、そのオブジェクトに関するすべての権限を自動的に持ちます。

Adaptive Server Enterprise と SQL Anywhere の両方に含まれるデータにスムーズにアクセスするために、適切な権限を持つユーザ ID をデータベースに作成し、このユーザ ID からオブジェクトを作成してください。同じユーザ ID を両方の環境で使用すると、オブジェクト名と修飾子が 2 つのデータベースで同一になり、互換性のあるアクセスが可能になります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.5.6 Adaptive Server Enterprise のユーザとグループ

SQL Anywhere では、ユーザとグループを管理する Adaptive Server Enterprise のシステムプロシージャがいくつかサポートされています。

システムプロシージャ	説明
sp_addlogin	Adaptive Server Enterprise では、ユーザをサーバに追加します。SQL Anywhere では、ユーザをデータベースに追加します。
sp_adduser	Adaptive Server Enterprise と SQL Anywhere の両方で、ユーザをデータベースに追加します。Adaptive Server Enterprise では、これは sp_addlogin とは異なるタスクですが、SQL Anywhere では同じです。
sp_addgroup	グループをデータベースに追加します。
sp_changegroup	ユーザをグループに追加するか、ユーザをあるグループから別のグループに移動します。
sp_droplogin	Adaptive Server Enterprise では、ユーザをサーバから削除します。SQL Anywhere では、ユーザをデータベースから削除します。
sp_dropuser	ユーザをデータベースから削除します。
sp_dropgroup	グループをデータベースから削除します。

Adaptive Server Enterprise では、ログイン ID はサーバ全体に適用されます。SQL Anywhere では、ユーザは個々のデータベースに属します。

データベースオブジェクトに適用される権限

個々のデータベースオブジェクトに対する権限を付与する GRANT 文と REVOKE 文は、Adaptive Server Enterprise と SQL Anywhere でよく似ています。どちらの文も、データベースのテーブルとビューに対する SELECT、INSERT、DELETE、UPDATE、REFERENCES 権限を付与でき、データベースのテーブルの選択したカラムに対する UPDATE 権限を付与できます。どちらも、ストアドプロシージャに対する EXECUTE 権限を付与できます。

たとえば、次の文は Adaptive Server Enterprise と SQL Anywhere の両方で有効です。

```
GRANT INSERT, DELETE
ON Employees
TO MARY, SALES;
```

この文は、Employees テーブルで INSERT 文と DELETE 文を使用するのに必要な権限を MARY というユーザと SALES グループに付与します。

SQL Anywhere と Adaptive Server Enterprise の両方で、WITH GRANT OPTION 句は、権限を付与されるユーザがそれを他のユーザに付与することを許可します。ただし、SQL Anywhere は WITH GRANT OPTION を GRANT EXECUTE 文で使用することを許可しません。SQL Anywhere では、WITH GRANT OPTION はユーザにのみ指定できます。グループのメンバーは、グループに付与されている WITH GRANT OPTION を継承しません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベース全体に適用される権限

Adaptive Server Enterprise と SQL Anywhere は、データベース全体に適用される権限に異なったモデルを使用します。SQL Anywhere は DBA ロールを使用して、データベース内での完全な権限をユーザに許可します。Adaptive Server Enterprise のシステム管理者は、この権限をサーバ上のすべてのデータベースに対して使用できます。しかし、SQL Anywhere データベース上での DBA ロールは、Adaptive Server Enterprise のデータベース所有者のパーミッションとは異なります。Adaptive Server Enterprise のデータベース所有者は、他のユーザが所有するオブジェクトに対するパーミッションを得るには、Adaptive Server Enterprise の SETUSER 文を使用してください。

1.4.6 Transact-SQL と互換性のあるデータベース

データベースを作成するとき、または既存のデータベースを再構築するときに適切なオプションを選択すると、SQL Anywhere と Adaptive Server Enterprise の相違点のいくつかを解除できます。

他の相違点は、SQL Anywhere では SET TEMPORARY OPTION 文、Adaptive Server Enterprise では SET 文を使用して接続レベルオプションを設定することで制御できます。

大文字と小文字を区別するデータベースの作成

デフォルトでは、Adaptive Server Enterprise データベースでの文字列比較は大文字と小文字を区別しますが、SQL Anywhere では大文字と小文字を区別しません。

SQL Anywhere を使用して Adaptive Server Enterprise と互換性のあるデータベースを構築する場合は、大文字と小文字を区別するオプションを選択します。

- SQL Central を使用している場合は、このオプションはデータベース作成ウィザードにあります。
- dbinit ユーティリティを使用している場合は、-c オプションを指定します。
- CREATE DATABASE 文を使用する場合は、CASE RESPECT 句を指定します。

比較の後続ブランクを無視

SQL Anywhere を使用して Adaptive Server Enterprise と互換性のあるデータベースを構築するときには、比較するときに後続ブランクを無視するオプションを選択します。

- SQL Central を使用している場合は、このオプションはデータベース作成ウィザードにあります。
- dbinit ユーティリティを使用している場合は、-b オプションを指定します。
- CREATE DATABASE 文を使用する場合は、BLANK PADDING ON 句を指定します。

このオプションを選択すると、Adaptive Server Enterprise と SQL Anywhere では次の 2 つの文字列を等しいと見なします。

```
'ignore the trailing blanks    '
'ignore the trailing blanks'
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このオプションを選択しないと、SQL Anywhere ではこの 2 つの文字列は異なっていると見なします。

このオプションを選択すると、クライアントアプリケーションでフェッチした文字列にブランクが埋め込まれます。

古いシステムビューの削除

SQL Anywhere の古いバージョンでは 2 つのシステムビューを使用していましたが、互換性を確保しようとすると、それらの名前は Adaptive Server Enterprise のシステムビューと矛盾します。その 2 つのビューは SYSCOLUMNS と SYSINDEXES です。Open Client または JDBC のインタフェースを使用している場合は、これらのビューを除外してデータベースを作成します。この処理には dbinit -k オプションを使用します。

データベースの作成時にこのオプションを使用しなかった場合、`SELECT * FROM SYSCOLUMNS;` を実行すると、SQLE_AMBIGUOUS_TABLE_NAME というエラーが発生します。

このセクションの内容:

Transact-SQL と互換性のあるデータベースの作成 (SQL Central の場合) [532 ページ]
SQL Central を使用して、Transact-SQL と互換性のあるデータベースを作成します。

Transact-SQL と互換性のあるデータベースの作成 (コマンドラインの場合) [532 ページ]
初期化ユーティリティ (dbinit) を使用して、Transact-SQL と互換性のあるデータベースを作成します。

Transact-SQL と互換性のあるデータベースの作成 (SQL の場合) [533 ページ]
CREATE DATABASE 文を使用して、Transact-SQL と互換性のあるデータベースを作成します。

Transact-SQL 互換性のオプション [534 ページ]
データベースオプションの設定値のいくつかは Transact-SQL の動作を規定します。

大文字と小文字の区別 [535 ページ]
データベースでの大文字と小文字の区別は、データ、識別子、パスワードに影響します。

オブジェクト名の互換性 [536 ページ]
データベースオブジェクトは、ネームスペース内にユニークな名前を持っていなければなりません。

Transact-SQL の特殊な TIMESTAMP カラムとデータ型 [536 ページ]
Transact-SQL の特殊な TIMESTAMP カラムがサポートされています。

特殊な IDENTITY カラム [537 ページ]
IDENTITY カラムの値はテーブルの各ローを一意に識別します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.6.1 Transact-SQL と互換性のあるデータベースの作成 (SQL Central の場合)

SQL Central を使用して、Transact-SQL と互換性のあるデータベースを作成します。

前提条件

デフォルトでは、SERVER OPERATOR システム権限が必要です。-gu データベースサーバオプションを使用すると、必要な権限を変更できます。

手順

1. SQL Central で、*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. **ツール** › *SQL Anywhere17* › **データベースの作成** をクリックします。
3. ウィザードの指示に従います。

   追加設定の指定画面で、*Adaptive Server Enterprise* **のエミュレート**をクリックして、**次へ**をクリックします。
4. ウィザードの残りの指示に従います。

結果

Transact-SQL と互換性のあるデータベースが作成されます。このデータベースは、ブランクが埋め込まれ、大文字と小文字が区別されます。また、SYS.SYSCOLUMNS と SYS.SYSINDEXES のシステムビューはありません。

1.4.6.2 Transact-SQL と互換性のあるデータベースの作成 (コマンドラインの場合)

初期化ユーティリティ (dbinit) を使用して、Transact-SQL と互換性のあるデータベースを作成します。

手順

次の dbinit コマンドを実行します。

```
dbinit -b -c -k -dba DBA,passwd db-name.db
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このコマンドでは、-b でデータベースにブランクを埋め込み、-c でデータベースで大文字と小文字が区別されるようにし、-k で SYS.SYSCOLUMNS と SYS.SYSINDEXES のシステムビューが作成されないようにします。

結果

Transact-SQL と互換性のあるデータベースが作成されます。

1.4.6.3 Transact-SQL と互換性のあるデータベースの作成 (SQL の場合)

CREATE DATABASE 文を使用して、Transact-SQL と互換性のあるデータベースを作成します。

前提条件

デフォルトでは、SERVER OPERATOR システム権限が必要です。-gu データベースサーバオプションを使用すると、必要な権限を変更できます。

手順

1. SQL Anywhere データベースに接続します。
2. Interactive SQL で次の文を入力します。

```
CREATE DATABASE 'db-name.db'
DBA USER 'DBA' DBA PASSWORD 'passwd'
ASE COMPATIBLE
CASE RESPECT
BLANK PADDING ON;
```

   この文で ASE COMPATIBLE 句を使用すると、SYS.SYSCOLUMNS と SYS.SYSINDEXES のシステムビューが作成されないようになります。

結果

Transact-SQL と互換性のあるデータベースが作成されます。このデータベースは、ブランクが埋め込まれ、大文字と小文字が区別されます。また、SYS.SYSCOLUMNS と SYS.SYSINDEXES のシステムビューはありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.6.4 Transact-SQL 互換性のオプション

データベースオプションの設定値のいくつかは Transact-SQL の動作を規定します。

データベースオプションの設定は、SET OPTION 文を使用します。

allow_nulls_by_default オプションの設定

デフォルトでは、Adaptive Server Enterprise はカラムに対して NULL を許可すると定義しないかぎり、新しいカラムに NULL を許可しません。ソフトウェアによってデフォルトで新しいカラムに NULL が認められますが、これは ANSI/ISO SQL 標準と互換性があります。

Adaptive Server Enterprise を ANSI/ISO SQL 標準互換で動作させるには、sp_dboption システムプロシージャを使用して allow_nulls_by_default オプションを true に設定します。

ソフトウェアを Transact-SQL 互換で動作させるには、allow_nulls_by_default オプションを Off に設定します。これを行うには、次のように SET OPTION 文を使用します。

```
SET OPTION PUBLIC.allow_nulls_by_default = 'Off';
```

quoted_identifier オプションの設定

Adaptive Server Enterprise のデフォルトの識別子と文字列の処理は、ANSI/ISO SQL 標準に一致する SQL Anywhere とは異なります。

quoted_identifier オプションは、Adaptive Server Enterprise と SQL Anywhere の両方で使用できます。識別子と文字列が互換性を持って処理されるように、両方のデータベースでこのオプションが同じ値に設定されていることを確認します。

ANSI/ISO SQL 標準に準拠させるには、Adaptive Server Enterprise と SQL Anywhere の両方で quoted_identifier オプションを On に設定します。

Transact-SQL に準拠させるには、Adaptive Server Enterprise と SQL Anywhere の両方で quoted_identifier オプションを Off に設定します。この設定では、キーワードと同じように複数の同一の識別子を二重引用符で囲んで使用することはできません。quoted_identifier を Off に設定しないで、アプリケーション内の SQL 文で使用しているすべての文字列を二重引用符ではなく一重引用符で囲む方法もあります。

string_rtruncation オプションの設定

Adaptive Server Enterprise と SQL Anywhere では、string_rtruncation オプションがサポートされています。このオプションは、INSERT 文または UPDATE 文字列がトランケートされたときのエラーメッセージの表示を制御します。各データベースのオプション設定は同じ値にしてください。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.6.5 大文字と小文字の区別

データベースでの大文字と小文字の区別は、データ、識別子、パスワードに影響します。

データ

データの大文字と小文字を区別するかしないかは、インデックスなどに反映されます。

識別子

識別子には、テーブル名、カラム名などがあります。

パスワード

SQL Anywhere データベースのパスワードは常に大文字と小文字が区別されます。

データの大文字と小文字の区別

SQL Anywhere のデータ比較を行う場合の大文字と小文字の区別については、データベース作成時に決定します。SQL Anywhere データベースは、デフォルトでは、データは常に入力されたとおりの大文字と小文字で保持されていますが、比較において大文字と小文字を区別しません。

Adaptive Server Enterprise での大文字と小文字の区別は、Adaptive Server Enterprise システムにインストールされているソート順によって異なります。大文字と小文字の区別は、シングルバイト文字セットの場合は、Adaptive Server Enterprise のソート順を再設定して変更できます。

識別子の大文字と小文字の区別

SQL Anywhere は、大文字と小文字を区別する識別子をサポートしていません。Adaptive Server Enterprise では、識別子の大文字と小文字の区別はデータの大文字と小文字の区別に従います。

SQL Anywhere では、Java データ型を例外として、識別子の大文字と小文字を区別しません。

パスワードの大文字と小文字の区別

SQL Anywhere パスワードについては、常に大文字と小文字が区別されます。

Adaptive Server Enterprise では、ユーザ ID とパスワードの大文字と小文字の区別は、サーバの大文字と小文字の区別に従います。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.6.6 オブジェクト名の互換性

データベースオブジェクトは、ネームスペース内にユニークな名前を持っていなければなりません。

ネームスペース外では重複した名前も許可されます。データベースオブジェクトによっては、Adaptive Server Enterprise と SQL Anywhere で異なるネームスペースを持っています。

Adaptive Server Enterprise は、トリガ名について SQL Anywhere よりも制限の多いネームスペースを持っています。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。互換性のある SQL を作成するには、Adaptive Server Enterprise の、データベース内でユニークなトリガ名を必要とする制限を採用してください。

1.4.6.7 Transact-SQL の特殊な TIMESTAMP カラムとデータ型

Transact-SQL の特殊な TIMESTAMP カラムがサポートされています。

TIMESTAMP カラムは、TSEQUAL システム関数とともに、ローが更新されたかどうかチェックするのに使用されます。

> **i 注記**
>
> SQL Anywhere は正確な日付と時間の情報を持つ TIMESTAMP データ型を持っています。これは、Transact-SQL の特殊な TIMESTAMP カラムとデータ型とは異なります。

SQL Anywhere での Transact-SQL の TIMESTAMP カラムの作成

Transact-SQL の TIMESTAMP カラムを作成するには、(SQL Anywhere の) TIMESTAMP データ型とタイムスタンプのデフォルト設定値を持つカラムを作成します。このカラムにはどのような名前も付けられますが、通常は timestamp が使われます。

次に、Transact-SQL の TIMESTAMP カラムを含む CREATE TABLE 文の例を示します

```
CREATE TABLE tablename (
   column_1 INTEGER,
   column_2 TIMESTAMP DEFAULT TIMESTAMP
);
```

次の ALTER TABLE 文は、SalesOrders テーブルに Transact-SQL の TIMESTAMP カラムを追加します。

```
ALTER TABLE SalesOrders
ADD timestamp TIMESTAMP DEFAULT TIMESTAMP;
```

Adaptive Server Enterprise ではカラム名が timestamp で、データ型の指定がなければ自動的に TIMESTAMP データ型が与えられます。SQL Anywhere ではデータ型を明示的に割り当てる必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

TIMESTAMP カラムのデータ型

Adaptive Server Enterprise では、TIMESTAMP カラムはドメインの NULL を許容する VARBINARY(8) として扱われます。SQL Anywhere では、TIMESTAMP カラムは日付と時間からなる TIMESTAMP データ型として扱われ、時間は秒以下小数点 6 桁からなります。

後で更新するためにテーブルからフェッチするときは、TIMESTAMP 値がフェッチされる先の変数は、カラムの記述に従わなければなりません。

Interactive SQL では、ローの値の違いを調べるために timestamp_format オプションを設定することが必要な場合があります。次の文は、秒の小数点以下 6 桁すべてを表示するように timestamp_format オプションを設定します。

```
SET OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSSSSS';
```

小数点以下 6 桁の数字すべてが表示されなければ、TIMESTAMP カラムの値は等しいように見えても、実際は等しくないことがあります。

更新時の TSEQUAL の使用

TSEQUAL システム関数を使えば、TIMESTAMP カラムが更新されたかどうかがわかります。

アプリケーションが TIMESTAMP カラムを変数に SELECT したとします。選択されたロー内の 1 つの UPDATE が送信されたときに、アプリケーションは TSEQUAL 関数を使用してそのローが修正されたかどうかをチェックできます。TSEQUAL 関数は、テーブルの TIMESTAMP 値を SELECT で取得した TIMESTAMP 値と比較します。これらの値が同じなら、変更はありません。値が違う場合は、ローが SELECT の実行以降に変更されています。次に例を示します。

```
CREATE VARIABLE old_ts_value TIMESTAMP;
```

```
SELECT timestamp into old_ts_value
FROM publishers
WHERE pub_id = '0736';
```

```
UPDATE publishers
SET city = 'Springfield'
WHERE pub_id = '0736'
AND TSEQUAL(timestamp, old_ts_value);
```

1.4.6.8 特殊な IDENTITY カラム

IDENTITY カラムの値はテーブルの各ローを一意に識別します。

IDENTITY カラムは、送り状番号や従業員番号のような連続番号を格納します。このカラムは、自動生成されます。

Adaptive Server Enterprise では、データベースの各テーブルは IDENTITY カラムを 1 つだけ持つことができます。データ型は numeric、位取りは 0 (ゼロ)、NULL 値は許可されません。

SQL Anywhere では、IDENTITY カラムはカラムのデフォルトとして設定されます。連続番号でない値も、INSERT 文を使用してカラムに明示的に挿入できます。Adaptive Server Enterprise は、identity_insert オプションが *on* に設定されないかぎ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

り、identity カラムへの INSERT を許可しません。SQL Anywhere では、NOT NULL プロパティを設定する必要があります。また、1 つのカラムのみが IDENTITY カラムであることを確認する必要があります。SQL Anywhere では、IDENTITY カラムにどの数値データ型でも使用できます。パフォーマンスのためには、整数データ型の使用をお奨めします。

SQL Anywhere では、IDENTITY カラムとカラムのデフォルト設定の AUTOINCREMENT は同じです。

IDENTITY カラムを作成するには、次の CREATE TABLE 構文を使用します。ただし、n にはテーブルに挿入される可能性のある最大ロー数の値を保持できる大きさである必要があります。

```
CREATE TABLE table-name (
   ...
   column-name numeric(n,0) IDENTITY NOT NULL,
   ...
)
```

このセクションの内容:

@@identity を使用する IDENTITY カラム値の検索 [538 ページ]
初めてローを 1 つテーブルに挿入すると、IDENTITY カラムに 1 という値が割り当てられます。

1.4.6.8.1 @@identity を使用する IDENTITY カラム値の検索

初めてローを 1 つテーブルに挿入すると、IDENTITY カラムに 1 という値が割り当てられます。

その後は挿入するたびに、カラムの値が 1 ずつ増えます。最後に IDENTITY カラムに挿入した値は、@@identity グローバル変数として使用できます。

1.4.7 Transact-SQL 互換性のある SQL 文

Transact-SQL で動作する SQL 文を記述する際に、いくつか考慮すべき事項があります。

このセクションの内容:

移植可能な SQL を記述するための一般的なガイドライン [539 ページ]
SQL 文は、できる限り明示的にする必要があります。

Transact-SQL と互換性のあるテーブル [539 ページ]
SQL Anywhere は、制約とデフォルト定義をデータ型定義内にカプセル化できるドメインをサポートします。

Transact-SQL クエリのサポート [540 ページ]
SQL Anywhere と Adaptive Server Enterprise データベースの両方を実行するクエリを記述する場合、そのクエリのデータ型、式、検索条件、および SQL 構文に互換性がある必要があります。

ジョインの互換性 [543 ページ]
Transact-SQL の SQL Anywhere の実装では、ANSI/ISO SQL 標準からジョイン構文を指定することができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.7.1 移植可能な SQL を記述するための一般的なガイドライン

SQL 文は、できる限り明示的にする必要があります。

指定した SQL 文を複数のサーバがサポートしている場合でも、デフォルトの動作が各システムで同じであると仮定するのは間違っていることもあります。

SQL Anywhere では、データベースサーバと SQL プリプロセッサ (sqlpp) で、特定の ISO/ANSI SQL 標準に準拠しないか、Ultra Light でサポートされていない SQL 文を特定できます。この機能は SQL Flagger と呼ばれます。

次に、互換性のある SQL を記述するときの一般的なガイドラインを示します。

- デフォルトの動作を使用せずに、使用可能なオプションをすべて含めます。
- 演算子の優先順位のデフォルトが同じであるとするのではなく、文中の実行の順序をカッコを使用して明確にします。
- Adaptive Server Enterprise に移植できるように、変数名には @ をプレフィクスとして付ける Transact-SQL の規則に従います。
- プロシージャ、トリガ、バッチでは、BEGIN 文の直後で変数とカーソルを宣言します。Adaptive Server Enterprise では、プロシージャ、トリガ、バッチ内のどこででも宣言できますが、SQL Anywhere では、BEGIN 文の直後で行う必要があります。
- データベース内の識別子として、Adaptive Server Enterprise または SQL Anywhere の予約語を使用しないでください。
- 大きいネームスペースを想定します。たとえば、各インデックスにはユニークな名前を持たせるようにします。

関連情報

SQL Flagger を使用した SQL 準拠のテスト [516 ページ]

1.4.7.2 Transact-SQL と互換性のあるテーブル

SQL Anywhere は、制約とデフォルト定義をデータ型定義内にカプセル化できるドメインをサポートします。

Adaptive Server Anywhere はまた、CREATE TABLE 文の明示的なデフォルトと CHECK 条件もサポートします。ただし、名前付きデフォルトはサポートしません。

NULL

SQL Anywhere と Adaptive Server Enterprise は、NULL の処理に関して異なる点があります。Adaptive Server Enterprise では、NULL は値として処理されることがあります。

たとえば、Adaptive Server Enterprise のユニークインデックスには、NULL 値があるローと、NULL 値以外は同一のローを同時に格納できません。SQL Anywhere では、このようなローでもユニークインデックスに格納できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトでは、Adaptive Server Enterprise のカラムは NOT NULL に設定されていますが、SQL Anywhere のデフォルト設定値は NULL です。この設定は、allow_nulls_by_default オプションを使用して制御できます。NULL または NOT NULL を明示的に指定して、データ定義文を転送可能にします。

テンポラリテーブル

テンポラリテーブルを作成するには、CREATE TABLE 文のテーブル名の先頭にシャープ記号 (#) を付けます。このようなテンポラリテーブルは、SQL Anywhere の宣言されたテンポラリテーブルであり、現在の接続内でのみ使用できます。

テーブルの物理的配置は、Adaptive Server Enterprise と SQL Anywhere では実行方法が異なります。SQL Anywhere では、ON `segment-name` 句をサポートしていますが、`segment-name` は SQL Anywhere の DB 領域を意味します。

関連情報

SQL Flagger を使用した SQL 準拠のテスト [516 ページ]
Transact-SQL 互換性のオプション [534 ページ]

1.4.7.3 Transact-SQL クエリのサポート

SQL Anywhere と Adaptive Server Enterprise データベースの両方を実行するクエリを記述する場合、そのクエリのデータ型、式、検索条件、および SQL 構文に互換性がある必要があります。

データ型、式、検索条件も互換性がある必要があります。quoted_identifier オプションが OFF に設定され、それが Adaptive Server Enterprise のデフォルトの設定値であって、SQL Anywhere のデフォルトの設定値ではない、という例を想定します。

Transact-SQL ダイアレクトの SQL Anywhere の実装は Watcom SQL ダイアレクトのクエリ式構文のほとんどをサポートしていますが、Adaptive Server Enterprise はそれらの SQL 構成の一部をサポートしていません。Transact-SQL クエリで、SQL Anywhere は次の SQL 構成をサポートします。

- 識別子を指定する逆引用符 `、二重引用符 "、角カッコ []
- UNION、EXCEPT、INTERSECT クエリ式
- 派生テーブル
- テーブル関数
- 全文検索用の CONTAINS テーブル式
- REGEXP、SIMILAR、IS DISTINCT FROM、CONTAINS 述部
- ユーザ定義 SQL 関数または外部関数
- LEFT、RIGHT、FULL 外部ジョイン
- GROUP BY ROLLUP、CUBE、GROUPING SETS
- TOP N START AT M
- Window 集合関数および統計分析関数や線形回帰関数などの他の分析関数

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

サポートされている Transact-SQL クエリ構文 [541 ページ]
SQL Anywhere では、クエリの Transact-SQL 構文をサポートしています。

関連情報

SQL Flagger を使用した SQL 準拠のテスト [516 ページ]
OLAP のサポート [435 ページ]
GROUP BY と SQL/2008 標準 [363 ページ]
Transact-SQL の外部ジョイン (*= or =*) [392 ページ]

1.4.7.3.1 サポートされている Transact-SQL クエリ構文

SQL Anywhere では、クエリの Transact-SQL 構文をサポートしています。

構文

```
query-expression:
{ query-expression EXCEPT [ ALL ] query-expression
| query-expression INTERSECT [ ALL ] query-expression
| query-expression UNION [ ALL ] query-expression
| query-specification }
[ ORDER BY { expression | integer }
      [ ASC | DESC ], ... ]
[ FOR READ ONLY | for-update-clause ]
[ FOR XML xml-mode ]
```

```
query-specification:
SELECT [ ALL | DISTINCT ] [ cursor-range ] select-list
[ INTO #temporary-table-name ]
[ FROM  table-expression, ... ]
[ WHERE search-condition ]
[ GROUP BY group-by-term, ... ]
[ HAVING search-condition ]
[ WINDOW window-specification, ... ]
```

パラメータ

```
select-list:
 table-name.*
|   *
| expression
| alias-name = expression
| expression as identifier
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
| expression as string
```

```
table-expression: a valid FROM clause
```

```
group-by-term: a valid GROUP BY clause
```

```
for-update-clause: a valid FOR UPDATE or FOR READ ONLY clause
```

```
xml-mode: documented in the SELECT statement
```

```
alias-name:
identifier | 'string' | "string" | `string`
```

```
cursor-range:
{ FIRST | TOP constant-or-variable } [ START AT constant-or-variable ]
```

```
Transact-SQL-table-reference:
[ owner .]table-name [ [ AS ] correlation-name ]
[ ( INDEX index_name [ PREFETCH size ][ LRU | MRU ] ) ]
```

注記

- FROM 句の Watcom SQL 構文に加えて、SQL Anywhere は特定の Adaptive Server Enterprise テーブルヒントの Transact-SQL 構文をサポートします。テーブル参照について、Transact-SQL-table-reference は INDEX ヒントキーワードと PREFETCH、MRU、LRU キャッシュヒントをサポートします。PREFETCH、MRU、LRU は SQL Anywhere では無視されます。
- SQL Anywhere は、GROUP BY 句に含まれていないカラムの参照を可能にする GROUP BY 句への Transact-SQL 拡張をサポートしません。
  SQL Anywhere は Transact-SQL GROUP BY ALL 構成もサポートしません。
- SQL Anywhere は比較演算子 *= と =* を使用して、Transact-SQL 外部ジョイン構成のサブセットをサポートします。
- SQL Anywhere の Transact-SQL 構文は派生テーブル内に埋め込まれている場合を除き、共通テーブル式をサポートしません。このため、SQL Anywhere の Transact-SQL 構文は再帰 UNION クエリをサポートしません。この機能が必要な場合は、Use the Watcom SQL 構文を使用してください。
- テーブル仕様のパフォーマンスパラメータ部分は解析されますが、効果はありません。
- HOLDLOCK キーワードは SQL Anywhere でサポートされます。HOLDLOCK の場合、データページが不要になっても共有ロックが解放されないため、指定されたテーブルやビューの共有ロックの制限はより厳しくなります。クエリは、HOLDLOCK が指定されたテーブルで独立性レベル 3 で実行されます。
- HOLDLOCK オプションは、それが指定されたテーブルまたはビューにだけ、しかも、そのオプションが使用された文で定義されたトランザクションの間だけ適用されます。独立性レベルを 3 に設定すると、トランザクション内の各 SELECT に適用されます。1 つのクエリの中で HOLDLOCK と NOHOLDLOCK の両方のオプションは指定できません。
- NOHOLDLOCK キーワードは、SQL Anywhere によって認識されますが、効力はありません。
- Transact-SQL は SELECT 文を使用してローカル変数に値を割り当てます。

  ```
  SELECT @localvar = 42;
  ```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere でこれに対応する文は、SET 文です。

```
SET @localvar = 42;
```

- Adaptive Server Enterprise では、次のものをサポートしません。
  - SELECT...INTO `host-variable-list`
  - SELECT...INTO `variable-list`
  - EXCEPT [ALL] または INTERSECT [ALL]
  - START AT 句
  - SQL Anywhere 定義のテーブルヒント
  - テーブル関数
  - FULL OUTER JOIN
  - FOR UPDATE BY { LOCK | TIMESTAMP }
  - Window 集合関数および線形回帰関数
- SQL Anywhere は、次に示す Adaptive Server Enterprise Transact-SQL SELECT 構文のキーワードと句をサポートしません。
  - SHARED キーワード
  - PARTITION キーワード
  - COMPUTE 句
  - FOR BROWSE 句
  - GROUP BY ALL 句
  - PLAN 句
  - ISOLATION 句
- SQL Anywhere は、識別子またはエイリアスで次の文字をサポートしていません。
  - " (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

1.4.7.4 ジョインの互換性

Transact-SQL の SQL Anywhere の実装では、ANSI/ISO SQL 標準からジョイン構文を指定することができます。

キーワード JOIN、LEFT OUTER JOIN、RIGHT OUTER JOIN、FULL OUTER JOIN を使用してジョイン構文を指定できるのに加えて、文の WHERE 句で特別な比較演算子 *= と =* を使用する既存の Transact-SQL 外部ジョイン構文を指定できます。

> **注記**
>
> Transact-SQL 外部ジョイン演算子 *= と =* は旧式であるため、将来のリリースではサポートから除外されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

ジョイン: 複数テーブルからのデータ検索 [376 ページ]
Transact-SQL の外部ジョイン (*= or =*) [392 ページ]
SQL Flagger を使用した SQL 準拠のテスト [516 ページ]

1.4.8 Transact-SQL のプロシージャ言語

SQL Anywhere は、ISO/ANSI SQL 標準に基づく Watcom SQL ダイアレクトに加えて、Transact-SQL ストアドプロシージャ言語の大部分をサポートします。

このセクションの内容:

Transact-SQL のストアドプロシージャ [544 ページ]
Watcom-SQL のストアドプロシージャダイアレクトは、Transact-SQL ダイアレクトとは多くの点で異なります。

Transact-SQL のトリガ [545 ページ]
トリガの互換性を保つには、トリガ機能とトリガ構文の互換性が必要です。

Transact-SQL のバッチ [546 ページ]
Transact-SQL では、バッチは一緒に送信されてグループとして実行される一連の SQL 文です。

1.4.8.1 Transact-SQL のストアドプロシージャ

Watcom-SQL のストアドプロシージャダイアレクトは、Transact-SQL ダイアレクトとは多くの点で異なります。

SQL Anywhere のネイティブダイアレクト Watcom-SQL は、ISO/ANSI SQL 標準に基づいています。概念と機能は似ていますが、構文が異なります。概念が似ているため、Transact-SQL の SQL Anywhere でのサポートは Watcom-SQL と Transact-SQL 間の自動変換を行えます。ただし、プロシージャはどちらかの言語だけで記述する必要があり、混在させることはできません。

Transact-SQL のストアドプロシージャのサポート

ここでは、次に示す Transact-SQL ストアドプロシージャに対するサポートについて説明します。

- セミコロンを使用してプロシージャの SQL 文を区切る
- パラメータを引き渡す
- 結果セットを返す
- ステータス情報を返す
- パラメータにデフォルト値を提供する
- 制御文
- エラー処理

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ユーザ定義関数

1.4.8.2 Transact-SQL のトリガ

トリガの互換性を保つには、トリガ機能とトリガ構文の互換性が必要です。

Adaptive Server Enterprise は、文レベルの AFTER トリガをサポートしています。つまり、このトリガは、トリガを起動する文が完了してから実行されます。SQL Anywhere でサポートされる Watcom-SQL ダイアレクトは、ローレベルの BEFORE、AFTER、INSTEAD OF トリガと、文レベルの AFTER および INSTEAD OF トリガをサポートしています。

ローレベルのトリガは、Transact-SQL 互換性機能の一部ではありません。

サポートされない **Transact-SQL** トリガまたは異なる **Transact-SQL** トリガの説明

Transact-SQL トリガの機能の中で、SQL Anywhere ではサポートされない、または異なる機能を次に示します。

他のトリガを起動するトリガ

トリガが別のトリガを起動することがあります。この状況での SQL Anywhere と Adaptive Server Enterprise の対応は、やや異なります。Adaptive Server Enterprise のデフォルトでは、トリガは設定可能なネストレベル (デフォルトは 16) まで、他のトリガを起動します。ネストレベルの設定は、Adaptive Server Enterprise のネストされたトリガオプションを使用します。SQL Anywhere では、メモリが不足していないかぎり、トリガは無制限に他のトリガを起動できます。

自分自身を起動するトリガ

トリガが自分自身を起動する動作を行うことがあります。この状況での SQL Anywhere と Adaptive Server Enterprise の対応は、やや異なります。SQL Anywhere では、デフォルトで、Transact-SQL でないトリガは自分自身を再帰的に起動できます。一方、Transact-SQL ダイアレクトのトリガは自分自身を再帰的に起動することはできません。ただし、Transact-SQL ダイアレクトのトリガについては、SET 文 [T-SQL] の self_recursion オプションを使用して、トリガが自分自身を再帰的に呼び出すことを許可できます。

Adaptive Server Enterprise のデフォルトでは、トリガは自分自身を再帰的に呼び出すことはできません。再帰を許可するには、self_recursion オプションを使用します。

トリガでの **ROLLBACK** 文はサポートされない

Adaptive Server Enterprise は、トリガ中で、そのトリガを含むトランザクション全体をロールバックする ROLLBACK TRANSACTION 文を許可します。SQL Anywhere は、トリガで ROLLBACK (または ROLLBACK TRANSACTION) 文を許可しません。トリガとなる動作とそのトリガがともにアトミックな文を構成するためです。

SQL Anywhere は、Adaptive Server Enterprise と互換性のある ROLLBACK TRIGGER 文を提供します。この文は、トリガ内で動作を取り消すために使用します。

ORDER 句はサポートされない

Transact-SQL トリガでは ORDER `nn` 句を使用できません。trigger_order の値は自動的に 1 に設定されます。このため、文レベルのトリガがすでにある場合は、エラーが返され T-SQL トリガが作成されることがあります。これは、SYSTRIGGER システムテーブルには table_id、event、trigger_time、trigger_order にユニークインデックスがあるためです。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 1 つのみを設定できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

トリガ [107 ページ]
ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 [87 ページ]

1.4.8.3 Transact-SQL のバッチ

Transact-SQL では、バッチは一緒に送信されてグループとして実行される一連の SQL 文です。

バッチは SQL スクリプトファイルとして保存されます。Interactive SQL を使用してバッチ対話形式で実行できます。

プロシージャで使われる制御文はバッチでも使えます。SQL Anywhere はバッチ中の制御文の使用をサポートします。また Transact-SQL のように、デリミタのない、バッチの終わりを示す GO 文で終わる文のグループをサポートします。

Transact-SQL バッチでは、セミコロンを使用して文を区切ることもできます。

SQL スクリプトファイルに保存されているバッチでは、Interactive SQL はこれらのファイルにあるパラメータの使用をサポートします。

1.4.9 ストアドプロシージャの自動変換

SQL Anywhere は、Watcom SQL と Transact-SQL の間で文を変換する手助けをします。

次に示す SQL 言語組み込み関数は、SQL 文に関する情報を返して自動変換できるようにします。

SQLDIALECT(statement)

Watcom-SQL または Transact-SQL を返します。

WATCOM SQL(statement)

Watcom-SQL 構文を返します。

TRANSACTSQL(statement)

Transact-SQL 構文を返します。

これらは関数です。Interactive SQL の SELECT 文を使用してアクセスできます。たとえば、次の文は値 Watcom-SQL を返します。

```
SELECT SQLDIALECT( 'SELECT * FROM Employees' );
```

このセクションの内容:

ストアドプロシージャの変換 [547 ページ]
たとえば、Watcom-SQL と Transact-SQL のような SQL 構文間でストアドプロシージャを変換します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.9.1 ストアドプロシージャの変換

たとえば、Watcom-SQL と Transact-SQL のような SQL 構文間でストアドプロシージャを変換します。

前提条件

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

手順

1. SQL Central で、*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. プロシージャとファンクションフォルダをクリックし、リストからストアドプロシージャを 1 つ選択します。
3. 右ウィンドウ枠で、*[SQL]* タブをクリックして、テキストウィンドウをクリックします。
4. ファイルをクリックし、変換オプションの 1 つをクリックします。

   右ウィンドウ枠に、選択したダイアレクトでプロシージャが表示されます。選択されたダイアレクトが保存されているプロシージャと異なるときは、データベースサーバが変換を行います。変換されなかった行はコメントとして表示されます。
5. 変換されなかった行を書き直します。
6. ファイル > 保存 をクリックします。

結果

ストアドプロシージャが変換され、データベースに保存されます。

1.4.10 Transact-SQL プロシージャから返される結果セット

SQL Anywhere 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。

例

Transact-SQL プロシージャの例

次の Transact-SQL プロシージャは、Transact-SQL ストアドプロシージャが結果セットを返す方法を示します。

```
CREATE PROCEDURE ShowDepartment (@deptname VARCHAR(30))
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
AS
   SELECT Employees.Surname, Employees.GivenName
   FROM Departments, Employees
   WHERE Departments.DepartmentName = @deptname
   AND Departments.DepartmentID = Employees.DepartmentID;
```

Watcom SQL プロシージャの例

対応する SQL Anywhere のプロシージャは次のようになります。

```
CREATE PROCEDURE ShowDepartment(in deptname VARCHAR(30))
RESULT ( LastName CHAR(20), FirstName CHAR(20))
BEGIN
   SELECT Employees.Surname, Employees.GivenName
   FROM Departments, Employees
   WHERE Departments.DepartmentName = deptname
   AND Departments.DepartmentID = Employees.DepartmentID
END;
```

関連情報

結果セット [128 ページ]

1.4.11 Transact-SQL プロシージャの中の変数

SQL Anywhere は SET 文を使用して、プロシージャ内の変数に値を割り当てます。

Transact-SQL では、空のテーブルリストの SELECT 文、または SET 文を使用して値を割り当てます。次のプロシージャは、Transact-SQL 構文の働きを示します。

```
CREATE PROCEDURE multiply
               @mult1 int,
               @mult2 int,
               @result int output
AS
SELECT @result = @mult1 * @mult2;
```

このプロシージャを呼び出すには、次のようにします。

```
CREATE VARIABLE @product int
go
EXECUTE multiply 5, 6, @product OUTPUT
go
```

変数 @product の値は、プロシージャの実行後 30 になります。

関連情報

Transact-SQL クエリのサポート [540 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.4.12 Transact-SQL プロシージャでのエラー処理

デフォルトでのプロシージャのエラー処理は、Watcom SQL と Transact-SQL とでは違います。

Watcom SQL のデフォルトでは、エラーが起こった場合にプロシージャは終了し、呼び出した環境に SQLSTATE 値と SQLCODE 値を返します。

EXCEPTION 文を使って、Watcom SQL ストアドプロシージャに明示的なエラー処理を組み込むことができます。または、ON EXCEPTION RESUME 文を使って、エラーが起こった次の文から実行を再開するよう、プロシージャに指示することもできます。

Transact-SQL のプロシージャでエラーが起こった場合は、次の文から実行が継続されます。最後に実行された文のエラーステータスは、グローバル変数 @@error に保存されます。文の後ろにあるこの変数をチェックして、プロシージャから強制的に返すことができます。たとえば、次の文は、エラーが起こると終了させます。

```
IF @@error != 0 RETURN
```

プロシージャが実行を終了したときの戻り値で、プロシージャが成功したかどうかがわかります。この戻り値は整数で、次のように指定してアクセスできます。

```
DECLARE @Status INT
EXECUTE @Status = proc_sample
IF @Status = 0
   PRINT 'procedure succeeded'
ELSE
   PRINT 'procedure failed'
```

次のテーブルは、組み込みプロシージャの戻り値とその意味を示します。

値	定義	SQL Anywhere SQLSTATE
0	プロシージャはエラーを起こすことなく実行された	
-1	オブジェクトが見つからない	42W33、52W02、52003、52W07、42W05
-2	データ型エラー	53018
-3	プロセスはデッドロックの対象となった	40001、40W06
-4	パーミッションエラー	42501
-5	構文エラー	42W04
-6	その他のユーザエラー	
-7	リソースエラー (たとえば領域が足りない)	08W26
-10	致命的な内部の矛盾	40W01
-11	致命的な内部の矛盾	40000
-13	データベースが壊れている	WI004
-14	ハードウェアエラー	08W17、40W03、40W04

SQL Anywhere SQLSTATE が該当しない場合、デフォルト値の -6 が返されます。

RETURN 文は、この表の値以外の、ユーザが意味を定義した整数値を返すこともできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

RAISERROR 文を使用するプロシージャ [550 ページ]
RAISERROR 文を使用してユーザ定義エラーを作成します。

Watcom SQL での Transact-SQL のようなエラー処理 [550 ページ]
CREATE PROCEDURE 文に ON EXCEPTION RESUME 句を追加して、Watcom SQL のプロシージャが Transact-SQL に似た方法でエラー処理を行うようにすることができます。

1.4.12.1 RAISERROR 文を使用するプロシージャ

RAISERROR 文を使用してユーザ定義エラーを作成します。

RAISERROR 文は、SIGNAL 文と同じように機能します。

RAISERROR 文そのものは、プロシージャを終了しません。ただし、ユーザ定義エラー後の実行を制御するために、RETURN 文やグローバル変数 @@error のテストと組み合わせることができます。

on_tsql_error データベースオプションを Continue に設定すると、RAISERROR 文は実行終了時にエラーを通知しません。代わりに、プロシージャが完了し、RAISERROR のステータスコードとメッセージを保存してから、最新の RAISERROR を返します。RAISERROR を返したプロシージャが他のプロシージャから呼び出された場合、最も外側のプロシージャが終了してから RAISERROR が返されます。on_tsql_error オプションをデフォルト値 (Conditional) に設定した場合、continue_after_raiserror オプションは、RAISERROR 文の実行後の動作を制御します。on_tsql_error オプションを Stop または Continue に設定した場合、その設定は continue_after_raiserror の設定より優先されます。

中間レベルの RAISERROR ステータスとコードは、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 ととも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。

1.4.12.2 Watcom SQL での Transact-SQL のようなエラー処理

CREATE PROCEDURE 文に ON EXCEPTION RESUME 句を追加して、Watcom SQL のプロシージャが Transact-SQL に似た方法でエラー処理を行うようにすることができます。

```
CREATE PROCEDURE sample_proc()
ON EXCEPTION RESUME
BEGIN
   ...
END
```

ON EXCEPTION RESUME 句があると、明示的な例外処理コードは実行されません。このため、明示的なエラー処理ではこの句を避けます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5 データベースにおける XML

Extensible Markup Language (XML) は、構造化データをテキスト形式で表します。XML は、大規模な電子出版の課題を満たすために設計されました。

XML は、HTML のように単純なマークアップ言語ですが、SGML のように柔軟性があります。XML は、階層型で、その主な目的は、人間とコンピュータの両方が作成し、読むことのできるデータの構造を記述することです。

XML では、さまざまな形式のデータを記述する、一連の静的な要素は提供されておらず、ユーザが要素を定義できます。そのため、XML を使用して多くの種類の構造化データを記述できます。XML 文書では、オプションとして文書型定義 (DTD) または XML スキーマを使用し、XML ファイルで使用される構造、要素、属性を定義できます。

SQL Anywhere で XML を使用する方法はいくつかあります。

- データベースに XML 文書を格納する
- リレーショナルデータを XML としてエクスポートする
- データベースに XML をインポートする
- リレーショナルデータを XML として問い合わせる

このセクションの内容:

リレーショナルデータベースにおける XML 文書の格納 [552 ページ]
データベースで XML 文書を格納するために使用できるデータ型には、XML データ型と LONG VARCHAR データ型の 2 つのタイプがあります。

XML としてエクスポートされるリレーショナルデータ [553 ページ]
リレーショナルデータを XML としてエクスポートするために、Interactive SQL OUTPUT 文と ADO.NET DataSet オブジェクトの 2 つの方法を提供しています。

XML 文書をリレーショナルデータとしてインポートする方法 [554 ページ]
XML をデータベースにインポートするには、2 種類の方法があります。

XML としてのクエリ結果 [561 ページ]
リレーショナルデータからクエリ結果を XML として取得するための 2 種類の方法が提供されています。

結果を表示するための Interactive SQL の使用 [579 ページ]
FOR XML クエリの結果は文字列で返されます。

クエリ結果を XML として取得するための SQL/XML の使用 [580 ページ]
SQL/XML は、XML を SQL 言語に機能統合する方法を定める、ドラフト段階の標準です。SQL/XML は、XML とともに SQL を使用する方法を定めています。

関連情報

Extensible Markup Language (XML)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5.1 リレーショナルデータベースにおける XML 文書の格納

データベースで XML 文書を格納するために使用できるデータ型には、XML データ型と LONG VARCHAR データ型の 2 つのタイプがあります。

これらのデータ型は、いずれも XML 文書を文字列としてデータベースに格納します。

XML データ型は、データベースサーバの文字セットエンコードを使用します。XML エンコード属性は、データベースサーバが使用するエンコードと一致する必要があります。XML エンコード属性は、自動文字セット変換の実行方法を指定しません。

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。

リレーショナルデータから要素を生成するとき、XML で無効な文字は、データが XML 型でないかぎりエスケープされます。たとえば、次の内容の <product> という要素を生成するとします。この要素の内容には、不等号 (より小、より大) が含まれています。

```
<hat>bowler</hat>
```

要素の内容を XML 型として指定するクエリを記述した場合、次のように不等号はマークアップされません。

```
SELECT XMLFOREST( CAST( '<hat>bowler</hat>' AS XML ) AS product );
```

結果は次のようになります。

```
<product><hat>bowler</hat></product>
```

しかし、たとえば次のように、クエリでこの要素の内容を XML 型として指定しない場合があります。

```
SELECT XMLFOREST( '<hat>bowler</hat>' AS product );
```

この場合、不等号がエンティティの参照で置換されます。

```
<product>&lt;hat&gt;bowler&lt;/hat&gt;</product>
```

属性は、データ型に関係なく常にマークアップされます。

関連情報

不正な XML 名のエンコーディングのルール [564 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5.2 XML としてエクスポートされるリレーショナルデータ

リレーショナルデータを XML としてエクスポートするために、Interactive SQL OUTPUT 文と ADO.NET DataSet オブジェクトの 2 つの方法を提供しています。

FOR XML 句と SQL/XML 関数を使用して、データベースのリレーショナルデータから結果セットを XML として生成できます。次に、UNLOAD 文または xp_write_file システムプロシージャを使用して、生成された XML をファイルにエクスポートできます。

このセクションの内容:

Interactive SQL から XML としてエクスポートされるリレーショナルデータ [553 ページ]
Interactive SQL OUTPUT 文は、XML フォーマットをサポートしており、クエリ結果を生成された XML ファイルに出力します。

DataSet オブジェクトを使用して XML としてエクスポートされるリレーショナルデータ [553 ページ]
ADO.NET DataSet オブジェクトを使用して、DataSet の内容を XML 文書に保存できます。

1.5.2.1 Interactive SQL から XML としてエクスポートされるリレーショナルデータ

Interactive SQL OUTPUT 文は、XML フォーマットをサポートしており、クエリ結果を生成された XML ファイルに出力します。

この生成された XML ファイルは、UTF-8 でエンコードされており、埋め込み DTD が含まれます。XML ファイルでは、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として文字データ (CDATA) ブロック内にエンコードされます。

INPUT 文は、XML をファイルフォーマットとして受け入れません。ただし、OPENXML 演算子または ADO.NET DataSet オブジェクトを使用して XML をインポートできます。

関連情報

XML 文書をリレーショナルデータとしてインポートする方法 [554 ページ]

1.5.2.2 DataSet オブジェクトを使用して XML としてエクスポートされるリレーショナルデータ

ADO.NET DataSet オブジェクトを使用して、DataSet の内容を XML 文書に保存できます。

一度、データベースのクエリ結果などを DataSet に入力すると、DataSet からスキーマのみか、スキーマとデータの両方を XML ファイルに保存できます。WriteXml メソッドは、スキーマとデータの両方を XML ファイルに保存します。WriteXmlSchema メソッドは、スキーマのみを XML ファイルに保存します。SQL Anywhere .NET データプロバイダを使用して DataSet オブジェクトに入力することが可能です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5.3 XML 文書をリレーショナルデータとしてインポートする方法

XML をデータベースにインポートするには、2 種類の方法があります。

- OPENXML 演算子を使用して、XML 文書から結果セットを生成します
- ADO.NET DataSet オブジェクトを使用して、XML 文書から DataSet にデータかスキーマまたはその両方を読み込みます

このセクションの内容:

OPENXML 演算子を使用した XML のインポート [554 ページ]
: クエリの FROM 句で OPENXML 演算子を使用すると、XML 文書から結果セットを生成できます。

DataSet オブジェクトを使用した XML のインポート [560 ページ]
: ADO.NET DataSet オブジェクトを使用して、XML 文書から DataSet にデータかスキーマまたはその両方を読み込むことができます。

デフォルトの XML ネームスペースの定義 [560 ページ]
: デフォルトのネームスペースは、`xmlns="URI"` の形式の属性で、XML 文書の要素に定義します。

1.5.3.1 OPENXML 演算子を使用した XML のインポート

クエリの FROM 句で OPENXML 演算子を使用すると、XML 文書から結果セットを生成できます。

OPENXML は、XPath クエリ言語のサブセットを使用して、XML 文書からノードを選択します。

XPath 式の使用

OPENXML を使用すると、XML 文書が解析され、結果はツリーとしてモデル化されます。このツリーはノードで構成されています。XPath 式は、ツリー内のノードを選択するために使用されます。次のリストは、一般的に使用される XPath 式の一部を示しています。

/

XML 文書のルートノードを示します。

//

ルートのすべての子孫を示します。ルートノードも含まれます。

. (単一のピリオド)

XML 文書のカレントノードを示します。

.//

カレントノードのすべての子孫を示します。カレントノードも含まれます。

..

カレントノードの親ノードを示します。

./@attributename

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

attributename という名前を持つ、カレントノードの属性を示します。

./childname

カレントノードの子で、childname という名前を持つ要素を示します。

次の XML 文書を考えてみます。

```
<inventory>
  <product ID="301" size="Medium">Tee Shirt
    <quantity>54</quantity>
  </product>
  <product ID="302" size="One Size fits all">Tee Shirt
    <quantity>75</quantity>
  </product>
  <product ID="400" size="One Size fits all">Baseball Cap
    <quantity>112</quantity>
  </product>
</inventory>
```

<inventory> 要素は、ルートノードです。この要素は、次の XPath 式を使用して参照できます。

```
/inventory
```

カレントノードが <quantity> 要素であると仮定します。このノードは、次の XPath 式を使用して参照できます。

```
.
```

<inventory> 要素の子である <product> 要素をすべて検出するには、次の XPath 式を使用します。

```
/inventory/product
```

カレントノードが <product> 要素のときに、size 属性を参照したい場合は、次の XPath 式を使用します。

```
./@size
```

OPENXML を使用した結果セットの生成

OPENXML の最初の xpath-query 引数に一致するごとに、結果セットにローが 1 つ生成されます。WITH 句は、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します。次のクエリを例にとります。

```
SELECT * FROM OPENXML(
'<inventory>
   <product>Tee Shirt
     <quantity>54</quantity>
     <color>Orange</color>
   </product>
   <product>Baseball Cap
     <quantity>112</quantity>
     <color>Black</color>
   </product>
</inventory>',
'/inventory/product' )
WITH ( Name CHAR (25) './text()',
       Quantity CHAR(3) 'quantity',
       Color CHAR(20) 'color');
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

最初の xpath-query 引数は、/inventory/product です。そして XML には <product> 要素が 2 つあるため、このクエリによってローが 2 つ生成されます。

WITH 句は、カラムが Name、Quantity、Color の 3 つであることを指定します。これらのカラムの値は、<product>、<quantity>、<color> の各要素から取得されます。前述のクエリは、次の結果を生成します。

Name	Quantity	Color
Tee Shirt	54	Orange
Baseball Cap	112	Black

OPENXML を使用したエッジテーブルの生成

OPENXML 演算子を使用すると、XML 文書内の各要素に対応する各行から成るエッジテーブルを生成できます。エッジテーブルを生成すると、SQL を使用して結果セット内のデータを問い合わせできます。

次の SQL 文は、XML 文書を 1 つ含むテーブルを作成します。このクエリが生成する XML には、<root> と呼ばれるルート要素があります。このルート要素は、XMLELEMENT 関数を使用して生成されています。また、ELEMENTS 修飾子を持つ FOR XML AUTO を使用して、Employees テーブル、SalesOrders テーブル、Customers テーブルの指定された各カラムに対応する要素が生成されています。

```
CREATE TABLE IF NOT EXISTS xmldata (xmldoc XML);
INSERT INTO xmldata WITH AUTO NAME
    SELECT XMLELEMENT( NAME root,
         (SELECT EmployeeID, Employees.GivenName, Employees.Surname,
           Customers.ID, Customers.GivenName, Customers.Surname, Customers.Phone,
CompanyName,
           SalesOrders.ID, OrderDate, Region
         FROM Employees
         KEY JOIN SalesOrders
         KEY JOIN Customers
         ORDER BY EmployeeID, Customers.ID, SalesOrders.ID
         FOR XML AUTO, ELEMENTS)) AS xmldoc;
SELECT xmldoc FROM xmldata;
```

生成される XML ロックは、次のようになります (結果は読みやすいようにフォーマットされています。クエリから返される結果は 1 つの連続した文字列です)。

```
<root>
  <Employees>
    <EmployeeID>129</EmployeeID>
    <GivenName>Philip</GivenName>
    <Surname>Chin</Surname>

    <Customers>
      <ID>101</ID>
      <GivenName>Michaels</GivenName>
      <Surname>Devlin</Surname>
      <Phone>2015558966</Phone>
      <CompanyName>The Power Group</CompanyName>
        <SalesOrders>
          <ID>2560</ID>
          <OrderDate>2001-03-16</OrderDate>
          <Region>Eastern</Region>
        </SalesOrders>
    </Customers>
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    <Customers>
      <ID>103</ID>
      <GivenName>Erin</GivenName>
      <Surname>Niedringhaus</Surname>
      <Phone>2155556513</Phone>
      <CompanyName>Darling Associates</CompanyName>
        <SalesOrders>
          <ID>2451</ID>
          <OrderDate>2000-12-15</OrderDate>
          <Region>Eastern</Region>
        </SalesOrders>
    </Customers>

    <Customers>
      <ID>104</ID>
      <GivenName>Meghan</GivenName>
      <Surname>Mason</Surname>
      <Phone>6155555463</Phone>
      <CompanyName>P.S.C.</CompanyName>
        <SalesOrders>
          <ID>2331</ID>
          <OrderDate>2000-09-17</OrderDate>
          <Region>South</Region>
        </SalesOrders>

        <SalesOrders>
          <ID>2342</ID>
          <OrderDate>2000-09-28</OrderDate>
          <Region>South</Region>
        </SalesOrders>
    </Customers>
    ...
  </Employees>
  ...
  <Employees>
  ...
  </Employees>
</root>
```

次のクエリは、descendant-or-self (//*) XPath 式を使用して、前述の XML 文書内の各要素とのマッチングを行っています。次に、各要素に対し id メタプロパティを使用して、ノードの ID を取得しています。また、parent (../) XPath 式を id メタプロパティとともに使用して、親ノードを取得しています。localname メタプロパティは、各要素の名前を取得するために使用されています。メタプロパティ名では大文字と小文字が区別されます。そのため、ID や LOCALNAME はメタプロパティ名として使用できません。

```
CREATE OR REPLACE VARIABLE x XML;
SELECT xmldoc INTO x FROM xmldata;
SELECT *
FROM OPENXML( x, '//*' )
WITH (ID INT '@mp:id',
       parent INT '../@mp:id',
       name CHAR(25) '@mp:localname',
       text LONG VARCHAR 'text()' )
ORDER BY ID;
```

このクエリによって生成される結果セットには、XML 文書内の各ノードの ID、親ノードの ID、各要素の名前と内容が表示されています。

ID	parent	name	text
5	(NULL)	root	(NULL)
16	5	Employees	(NULL)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ID	parent	name	text
28	16	EmployeeID	129
55	16	GivenName	Phillip
82	16	Surname	Chin
...	...	...	...

xp_read_file での OPENXML の使用

これまで XMLELEMENT のようなプロシージャで生成された XML を使用してきましたが、xp_read_file プロシージャを使用して、ファイルからの XML を読み込んで解析することもできます。次のクエリを使用してファイル `c:¥temp¥inventory.xml` が書き込まれたとします。

```
SELECT xp_write_file( 'c:¥¥temp¥¥inventory.xml',
'<inventory>
   <product>Tee Shirt
      <quantity>54</quantity>
      <color>Orange</color>
   </product>
   <product>Baseball Cap
      <quantity>112</quantity>
      <color>Black</color>
   </product>
</inventory>'
);
```

この場合、次の文を使用してファイル内の XML を読み込み、解析できます。

```
SELECT *
FROM OPENXML( xp_read_file( 'c:¥¥temp¥¥inventory.xml' ),
  '//*' )
WITH (ID INT '@mp:id',
       parent INT '../@mp:id',
       name CHAR(128) '@mp:localname',
       text LONG VARCHAR 'text()' )
ORDER BY ID;
```

カラム内の XML の問い合わせ

XML を含むカラムを持つテーブルがある場合、OPENXML を使用して、カラム内のすべての XML 値を一度に問い合わせできます。これには、ラテラル派生テーブルを使用します。

次の文は、ManagerID と Reports という 2 つのカラムを持つテーブルを作成します。Reports カラムには、Employees テーブルから生成された XML データが含まれます。

```
CREATE TABLE IF NOT EXISTS xmltest (ManagerID INT, Reports XML);
INSERT INTO xmltest
  SELECT ManagerID, XMLELEMENT( NAME reports,
    XMLAGG( XMLELEMENT( NAME e, EmployeeID)))
  FROM Employees
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
  GROUP BY ManagerID;
```

次のクエリを実行して、テストテーブル内のデータを表示してください。

```
SELECT *
FROM xmltest
ORDER BY ManagerID;
```

このクエリは、次の結果を生成します。

<table>
<tr><th>ManagerID</th><th>Reports</th></tr>
<tr><td>501</td><td><pre><reports>
 <e>102</e>
 <e>105</e>
 <e>160</e>
 <e>243</e>
 ...
</reports></pre></td></tr>
<tr><td>703</td><td><pre><reports>
 <e>191</e>
 <e>750</e>
 <e>868</e>
 <e>921</e>
 ...
</reports></pre></td></tr>
<tr><td>902</td><td><pre><reports>
 <e>129</e>
 <e>195</e>
 <e>299</e>
 <e>467</e>
 ...
</reports></pre></td></tr>
<tr><td>1293</td><td><pre><reports>
 <e>148</e>
 <e>390</e>
 <e>586</e>
 <e>757</e>
 ...
</reports></pre></td></tr>
<tr><td>...</td><td>...</td></tr>
</table>

次のクエリは、ラテラル派生テーブルを使用して、2 つのカラムを持つ結果セットを生成しています。1 つのカラムは、各マネージャの ID をリストします。もう 1 つのカラムは、そのマネージャに報告を行う各従業員の ID をリストします。

```
SELECT ManagerID, EmployeeID
FROM xmltest, LATERAL( OPENXML( xmltest.Reports, '//e' )
WITH (EmployeeID INT '.') ) DerivedTable
ORDER BY ManagerID, EmployeeID;
```

このクエリは、次の結果を生成します。

 © 2016 SAP SE or an SAP affiliate company. All rights reserved.

ManagerID	EmployeeID
501	102
501	105
501	160
501	243
...	...

関連情報

XML Path 言語 (XPath)
FOR XML AUTO [568 ページ]

1.5.3.2 DataSet オブジェクトを使用した XML のインポート

ADO.NET DataSet オブジェクトを使用して、XML 文書から DataSet にデータかスキーマまたはその両方を読み込むことができます。

- ReadXml メソッドは、スキーマとデータの両方を含む XML 文書から DataSet に投入を行います。
- ReadXmlSchema メソッドは、XML 文書からスキーマのみを読み込みます。一度 DataSet に XML 文書のデータが入力されると、DataSet からの変更に基づいてデータベース内のテーブルを更新できます。

また、SQL Anywhere .NET データプロバイダを使用して、DataSet オブジェクトを操作できます。

1.5.3.3 デフォルトの XML ネームスペースの定義

デフォルトのネームスペースは、`xmlns="URI"` の形式の属性で、XML 文書の要素に定義します。

次の例では、文書には `http://www.sap.com/EmployeeDemo` という URI にバインドされるデフォルトのネームスペースがあります。

```
<x xmlns="http://www.sap.com/EmployeeDemo"/>
```

要素の名前にプレフィクスがない場合は、その要素およびその要素のすべての子孫要素にデフォルトのネームスペースが適用されます。コロンによって、プレフィクスと残りの要素名は区切られます。たとえば、<x/> にはプレフィクスがありませんが、<p:x/> にはプレフィクス p があります。`xmlns:prefix="URI"` の形式の属性によって、プレフィクスにバインドされるネームスペースを定義します。次の例では、文書がプレフィクス p を前述の例と同じ URI にバインドします。

```
<x xmlns:p="http://www.sap.com/EmployeeDemo"/>
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトのネームスペースが属性に適用されることはありません。プレフィクスがある場合を除き、属性は常に NULL ネームスペース URI にバインドされます。次の例では、ルート要素と子要素には iAnywhere1 ネームスペースがあり、x 属性には NULL ネームスペース URI、y 属性には iAnywhere2 ネームスペースがあります。

```
<root xmlns="iAnywhere1" xmlns:p="iAnywhere2">
<child x='1' p:y='2' />
</root>
```

XML 文書を、OPENXML クエリの namespace-declaration 引数として渡すと、文書のルート要素に定義されたネームスペースはクエリに適用されます。ルート要素以降の残りの文書はすべて無視されます。次の例では、p1 は文書では iAnywhere1 にバインドされ、namespace-declaration 引数では p2 にバインドされます。クエリはプレフィクス p2 を使用できます。

```
SELECT *
FROM OPENXML('<p1:x xmlns:p1="iAnywhere1">123</p1:x>', '/p2:x', 1, '<root
xmlns:p2="iAnywhere1"/>')
WITH ( c1 int '.' );
```

要素を一致させる場合、プレフィクスがバインドされる URI を正確に指定する必要があります。上の例では、xpath クエリの x 名と文書の x 要素は、どちらも iAnywhere1 ネームスペースがあるため、一致となります。

要素を一致させる場合、プレフィクスがバインドされる URI を正確に指定する必要があります。上の例では、xpath クエリの x 名と文書の x 要素は、どちらも iAnywhere1 ネームスペースがあるため、一致となります。xpath 要素 x のプレフィクスは、xml-data 内で x 要素に対して定義されたネームスペースと一致する、namespace-declaration 内で定義されたネームスペース iAnywhere1 を参照します。

OPENXML 演算子の namespace-declaration では、デフォルトのネームスペースを使用しないでください。NULL ネームスペースを含む任意の URI にバインドされる x 要素と一致する、/*:x 形式のワイルドカードクエリを使用してください。または、特定のプレフィクスに必要な URI をバインドし、それをクエリで使用します。

1.5.4 XML としてのクエリ結果

リレーショナルデータからクエリ結果を XML として取得するための 2 種類の方法が提供されています。

FOR XML 句

FOR XML 句を SELECT 文で使用して、XML 文書を生成できます。

SQL/XML

SQL Anywhere は、リレーショナルデータから XML 文書を生成する、ドラフト段階の SQL/XML 標準に基づく関数をサポートしています。

SQL Anywhere がサポートする FOR XML 句と SQL/XML 関数により、リレーショナルデータから XML を生成する 2 つの選択肢が提供されます。通常は、どちらかの方法を使用して同じ XML を生成できます。

たとえば、このクエリは、FOR XML AUTO を使用して XML を生成しています。

```
SELECT ID, Name
FROM Products
WHERE Color='black'
FOR XML AUTO;
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のクエリは、XMLELEMENT 関数を使用して XML を生成しています。

```
SELECT XMLELEMENT(NAME product,
          XMLATTRIBUTES(ID, Name))
FROM Products
WHERE Color='black';
```

どちらのクエリも次の XML を生成します (結果セットは読みやすいようにフォーマットされています)。

```
<product ID="302" Name="Tee Shirt"/>
<product ID="400" Name="Baseball Cap"/>
<product ID="501" Name="Visor"/>
<product ID="700" Name="Shorts"/>
```

ヒント

ネストの深い文書を生成する場合は、FOR XML EXPLICIT クエリの方が SQL/XML クエリよりも効率的である可能性が高くなります。これは、EXPLICIT モードクエリは、通常 UNION を使用してネストを生成するのに対し、SQL/XML はサブクエリを使用して必要なネストを生成するためです。

このセクションの内容:

クエリ結果を XML として取り出すための FOR XML 句の使用 [562 ページ]
SELECT 文内で FOR XML 句を使用して、データベースに対し SQL クエリを実行し、結果を XML 文書として返すことができます。

FOR XML RAW [566 ページ]
クエリ内で FOR XML RAW を指定すると、各ローは、<row> 要素として表され、各カラムは、<row> 要素の属性となります。

FOR XML AUTO [568 ページ]
AUTO モードは、XML 文書内にネストされた要素を生成します。

FOR XML EXPLICIT [570 ページ]
FOR XML EXPLICIT 句を使用して、クエリが返す XML 文書の構造を制御できます。

関連情報

クエリ結果を XML として取得するための SQL/XML の使用 [580 ページ]

1.5.4.1 クエリ結果を XML として取り出すための FOR XML 句の使用

SELECT 文内で FOR XML 句を使用して、データベースに対し SQL クエリを実行し、結果を XML 文書として返すことができます。

XML 文書は、XML 型です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

FOR XML 句は、サブクエリ、GROUP BY 句または集合関数のあるクエリ、ビュー定義など、どのような SELECT 文内でも使用できます。

SQL Anywhere は、FOR XML 句を使用して生成された XML 文書に対しスキーマは生成しません。

FOR XML 句内で、生成される XML のフォーマットを制御する 3 つの XML モードのうちの 1 つを指定できます。

RAW

クエリに一致する各ローを <row> XML 要素として、各カラムを属性として表します。

AUTO

クエリ結果をネストされた XML 要素として返します。SELECT リスト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、SELECT リスト内のカラムの順序に基づきます。

EXPLICIT

希望するネストに関する情報を含むクエリを記述できます。そのため、生成される XML の形式を制御できます。

以下の項では、FOR XML 句の 3 つのモードに共通する、バイナリデータ、NULL 値、無効な XML 名に関連する動作について説明します。また、FOR XML 句の使用例も示します。

このセクションの内容:

FOR XML とバイナリデータ [564 ページ]
SELECT 文で FOR XML 句では、使用するモードに関わらず、BINARY、LONG BINARY、IMAGE、または VARBINARY カラムが属性または要素として出力されます。

FOR XML と NULL 値 [564 ページ]
デフォルトでは、NULL 値を含む要素と属性は、結果セットから省略されます。for_xml_null_treatment オプションを使用すると、この動作を制御できます。

不正な XML 名のエンコーディングのルール [564 ページ]
XML 名として有効ではない名前 (たとえば、スペースを含むカラム名) をエンコードするためのルールがあります。

FOR XML の例 [565 ページ]
SELECT 文内での FOR XML 句の使用方法の例をいくつか示します。

関連情報

FOR XML RAW [566 ページ]
FOR XML AUTO [568 ページ]
FOR XML EXPLICIT [570 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5.4.1.1 FOR XML とバイナリデータ

SELECT 文で FOR XML 句では、使用するモードに関わらず、BINARY、LONG BINARY、IMAGE、または VARBINARY カラムが属性または要素として出力されます。

SELECT 文で FOR XML 句を使用すると、使用するモードに関わらず、すべての BINARY、LONG BINARY、IMAGE、または VARBINARY カラムは、自動的に Base64 エンコードフォーマットで表される属性または要素として出力されます。

OPENXML を使用して XML から結果セットを生成する場合、OPENXML は、BINARY、LONG BINARY、IMAGE、VARBINARY の各データ型を Base64 でエンコードされていると見なし、自動的にデコードします。

1.5.4.1.2 FOR XML と NULL 値

デフォルトでは、NULL 値を含む要素と属性は、結果セットから省略されます。for_xml_null_treatment オプションを使用すると、この動作を制御できます。

NULL 会社名を含む Customers テーブル内のエントリを考えてみます。

```
INSERT INTO Customers( ID, Surname, GivenName, Street, City, Phone)
VALUES (100,'Robert','Michael', '100 Anywhere Lane','Smallville','519-555-3344');
```

for_xml_null_treatment オプションを Omit (デフォルト) に設定して次のクエリを実行すると、属性は NULL カラム値について生成されません。

```
SELECT ID, GivenName, Surname, CompanyName
FROM Customers
WHERE GivenName LIKE 'Michael%'
ORDER BY ID
FOR XML RAW;
```

この場合、CompanyName 属性は Michael Robert について生成されません。

```
<row ID="100" GivenName="Michael" Surname="Robert"/>
<row ID="101" GivenName="Michaels" Surname="Devlin" CompanyName="The Power Group"/>
<row ID="110" GivenName="Michael" Surname="Agliori" CompanyName="The Pep Squad"/>
```

for_xml_null_treatment オプションを Empty に設定すると、空の属性も結果に含まれます。

```
<row ID="100" GivenName="Michael" Surname="Robert" CompanyName=""/>
<row ID="101" GivenName="Michaels" Surname="Devlin" CompanyName="The Power Group"/>
<row ID="110" GivenName="Michael" Surname="Agliori" CompanyName="The Pep Squad"/>
```

この場合、空の CompanyName 属性が Michael Robert について生成されています。

1.5.4.1.3 不正な XML 名のエンコーディングのルール

XML 名として有効ではない名前 (たとえば、スペースを含むカラム名) をエンコードするためのルールがあります。

XML には、SQL 名のルールとは異なる名前のルールがあります。たとえば XML 名にはスペースを使用できません。カラム名などの SQL 名が XML 名に変換されると、XML 名で有効でない文字はエンコードされるかエスケープされます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

エンコードされた各文字について、エンコーディングは文字の Unicode のコードポイント値が基になり、16 進数で表されます。

- ほとんどの文字のコードポイント値は、16 ビット、または 4 桁の 16 進数で表すことができ、_xHHHH_ というエンコーディングが使用されます。このような文字に対応する Unicode 文字の UTF-16 値は、16 ビットワード 1 つです。
- コードポイント値が 16 ビットよりも多く必要な文字では、8 桁の 16 進数が使用され、エンコーディングは _xHHHHHHHH_ になります。このような文字に対応する Unicode 文字の UTF-16 値は、16 ビットワード 2 つです。ただし、エンコーディングには UTF-16 値ではなく、Unicode のコードポイント値 (通常は 16 進数で 5 または 6 桁) が使用されます。
たとえば、次のクエリには、スペースを持つカラム名が含まれています。

```
SELECT EmployeeID AS "Employee ID"
FROM Employees
FOR XML RAW;
```

そのため、次の結果が返されます。

```
<row Employee_x0020_ID="102"/>
<row Employee_x0020_ID="105"/>
<row Employee_x0020_ID="129"/>
<row Employee_x0020_ID="148"/>
...
```

- アンダースコア (_) は、次に文字 x が続く場合、エスケープされます。たとえば、名前 Linu_x は Linu_x005F_x のようにエンコーディングされます。
- コロン (:) は、エスケープされません。そのため、FOR XML クエリを使用して、名前空間宣言と修飾された要素名、属性名を生成できます。

ヒント

Interactive SQL で FOR XML 句を含むクエリを実行する場合は、truncation_length オプションを設定するとカラム長を増やせます。

1.5.4.1.4 FOR XML の例

SELECT 文内での FOR XML 句の使用方法の例をいくつか示します。

- 次の例は、サブクエリ内での FOR XML 句の使用方法を示します。

```
SELECT XMLELEMENT( NAME root,
    (SELECT * FROM Employees FOR XML RAW) );
```

- 次の例は、GROUP BY 句と集合関数のあるクエリ内での FOR XML 句の使用方法を示します。

```
SELECT Name, AVG(UnitPrice) AS Price
FROM Products
GROUP BY Name
FOR XML RAW;
```

- 次の例は、ビュー定義内での FOR XML 句の使用方法を示します。

```
CREATE VIEW EmployeesDepartments
AS SELECT Surname, GivenName, DepartmentName
FROM Employees JOIN Departments
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
ON Employees.DepartmentID = Departments.DepartmentID
FOR XML AUTO;
```

1.5.4.2 FOR XML RAW

クエリ内で FOR XML RAW を指定すると、各ローは、<row> 要素として表され、各カラムは、<row> 要素の属性となります。

構文

FOR XML RAW [, *ELEMENTS*]

パラメータ

ELEMENTS

このパラメータを指定すると、FOR XML RAW は、結果における各カラムに対し属性の代わりに XML 要素を生成します。NULL 値がある場合は、その要素は、生成される XML 文書から省略されます。次のクエリは、<EmployeeID> 要素と <DepartmentName> 要素を生成します。

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
   ON Employees.DepartmentID=Departments.DepartmentID
FOR XML RAW, ELEMENTS;
```

このクエリは、次の結果を返します。

```
<row>
   <EmployeeID>102</EmployeeID>
   <DepartmentName>R &amp; D</DepartmentName>
</row>
<row>
   <EmployeeID>105</EmployeeID>
   <DepartmentName>R &amp; D</DepartmentName>
</row>
<row>
   <EmployeeID>160</EmployeeID>
   <DepartmentName>R &amp; D</DepartmentName>
</row>
<row>
   <EmployeeID>243</EmployeeID>
   <DepartmentName>R &amp; D</DepartmentName>
</row>
...
```

使用法

BINARY、LONG BINARY、IMAGE、VARBINARY カラムのデータは、FOR XML RAW を含むクエリを実行すると、自動的に Base64 エンコードフォーマットで返されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトでは、NULL 値は、結果から省略されます。for_xml_null_treatment オプションを使用すると、この動作を制御できます。

FOR XML RAW は、整形式 XML 文書を返しません。これは、文書に単一のルートノードが含まれないためです。<root> 要素が必要な場合は、1 つの方法として、XMLELEMENT 関数を使用して挿入できます。次に例を示します。

```
SELECT XMLELEMENT( NAME root,
                   (SELECT EmployeeID AS id, GivenName AS name
                   FROM Employees FOR XML RAW) );
```

XML 文書内で使用される属性名や要素名は、エイリアスを指定して変更できます。次のクエリは、ID 属性の名前を product_ID に変更します。

```
SELECT ID AS product_ID
FROM Products
WHERE Color='black'
FOR XML RAW;
```

このクエリは、次の結果を返します。

```
<row product_ID="302"/>
<row product_ID="400"/>
<row product_ID="501"/>
<row product_ID="700"/>
```

結果の順序は、特に指定しないかぎり、オプティマイザが選択するプランによって決まります。特定の順序で結果を表示したい場合は、次のように、クエリに ORDER BY 句を含めてください。

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
   ON Employees.DepartmentID=Departments.DepartmentID
ORDER BY EmployeeID
FOR XML RAW;
```

例

従業員が所属する部署の情報を取り出したい場合、次のように入力します。

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
   ON Employees.DepartmentID=Departments.DepartmentID
FOR XML RAW;
```

次の XML 文書が返されます。

```
<row EmployeeID="102" DepartmentName="R &amp; D"/>
<row EmployeeID="105" DepartmentName="R &amp; D"/>
<row EmployeeID="160" DepartmentName="R &amp; D"/>
<row EmployeeID="243" DepartmentName="R &amp; D"/>
...
```

関連情報

FOR XML と NULL 値 [564 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5.4.3 FOR XML AUTO

AUTO モードは、XML 文書内にネストされた要素を生成します。

ELEMENTS 句を省略すると、SELECT リスト内で参照される各テーブルは、生成された XML 内で要素として表されます。ネストの順序は、SELECT リスト内でカラムが参照される順序に基づきます。SELECT リスト内の各カラムに対して属性が作成されます。

ELEMENTS 句があると、SELECT リスト内で参照される各テーブルおよびカラムは、生成された XML 内で要素として表されます。ネストの順序は、SELECT リスト内でカラムが参照される順序に基づきます。SELECT リスト内の各カラムに対して要素が作成されます。

構文

```
FOR XML AUTO [, ELEMENTS ]
```

パラメータ

ELEMENTS

このパラメータを指定すると、FOR XML AUTO は、結果における各カラムに対し属性の代わりに XML 要素を生成します。次に例を示します。

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
   ON Employees.DepartmentID=Departments.DepartmentID
ORDER BY EmployeeID
FOR XML AUTO, ELEMENTS;
```

この場合、結果セット内の各カラムは、<Employees> 要素または <Departments> 要素の属性としてではなく、別個の要素として返されています。NULL 値がある場合は、その要素は、生成される XML 文書から省略されます。

```
<Employees>
   <EmployeeID>102</EmployeeID>
   <Departments>
      <DepartmentName>R &amp; D</DepartmentName>
   </Departments>
</Employees>
<Employees>
   <EmployeeID>105</EmployeeID>
   <Departments>
      <DepartmentName>R &amp; D</DepartmentName>
   </Departments>
</Employees>
<Employees>
   <EmployeeID>129</EmployeeID>
   <Departments>
   <DepartmentName>Sales</DepartmentName>
   </Departments>
</Employees>
...
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

使用法

FOR XML AUTO を使用してクエリを実行すると、BINARY、LONG BINARY、IMAGE、VARBINARY カラムのデータは、自動的に Base64 エンコードフォーマットで返されます。デフォルトでは、NULL 値は、結果から省略されます。for_xml_null_treatment オプションを EMPTY に設定すると、NULL 値を空の属性として返すことができます。

特に指定しないかぎり、データベースサーバは、テーブルのローを意味のない順序で返します。特定の順序で結果を表示したい場合、または親要素に複数の子を持たせたい場合は、クエリに ORDER BY 句を含めて、すべての子が隣接するようにします。ORDER BY 句を指定しないと、結果のネストはオプティマイザが選択するプランによって決まり、必要なネストが得られないことがあります。

FOR XML AUTO は、整形式 XML 文書を返しません。これは、文書に単一のルートノードが含まれないためです。<root> 要素が必要な場合は、1 つの方法として、XMLELEMENT 関数を使用して挿入できます。次に例を示します。

```
SELECT XMLELEMENT( NAME root,
                   (SELECT EmployeeID AS id, GivenName AS name
                   FROM Employees FOR XML AUTO ) );
```

XML 文書内で使用される属性名や要素名は、エイリアスを指定して変更できます。次のクエリは、ID 属性の名前を product_ID に変更します。

```
SELECT ID AS product_ID
FROM Products
WHERE Color='Black'
FOR XML AUTO;
```

次の XML が生成されます。

```
<Products product_ID="302"/>
<Products product_ID="400"/>
<Products product_ID="501"/>
<Products product_ID="700"/>
```

テーブルの名前をエイリアスに変更することもできます。次のクエリは、テーブルを product_info に名前を変更します。

```
SELECT ID AS product_ID
FROM Products AS product_info
WHERE Color='Black'
FOR XML AUTO;
```

次の XML が生成されます。

```
<product_info product_ID="302"/>
<product_info product_ID="400"/>
<product_info product_ID="501"/>
<product_info product_ID="700"/>
```

例

次のクエリは、<employee> 要素と <department> 要素の両方を含む XML を生成します。<employee> 要素 (SELECT リストで最初にリストされたテーブル) は、<department> 要素の親です。

```
SELECT EmployeeID, DepartmentName
FROM Employees AS employee
JOIN Departments AS department
   ON employee.DepartmentID=department.DepartmentID
ORDER BY EmployeeID
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FOR XML AUTO;
```

前述のクエリによって、次の XML が生成されます。

```
<employee EmployeeID="102">
   <department DepartmentName="R &amp; D"/>
</employee>
<employee EmployeeID="105">
   <department DepartmentName="R &amp; D"/>
</employee>
<employee EmployeeID="129">
   <department DepartmentName="Sales;"/>
</employee>
<employee EmployeeID="148">
   <department DepartmentName="Finance;"/>
</employee>
...
```

次のように、SELECT リスト内でカラムの順序を変更するとします。

```
SELECT DepartmentName, EmployeeID
FROM Employees AS employee JOIN Departments AS department
   ON employee.DepartmentID=department.DepartmentID
ORDER BY 1, 2
FOR XML AUTO;
```

結果は次のようにネストされます。

```
<department DepartmentName="Finance">
   <employee EmployeeID="148"/>
   <employee EmployeeID="390"/>
   <employee EmployeeID="586"/>
   ...
</department>
<department DepartmentName="Marketing">
   <employee EmployeeID="184"/>
   <employee EmployeeID="207"/>
   <employee EmployeeID="318"/>
   ...
</department>
...
```

ここでも、クエリによって生成された XML には、<employee> 要素と <department> 要素の両方が含まれています。しかしこの場合は、<department> 要素が <employee> 要素の親となっています。

1.5.4.4 FOR XML EXPLICIT

FOR XML EXPLICIT 句を使用して、クエリが返す XML 文書の構造を制御できます。

クエリは特定の方法で記述して、必要なネストに関する情報がクエリ結果内で指定されるようにしてください。FOR XML EXPLICIT がサポートするオプションのディレクティブを使用すると、個別のカラムの扱いを設定できます。たとえば、あるカラムが要素内容と属性内容のどちらとして表示されるかを制御できます。また、あるカラムが生成された XML に含まれるのではなく、結果の順序付けのみに使用されるように制御できます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

パラメータ

EXPLICIT モードでは、SELECT 文の最初の 2 つのカラムに、それぞれ Tag と Parent と名前を付けてください。Tag と Parent はメタデータカラムで、それらの値は、クエリが返す XML 文書内の要素の親子関係、またはネストを決定するために使用されます。

Tag カラム

これは、SELECT リスト内で最初に指定されるカラムです。Tag カラムは、現在の要素のタグ番号を格納します。タグ番号として許可されている値は、1 から 255 までです。

Parent カラム

このカラムは、現在の要素の親のタグ番号を格納します。このカラムの値が NULL の場合、そのローは XML 階層のトップレベルに位置付けられています。

たとえば、FOR XML EXPLICIT が指定されていない場合に、次の結果セットを返すクエリを考えてみます

Tag	Parent	GivenName!1	ID!2
1	NULL	'Beth'	NULL
2	NULL	NULL	'102'

この例では、Tag カラムの値は、結果セット内の各要素のタグ番号です。両方のローの Parent カラムには、値 NULL が含まれています。両要素とも階層のトップレベルに生成され、クエリに FOR XML EXPLICIT 句が含まれる場合は、次の結果が得られます。

```
<GivenName>Beth</GivenName>
<ID>102</ID>
```

しかし、2 番目のローの Parent カラムが値 1 を持つ場合は、結果は次のようになります。

```
<GivenName>Beth
   <ID>102</ID>
</GivenName>
```

クエリへのデータカラムの追加

Tag カラムと Parent カラムに加えて、クエリには、1 つ以上のデータカラムを含めてください。これらのデータカラムの名前は、タグ付け中にカラムが解釈される方法を制御します。各カラム名は、感嘆符 (!) で区切られるフィールドに分割されます。次のフィールドをデータカラムに指定できます。

```
ElementName!TagNumber!AttributeName!Directive
```

ElementName

要素の名前。ある特定のローに関して、ローに対して生成される要素名は、一致するタグ番号を持つ最初のカラムの `ElementName` フィールドから取得されます。同じ `TagNumber` を持つ複数のカラムがある場合は、`ElementName` は、同じ `TagNumber` を持つ後続のカラムについては無視されます。前述の例では、最初のローは、<GivenName> と呼ばれる要素を生成します。

TagNumber

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

要素のタグ番号。ある特定のタグ値を持つローに関して、`TagNumber` フィールドに同じ値を持つすべてのカラムは、そのローに対応する要素に内容を提供します。

AttributeName

カラム値が `ElementName` 要素の属性であることを指定します。たとえば、データカラムが productID!1!Color という名前の場合、Color は <productID> 要素の属性として表示されます。

Directive

このオプションフィールドを使用して、XML 文書のフォーマットをさらに制御できます。`Directive` に対して次の値のいずれか 1 つを指定できます。

hide

結果を生成するときにこのカラムが無視されることを示します。このディレクティブは、テーブルを順序付ける目的のみに使用されるカラムを含めるために使用できます。属性名は無視され、結果には含まれません。

element

カラム値が、属性としてではなく、名前 `AttributeName` を持つ、ネストされた要素として挿入されることを示します。

xml

カラム値が、引用されずに挿入されることを示します。`AttributeName` が指定されている場合は、値はその名前を持つ要素として挿入されます。それ以外の場合は、値は、要素がラップされずに挿入されます。このディレクティブが使用されていない場合は、カラムが XML 型でないかぎり、マークアップ文字でエスケープされます。たとえば、値 <a/> は、<a/> として挿入されます。

cdata

カラム値が CDATA セクションとして挿入されることを示します。`AttributeName` は無視されます。

使用方法

BINARY、LONG BINARY、IMAGE、VARBINARY カラムのデータは、FOR XML EXPLICIT を含むクエリを実行すると、自動的に Base64 エンコードフォーマットで返されます。デフォルトでは、結果セット内のすべての NULL 値は省略されます。for_xml_null_treatment オプションの設定を変更すると、この動作を変更できます。

EXPLICIT モードのクエリの記述

次の XML 文書を生成するクエリを FOR XML EXPLICIT を使用して記述するとします。

```
<employee employeeID='129'>
   <customer customerID='107' region='Eastern'/>
   <customer customerID='119' region='Western'/>
   <customer customerID='131' region='Eastern'/>
</employee>
<employee employeeID='195'>
   <customer customerID='109' region='Eastern'/>
   <customer customerID='121' region='Central'/>
</employee>
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このためには、次の結果セットを指定された順序どおりに返す SELECT 文を記述し、クエリに FOR XML EXPLICIT を追加します。

Tag	Parent	employee!1!employeeID	customer!2!customerID	customer!2!region
1	NULL	129	NULL	NULL
2	1	129	107	Eastern
2	1	129	119	Western
2	1	129	131	Central
1	NULL	195	NULL	NULL
2	1	195	109	Eastern
2	1	195	121	Central

クエリを記述すると、ある特定のローの一部のカラムのみが、生成された XML 文書の一部となります。カラムは、`TagNumber` フィールド (カラム名の 2 つ目のフィールド) の値が、Tag カラムの値と一致する場合のみ、XML 文書に含められます。

この例では、Tag カラムに値 1 を持つ 2 つのローの場合に 3 番目のカラムが使用されます。4 番目と 5 番目のカラムでは、Tag カラムに値 2 を持つローの場合に値が使用されます。要素名は、カラム名の最初のフィールドから取得されます。この例の場合、<employee> 要素と <customer> 要素が作成されます。

属性名は、カラム名の 3 番目のフィールドから取得されます。したがって、<employee> 要素に対して EmployeeID 属性が作成され、<customer> 要素に対して CustomerID 属性と Region 属性が作成されます。

次の手順では、サンプルデータベースを使用して、前述の最初にある XML 文書に似た XML 文書を生成する FOR XML EXPLICIT クエリを構成する方法を説明します。

例

1. トップレベルの要素を生成する SELECT 文を記述します。
   この例では、クエリの最初の SELECT 文は、<employee> 要素を生成します。クエリの最初の 2 つの値は、Tag カラム値と Parent カラム値にしてください。<employee> 要素は、階層のトップにあるため、Tag 値に 1 を、Parent 値に NULL を割り当ててください。

   注記

   UNION を使用する EXPLICIT モードのクエリを記述する場合、最初の SELECT 文で指定されたカラム名のみが使用されます。要素名または属性名として使用するカラム名は、最初の SELECT 文で指定してください。これは、後続の SELECT 文で指定されたカラム名は無視されるためです。

2. 前述のテーブルの <employee> 要素を生成するためには、最初の SELECT 文は、次のようになります。

```
SELECT
      1          AS tag,
      NULL       AS parent,
      EmployeeID AS [employee!1!employeeID],
      NULL       AS [customer!2!customerID],
      NULL       AS [customer!2!region]
FROM Employees;
```

3. 子要素を生成する SELECT 文を記述します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2 つめのクエリは、<customer> 要素を生成します。これは EXPLICIT モードクエリのため、すべての SELECT 文において、最初の 2 つの値に Tag 値と Parent 値を指定してください。<customer> 要素には、タグ番号 2 を与えます。また、この要素は <employee> 要素の子であるため、Parent 値は 1 になります。最初の SELECT 文で、すでに EmployeeID、CustomerID、Region は属性であると指定しています。

```
SELECT
      2,
      1,
      EmployeeID,
      CustomerID,
      Region
FROM Employees KEY JOIN SalesOrders
```

4. クエリに UNION DISTINCT を追加して、2 つの SELECT 文を結合します。

```
SELECT
      1          AS tag,
      NULL       AS parent,
      EmployeeID AS [employee!1!employeeID],
      NULL       AS [customer!2!customerID],
      NULL       AS [customer!2!region]
FROM Employees
UNION DISTINCT
SELECT
      2,
      1,
      EmployeeID,
      CustomerID,
      Region
FROM Employees KEY JOIN SalesOrders
```

5. ORDER BY 句を追加して、結果内のローの順序を指定します。ローの順序は、生成される文書内で使用される順序です。

```
SELECT
      1          AS tag,
      NULL       AS parent,
      EmployeeID AS [employee!1!employeeID],
      NULL       AS [customer!2!customerID],
      NULL       AS [customer!2!region]
FROM Employees
UNION DISTINCT
SELECT
      2,
      1,
      EmployeeID,
      CustomerID,
      Region
FROM Employees KEY JOIN SalesOrders
ORDER BY 3, 1
FOR XML EXPLICIT;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

FOR XML EXPLICIT の例

次のクエリ例は、従業員による発注に関する情報を取り出します。この例では、<employee>、<order>、<department> という 3 種類の要素があります。<employee> 要素は ID 属性と name 属性を持ち、<order> 要素は date 属性を、また <department> 要素は name 属性を持ちます。

```
SELECT
        1           tag,
        NULL        parent,
        EmployeeID  [employee!1!id],
        GivenName   [employee!1!name],
        NULL        [order!2!date],
        NULL        [department!3!name]
FROM Employees
UNION DISTINCT
SELECT
        2,
        1,
        EmployeeID,
        NULL,
        OrderDate,
        NULL
FROM Employees KEY JOIN SalesOrders
UNION DISTINCT
SELECT
        3,
        1,
        EmployeeID,
        NULL,
        NULL,
        DepartmentName
FROM Employees e JOIN Departments d
   ON e.DepartmentID=d.DepartmentID
ORDER BY 3, 1
FOR XML EXPLICIT;
```

このクエリから次の結果が得られます。

```
<employee id="102" name="Fran">
   <department name="R &amp; D"/>
</employee>
<employee id="105" name="Matthew">
   <department name="R &amp; D"/>
</employee>
<employee id="129" name="Philip">
   <order date="2000-07-24"/>
   <order date="2000-07-13"/>
   <order date="2000-06-24"/>
   <order date="2000-06-08"/>
   ...
   <department name="Sales"/>
</employee>
<employee id="148" name="Julie">
   <department name="Finance"/>
</employee>
...
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

element ディレクティブの使用

属性ではなくサブ要素を生成する場合は、次のように、クエリに element ディレクティブを追加します。

```
SELECT
        1          tag,
        NULL       parent,
        EmployeeID [employee!1!id!element],
        GivenName  [employee!1!name!element],
        NULL       [order!2!date!element],
        NULL       [department!3!name!element]
FROM Employees
UNION DISTINCT
SELECT
        2,
        1,
        EmployeeID,
        NULL,
        OrderDate,
        NULL
FROM Employees KEY JOIN SalesOrders
UNION DISTINCT
SELECT
        3,
        1,
        EmployeeID,
        NULL,
        NULL,
        DepartmentName
FROM Employees e JOIN Departments d
   ON e.DepartmentID=d.DepartmentID
ORDER BY 3, 1
FOR XML EXPLICIT;
```

このクエリから次の結果が得られます。

```
<employee>
   <id>102</id>
   <name>Fran</name>
   <department>
      <name>R &amp; D</name>
   </department>
</employee>
<employee>
   <id>105</id>
   <name>Matthew</name>
   <department>
      <name>R &amp; D</name>
   </department>
</employee>
<employee>
   <id>129</id>
   <name>Philip</name>
   <order>
      <date>2000-07-24</date>
   </order>
   <order>
      <date>2000-07-13</date>
   </order>
   <order>
      <date>2000-06-24</date>
   </order>
   ...
   <department>
      <name>Sales</name>
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
   </department>
</employee>
...
```

hide ディレクティブの使用

次のクエリでは、employee ID は、結果の順序付けに使用されていますが、hide ディレクティブが指定されているため、結果には表示されません。

```
SELECT
        1           tag,
        NULL        parent,
        EmployeeID  [employee!1!id!hide],
        GivenName   [employee!1!name],
        NULL        [order!2!date],
        NULL        [department!3!name]
FROM Employees
UNION DISTINCT
SELECT
        2,
        1,
        EmployeeID,
        NULL,
        OrderDate,
        NULL
FROM Employees KEY JOIN SalesOrders
UNION DISTINCT
SELECT
        3,
        1,
        EmployeeID,
        NULL,
        NULL,
        DepartmentName
FROM Employees e JOIN Departments d
   ON e.DepartmentID=d.DepartmentID
ORDER BY 3, 1
FOR XML EXPLICIT;
```

このクエリは、次の結果を返します。

```
<employee name="Fran">
   <department name="R &amp; D"/>
</employee>
<employee name="Matthew">
   <department name="R &amp; D"/>
</employee>
<employee name="Philip">
   <order date="2000-04-21"/>
   <order date="2001-07-23"/>
   <order date="2000-12-30"/>
   <order date="2000-12-20"/>
   ...
   <department name="Sales"/>
</employee>
<employee name="Julie">
   <department name="Finance"/>
</employee>
...
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

xml ディレクティブの使用

デフォルトでは、FOR XML EXPLICIT クエリの結果に XML 文字として有効ではない文字が含まれる場合、カラムが XML 型でないかぎり、無効な文字はエスケープされます。

たとえば、次のクエリは、アンパサンド (&) を含む XML を生成します。

```
SELECT
        1               AS tag,
        NULL            AS parent,
        ID              AS [customer!1!id!element],
        CompanyName     AS [customer!1!company!element]
FROM Customers
WHERE ID = '115'
FOR XML EXPLICIT;
```

このクエリによって生成される結果では、アンパサンドはエスケープされます。これは、このカラムが XML 型ではないためです。

```
<customer><id>115</id>
<company>Sterling &amp; Co.</company>
</customer>
```

xml ディレクティブは、生成される XML にカラム値がエスケープされずに挿入されることを示します。前述のクエリに xml ディレクティブを付けて実行します。

```
SELECT
        1               AS tag,
        NULL            AS parent,
        ID              AS [customer!1!id!element],
        CompanyName     AS [customer!1!company!xml]
FROM Customers
WHERE ID = '115'
FOR XML EXPLICIT;
```

結果内で、アンパサンドはエスケープされていません。

```
<customer>
 <id>115</id>
 <company>Sterling & Co.</company>
</customer>
```

この XML は、アンパサンドが含まれるため、整形式ではありません。アンパサンドは、XML においては特別な文字です。クエリを使用して XML を生成する場合は、その XML が整形式であり、妥当であることを確認してください。SQL Anywhere は、生成される XML が整形式または妥当であるかをチェックしません。

xml ディレクティブを指定すると、AttributeName フィールドは属性ではなく要素を生成するために使用されます。

cdata ディレクティブの使用

次のクエリは、cdata ディレクティブを使用して、製品名を CDATA セクションに入れて返します。

```
SELECT
        1                AS tag,
        NULL             AS parent,
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
        ID                AS [product!1!id],
        Description       AS [product!1!!cdata]
FROM Products
FOR XML EXPLICIT;
```

このクエリによって生成される結果は、各製品の説明を CDATA セクション内にリストします。CDATA セクションに含まれるデータは、引用されません。

```
<product id="300">
   Tank Top
</product>
<product id="301">
   V-neck
</product>
<product id="302">
   Crew Neck
</product>
<product id="400">
   Cotton Cap
</product>
...
```

関連情報

FOR XML と NULL 値 [564 ページ]

不正な XML 名のエンコーディングのルール [564 ページ]

1.5.5 結果を表示するための Interactive SQL の使用

FOR XML クエリの結果は文字列で返されます。

多くの場合、文字列の結果は長くなります。Interactive SQL には、*[ウィンドウで表示]* オプションを使用して整形式 XML 文書の構造を表示する機能があります。

FOR XML クエリの結果は、<?xml?> タグを含め、任意でタグのペアで囲んで (<root>...</root> など)、整形式 XML 文書にキャストできます。次のクエリは、これを行う方法を示します。

```
SELECT XMLCONCAT( CAST('<?xml version="1.0"?>' AS XML),
  XMLELEMENT( NAME root, (
    SELECT
          1          AS tag,
          NULL       AS parent,
          EmployeeID AS [employee!1!employeeID],
          NULL       AS [customer!2!customerID],
          NULL       AS [customer!2!region],
          NULL       AS [custname!3!given_name!element],
          NULL       AS [custname!3!surname!element]
    FROM Employees
    UNION DISTINCT
    SELECT
          2,
          1,
          EmployeeID,
          CustomerID,
          Region,
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
            NULL,
            NULL
     FROM Employees KEY JOIN SalesOrders
     UNION DISTINCT
     SELECT
            3,
            2,
            EmployeeID,
            CustomerID,
            NULL,
            Customers.GivenName,
            Customers.SurName
     FROM SalesOrders
     JOIN Customers
         ON SalesOrders.CustomerID = Customers.ID
     JOIN Employees
         ON SalesOrders.SalesRepresentative = Employees.EmployeeID
     ORDER BY 3, 4, 1
     FOR XML EXPLICIT
   ) )
);
```

Interactive SQL のカラムの *[トランケーションの長さ]* の値は、カラム全体をフェッチできる大きさに設定する必要があります。これは、**ツール** › **オプション** メニューを使用するか、次のような Interactive SQL 文を実行して行うことができます。

```
SET OPTION truncation_length = 80000;
```

XML 文書の結果を表示するには、**結果** ウィンドウ枠でカラムの内容をダブルクリックし、*XML* **アウトライン** タブを選択します。

1.5.6 クエリ結果を XML として取得するための SQL/XML の使用

SQL/XML は、XML を SQL 言語に機能統合する方法を定める、ドラフト段階の標準です。SQL/XML は、XML とともに SQL を使用する方法を定めています。

サポートされる関数を使用して、リレーショナルデータから XML 文書を構成するクエリを記述できます。

無効な名前と SQL/XML

SQL/XML では、スペースを含む式など、有効な XML 名でない式は、FOR XML 句と同様にエスケープされます。XML 型の要素の内容は、引用されません。

このセクションの内容:

XMLAGG 関数の使用 [581 ページ]
XMLAGG は、集合関数で、クエリ内のすべてのローに対して単一の集約された XML 結果を生成します。

XMLCONCAT 関数の使用 [582 ページ]
XMLCONCAT 関数は、渡されるすべての XML 値を連結して、XML 要素のフォレストを作成します。

XMLELEMENT 関数の使用 [582 ページ]
XMLELEMENT 関数は、リレーショナルデータから XML 要素を構成します。

XMLFOREST 関数の使用 [585 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

XMLFOREST 関数は、XML 要素のフォレストを構成します。

XMLGEN 関数の使用 [586 ページ]

XMLGEN 関数は、XQuery コンストラクタに基づいて XML 値を生成するために使用されます。

関連情報

リレーショナルデータベースにおける XML 文書の格納 [552 ページ]

不正な XML 名のエンコーディングのルール [564 ページ]

1.5.6.1 XMLAGG 関数の使用

XMLAGG は、集合関数で、クエリ内のすべてのローに対して単一の集約された XML 結果を生成します。

XMLAGG 関数は、XML 要素の集合から XML 要素のフォレストを生成するために使用されます。

次のクエリでは、XMLAGG は、各ローに対し <name> 要素を生成するために使用されています。<name> 要素は、従業員名で順序付けされています。ORDER BY 句は、XML 要素を順序付けるために指定されています。

```
SELECT XMLELEMENT( NAME Departments,
                   XMLATTRIBUTES ( DepartmentID ),
                   XMLAGG( XMLELEMENT( NAME name,
                                       Surname )
                                       ORDER BY Surname )
                 ) AS department_list
FROM Employees
GROUP BY DepartmentID
ORDER BY DepartmentID;
```

このクエリは、次の結果を生成します。

department_list
`<Departments DepartmentID="100"> <name>Breault</name> <name>Cobb</name> <name>Diaz</name> <name>Driscoll</name> ... </Departments>`
`<Departments DepartmentID="200"> <name>Chao</name> <name>Chin</name> <name>Clark</name> <name>Dill</name> ... </Departments>`

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

department_list

`<Departments DepartmentID="300">`
` <name>Bigelow</name>`
` <name>Coe</name>`
` <name>Coleman</name>`
` <name>Davidson</name>`
` ...`
`</Departments>`

...

1.5.6.2 XMLCONCAT 関数の使用

XMLCONCAT 関数は、渡されるすべての XML 値を連結して、XML 要素のフォレストを作成します。

たとえば、次のクエリは、Employees テーブルの従業員ごとに、<given_name> 要素と <surname> 要素を連結します。

```
SELECT XMLCONCAT( XMLELEMENT( NAME given_name, GivenName ),
                  XMLELEMENT( NAME surname, Surname )
                 ) AS "Employee_Name"
FROM Employees;
```

このクエリは、次の結果を返します。

Employee_Name
`<given_name>Fran</given_name>` `<surname>Whitney</surname>`
`<given_name>Matthew</given_name>` `<surname>Cobb</surname>`
`<given_name>Philip</given_name>` `<surname>Chin</surname>`
`<given_name>Julie</given_name>` `<surname>Jordan</surname>`
...

1.5.6.3 XMLELEMENT 関数の使用

XMLELEMENT 関数は、リレーショナルデータから XML 要素を構成します。

生成される要素の内容を指定できます。また、その要素の属性と、属性の内容も指定できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ネストされた要素の生成

次のクエリは、ネストされた XML を生成します。製品ごとに <product_info> 要素が生成され、その中に各製品の名前、数量、説明を示す要素が生成されます。

```
SELECT ID,
XMLELEMENT( NAME product_info,
            XMLELEMENT( NAME item_name, Products.name ),
            XMLELEMENT( NAME quantity_left, Products.Quantity ),
            XMLELEMENT( NAME description, Products.Size || ' ' ||
                        Products.Color || ' ' || Products.name )
          ) AS results
FROM Products
WHERE Quantity > 30;
```

このクエリは、次の結果を生成します。

ID	results

301	`<product_info>`
	` <item_name>Tee Shirt`
	`  </item_name>`
	` <quantity_left>54`
	`  </quantity_left>`
	` <description>Medium Orange`
	`  Tee Shirt</description>`
	`</product_info>`

302	`<product_info>`
	` <item_name>Tee Shirt`
	`  </item_name>`
	` <quantity_left>75`
	`  </quantity_left>`
	` <description>One Size fits`
	`  all Black Tee Shirt`
	`  </description>`
	`</product_info>`

400	`<product_info>`
	` <item_name>Baseball Cap`
	`  </item_name>`
	` <quantity_left>112`
	`  </quantity_left>`
	` <description>One Size fits`
	`  all Black Baseball Cap`
	`  </description>`
	`</product_info>`

...	...

要素の内容の指定

XMLELEMENT 関数を使用して、要素の内容を指定できます。次の文は、内容 `hat` を持つ XML 要素を生成します。

```
SELECT ID, XMLELEMENT( NAME product_type, 'hat' )
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM Products
WHERE Name IN ( 'Baseball Cap', 'Visor' );
```

属性を持つ要素の生成

クエリに XMLATTRIBUTES 引数を含めると、要素に属性を追加できます。この引数は、属性名と内容を指定します。次の文は、各品目の name、Color、UnitPrice に対して属性を生成します。

```
SELECT ID, XMLELEMENT( NAME item_description,
                       XMLATTRIBUTES( Name,
                                      Color,
                                      UnitPrice )
                     ) AS item_description_element
FROM Products
WHERE ID > 400;
```

AS 句を指定して、属性に名前を付けることができます。

```
SELECT ID, XMLELEMENT( NAME item_description,
                       XMLATTRIBUTES ( Color AS color,
                                      UnitPrice AS price ),
                       Products.Name
                    ) AS products
FROM Products
WHERE ID > 400;
```

例

次の例では HTTP Web サービスで XMLELEMENT を使用します。

```
CREATE OR REPLACE PROCEDURE "DBA"."http_header_example_with_table_proc"()
RESULT ( res LONG VARCHAR )
BEGIN
    DECLARE var LONG VARCHAR;
    DECLARE varval LONG VARCHAR;
    DECLARE I INT;
    DECLARE res LONG VARCHAR;
    DECLARE htmltable XML;
    SET var  = NULL;
loop_h:
    LOOP
        SET var = NEXT_HTTP_HEADER( var );
        IF var IS NULL THEN LEAVE loop_h END IF;
        SET varval = http_header( var );
        -- ... do some action for <var,varval> pair...
        SET htmltable = htmltable ||
            XMLELEMENT( name "tr",
            XMLATTRIBUTES( 'left' AS "align", 'top' AS "valign" ),
            XMLELEMENT( name "td", var ),
            XMLELEMENT( name "td", varval ) )  ;
    END LOOP;
    SET res = XMLELEMENT( NAME "table",
        XMLATTRIBUTES( '' AS "BORDER", '10' as "CELLPADDING", '0' AS
"CELLSPACING" ),
        XMLELEMENT( NAME "th",
            XMLATTRIBUTES( 'left' AS "align", 'top' AS "valign" ),
            'Header Name' ),

        XMLELEMENT( NAME "th",
             XMLATTRIBUTES( 'left' AS "align", 'top' AS "valign" ),
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
                'Header Value' ),

          htmltable);
     SELECT res;
END;
```

1.5.6.4 XMLFOREST 関数の使用

XMLFOREST 関数は、XML 要素のフォレストを構成します。

各 XMLFOREST 引数に対して、1 つの要素が生成されます。

次のクエリは、<item_description> 要素を生成します。この要素には、<name>、<color>、<price> 要素があります。

```
SELECT ID, XMLELEMENT( NAME item_description,
                       XMLFOREST( Name as name,
                                  Color as color,
                                  UnitPrice AS price )
                      ) AS product_info
FROM Products
WHERE ID > 400;
```

このクエリによって、次の結果が生成されます。

<table>
<tr><th>ID</th><th>product_info</th></tr>
<tr><td>401</td><td><pre><item_description>
 <name>Baseball Cap</name>
 <color>White</color>
 <price>10.00</price>
</item_description></pre></td></tr>
<tr><td>500</td><td><pre><item_description>
 <name>Visor</name>
 <color>White</color>
 <price>7.00</price>
</item_description></pre></td></tr>
<tr><td>501</td><td><pre><item_description>
 <name>Visor</name>
 <color>Black</color>
 <price>7.00</price>
</item_description></pre></td></tr>
<tr><td>...</td><td>...</td></tr>
</table>

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.5.6.5 XMLGEN 関数の使用

XMLGEN 関数は、XQuery コンストラクタに基づいて XML 値を生成するために使用されます。

次のクエリによって生成される XML は、サンプルデータベース内の顧客の注文に関する情報を提供します。このクエリでは、次の変数参照を使用します。

{$ID}

SalesOrders テーブルの ID カラムの値を使用して、<ID> 要素の内容を生成します。

{$OrderDate}

SalesOrders テーブルの OrderDate カラムの値を使用して、<date> 要素の内容を生成します。

{$Customers}

Customers テーブルの CompanyName カラムから <customer> 要素の内容を生成します。

```
SELECT XMLGEN ( '<order>
              <ID>{$ID}</ID>
              <date>{$OrderDate}</date>
              <customer>{$Customers}</customer>
              </order>',
              SalesOrders.ID,
              SalesOrders.OrderDate,
              Customers.CompanyName AS Customers
              ) AS order_info
FROM SalesOrders JOIN Customers
ON Customers.ID = SalesOrders.CustomerID
ORDER BY SalesOrders.CustomerID;
```

このクエリは、次の結果を生成します。

order_info
`<order> <ID>2001</ID> <date>2000-03-16</date> <customer>The Power Group</customer> </order>`
`<order> <ID>2005</ID> <date>2001-03-26</date> <customer>The Power Group</customer> </order>`
`<order> <ID>2125</ID> <date>2001-06-24</date> <customer>The Power Group</customer> </order>`

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

order_info
<order> <ID>2206</ID> <date>2000-04-16</date> <customer>The Power Group</customer> </order>
...

属性の生成

注文 ID 番号を <order> 要素の属性としたい場合は、次のようにクエリを記述します (変数参照が二重引用符で囲まれています。これは、属性値を指定しているためです)。

```
SELECT XMLGEN ( '<order ID="{$ID}">
                 <date>{$OrderDate}</date>
                 <customer>{$Customers}</customer>
                 </order>',
                SalesOrders.ID,
                SalesOrders.OrderDate,
                Customers.CompanyName AS Customers
              ) AS order_info
FROM SalesOrders JOIN Customers
ON Customers.ID = SalesOrders.CustomerID
ORDER BY SalesOrders.OrderDate;
```

このクエリは、次の結果を生成します。

order_info
<order ID="2131"> <date>2000-01-02</date> <customer>BoSox Club</customer> </order>
<order ID="2065"> <date>2000-01-03</date> <customer>Bloomfield's</customer> </order>
<order ID="2126"> <date>2000-01-03</date> <customer>Leisure Time</customer> </order>
<order ID="2127"> <date>2000-01-06</date> <customer>Creative Customs Inc.</customer> </order>
...

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

両方の結果セットにおいて、顧客名 Bloomfield's は、Bloomfield's と引用されています。これは、アポストロフィは XML において特別な文字であり、<customer> 要素が生成される元となったカラムは XML 型ではないためです。

XML 文書のヘッダ情報の指定

SQL Anywhere がサポートする FOR XML 句と SQL/XML 関数は、生成する XML 文書にバージョン宣言情報を含めません。XMLGEN 関数を使用すると、ヘッダ情報を生成できます。

```
SELECT XMLGEN( '<?xml version="1.0"
                encoding="ISO-8859-1" ?>
                <r>{$x}</r>',
                (SELECT GivenName, Surname
      FROM Customers FOR XML RAW) AS x );
```

これは、次の結果を生成します。

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<r>
 <row GivenName="Michaels" Surname="Devlin"/>
 <row GivenName="Beth" Surname="Reiser"/>
 <row GivenName="Erin" Surname="Niedringhaus"/>
 <row GivenName="Meghan" Surname="Mason"/>
 ...
</r>
```

1.6 データベース内の JSON

JavaScript Object Notation (JSON) は、言語に依存しないテキストベースのデータ交換フォーマットで、JavaScript データの直列化のために開発されました。

JSON には、次の 4 つの基本型があります。文字列、数値、ブール、および NULL です。また、JSON では、オブジェクトと配列という 2 つの構造型も表現されます。これ以外のデータ型は、同等の適切な型に変換されます。

このセクションの内容:

クエリ結果を JSON として取り出すための FOR JSON 句の使用 [589 ページ]
SELECT 文内で FOR JSON 句を使用することによって、データベースに対して SQL クエリを実行し、その結果を JSON ドキュメントとして返すことができます。

FOR JSON RAW [589 ページ]
クエリで FOR JSON RAW を使用すると、各ローがフラットにされた JSON 表現として返されます。

FOR JSON AUTO [590 ページ]
クエリで FOR JSON AUTO を指定すると、クエリのジョインに基づいて、クエリが JSON オブジェクトのネストされた階層を返します。

FOR JSON EXPLICIT [591 ページ]
クエリで FOR JSON EXPLICIT を指定すると、カラムを単純な値、オブジェクト、およびネストされた階層オブジェクトとして指定して、統一された配列または異なる配列を生成できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

JSON の紹介

1.6.1 クエリ結果を JSON として取り出すための FOR JSON 句の使用

SELECT 文内で FOR JSON 句を使用することによって、データベースに対して SQL クエリを実行し、その結果を JSON ドキュメントとして返すことができます。

FOR JSON 句は、サブクエリ、GROUP BY 句または集合関数のあるクエリ、ビュー定義など、どのような SELECT 文内でも使用できます。FOR JSON 句の使用は、配列、オブジェクト、スカラ要素で構成される JSON 配列としてのリレーショナルデータを示します。

FOR JSON 句内で、生成される JSON のフォーマットを制御する次の JSON モードのうちの 1 つを指定できます。

RAW

クエリ結果をフラットにされた JSON 表現として返します。このモードはより冗長性が高いですが、解析が容易です。

AUTO

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

カラムデータの表現方法を指定できます。カラムを単一の値、オブジェクト、またはネストされたオブジェクトとして指定して、統一された配列または異なる配列を生成できます。

SQL Anywhere は、JSON 仕様に含まれないフォーマットも扱います。たとえば、SQL バイナリ値は BASE64 でエンコードされます。次のクエリは、BASE64 エンコーディングを使用してバイナリ型のカラム Photo を表示する例です。

```
SELECT Name, Photo FROM Products WHERE ID=300 FOR JSON AUTO;
```

関連情報

FOR JSON RAW [589 ページ]
FOR JSON AUTO [590 ページ]
FOR JSON EXPLICIT [591 ページ]

1.6.2 FOR JSON RAW

クエリで FOR JSON RAW を使用すると、各ローがフラットにされた JSON 表現として返されます。

構文

FOR JSON RAW

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

使用法

この句は、解析と理解が最も容易な方法であるため、クエリ結果を JSON オブジェクトとして取得する推奨の方法です。

例

次のクエリは、FOR JSON RAW を使用して、Employees テーブルから従業員情報を返します。

```
SELECT
   Empl.EmployeeID,
   SalesO.CustomerID,
   SalesO.Region
FROM Employees AS Empl KEY JOIN SalesOrders AS SalesO WHERE Empl.EmployeeID <= 195
ORDER BY 1
FOR JSON RAW;
```

結果を階層でネストするような、FOR JSON AUTO を使用した場合に返される結果とは異なり、FOR JSON RAW はフラット化された結果セットを返します。

```
[
   { "EmployeeID" : 129, "CustomerID" : 107, "Region" : "Eastern" },
   { "EmployeeID" : 129, "CustomerID" : 119, "Region" : "Western" },
   ...
   { "EmployeeID" : 129, "CustomerID" : 131, "Region" : "Eastern" },
   { "EmployeeID" " 195, "CustomerID" : 176, "Region" : "Eastern" }
]
```

1.6.3 FOR JSON AUTO

クエリで FOR JSON AUTO を指定すると、クエリのジョインに基づいて、クエリが JSON オブジェクトのネストされた階層を返します。

構文

FOR JSON AUTO

使用法

結果セットで JSON オブジェクト間の階層関係を表示する場合は、クエリで FOR JSON AUTO 句を使用します。

例

次の例は、Empl オブジェクトの JSON 配列を返します。それぞれに EmployeeID と SalesO オブジェクトが含まれます。SalesO オブジェクトは、CustomerID と地域で構成されるオブジェクトの配列です。

```
SELECT
   Empl.EmployeeID,
   SalesO.CustomerID,
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    SalesO.Region
FROM Employees AS Empl KEY JOIN SalesOrders AS SalesO WHERE Empl.EmployeeID <= 195
ORDER BY 1
FOR JSON AUTO;
```

FOR JSON RAW とは異なり、FOR JSON AUTO を使用すると、データのネストされた階層が返されます。この階層は、Empl または Employee オブジェクトが CustomerID データの配列を含む SalesO または SalesOrders オブジェクトで構成されています。

```
[
   { "Empl":
     { "EmployeeID" : 129,
     "SalesO" : [
       { "CustomerID" : 107 , "Region" : "Eastern" },
       ...
       { "CustomerID" : 131 , "Region" : "Eastern" }
     ]
     }
   },
   { "Empl" :
     { "EmployeeID" : 195,
     "SalesO" : [
       { "CustomerID" : 109 , "Region" : "Eastern" },
       ...
       { "CustomerID" : 176 , "Region" : "Eastern" }
            ]
     }
   }
]
```

1.6.4 FOR JSON EXPLICIT

クエリで FOR JSON EXPLICIT を指定すると、カラムを単純な値、オブジェクト、およびネストされた階層オブジェクトとして指定して、統一された配列または異なる配列を生成できます。

構文

FOR JSON EXPLICIT

使用法

FOR JSON EXPLICIT は、カラムエイリアスを使用して、詳細なフォーマット仕様を提供します。エイリアスが存在しない場合は、指定されたカラムが値として出力されます。エイリアスは、ネストされた構造内で値 (またはオブジェクト) を表すのに必要です。

select-list の最初の 2 つのカラムに、**TAG** および **PARENT** という名前を付けます。複数のクエリを結合すると、各クエリ内のタグと親の関係を指定して、ネストされた JSON 出力を返すことができます。

エイリアスディレクティブのフォーマットは `[encapsulating_object!tag_id!name!qualifier]` で、

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

- *!* がディレクティブの基準を区切ります。
- *encapsulating_object* は、select-list 項目のカプセル化 (配列) オブジェクトを生成します。
- *tag_id* は、後続のクエリで使用されるカラムの識別子を参照します。また、ネスティングの基準 (親に対して) を定めます。
- *name* は、(名前と値のペア) オブジェクトの名前を割り当てます。
- *qualifier* は、*ELEMENT* (デフォルト) または *HIDE* のいずれかで、後者の場合、要素が結果セットから省かれます。

例

次のクエリは、FOR JSON EXPLICIT を使用して、Employees テーブルから従業員情報を返します。

```
SELECT
    1                    AS TAG,
    NULL                 AS PARENT,
    Empl.EmployeeID      AS [!1!EmployeeID],
    SalesO.CustomerID    AS [!1!CustomerID],
    SalesO.Region        AS [!1!Region]
FROM Employees AS Empl
    KEY JOIN SalesOrders AS SalesO
WHERE Empl.EmployeeID <= 195
ORDER BY 3
FOR JSON EXPLICIT;
```

結果は FOR JSON RAW の例と同じです。

```
[
   { "EmployeeID" : 129, "CustomerID" : 107, "Region" : "Eastern" },
   { "EmployeeID" : 129, "CustomerID" : 119, "Region" : "Western" },
   ...
   { "EmployeeID" : 129, "CustomerID" : 131, "Region" : "Eastern" },
   { "EmployeeID" " 195, "CustomerID" : 176, "Region" : "Eastern" }
]
```

次の例は、FOR JSON AUTO の例に似た結果を返します。

```
SELECT
    1                    AS TAG,
    NULL                 AS PARENT,
    Empl.EmployeeID      AS [Empl!1!EmployeeID],
    NULL                 AS [SalesO!2!CustomerID],
    NULL                 AS [!2!Region]
FROM Employees AS Empl
WHERE Empl.EmployeeID <= 195
UNION ALL
SELECT
    2                    AS TAG,
    1                    AS PARENT,
    Empl.EmployeeID,
    SalesO.CustomerID,
    SalesO.Region
FROM Employees AS Empl
    KEY JOIN SalesOrders AS SalesO
WHERE Empl.EmployeeID <= 195
ORDER BY 3, 1
FOR JSON EXPLICIT;
```

上記のクエリは、次の結果を返します。

```
[
    {"Empl": [{"EmployeeID":102}]},
    {"Empl":[{"EmployeeID":105}]},
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    {"Empl":
        [{"EmployeeID":129,
            "SalesO":[
                {"CustomerID":101,"Region":"Eastern"},
                ...
                {"CustomerID":205,"Region":"Eastern"}
            ]
         }]
    },
    {"Empl":[{"EmployeeID":148}]},
    {"Empl":[{"EmployeeID":160}]},
    {"Empl":[{"EmployeeID":184}]},
    {"Empl":[{"EmployeeID":191}]},
    {"Empl":
        [{"EmployeeID":195,
            "SalesO":[
                {"CustomerID":101,"Region":"Eastern"},
               ...
                {"CustomerID":209,"Region":"Western"}
            ]
         }]
    }
]
```

配列の順序と受注のない従業員を含めること以外で、上記のフォーマットが FOR JSON AUTO の結果と異なるのは、Empl が構造の配列であることだけです。FOR JSON AUTO では、Empl に存在するオブジェクトが 1 つだけであると理解されます。FOR JSON EXPLICIT は、集合をサポートする配列のカプセル化を使用します。

次の例では、Empl のカプセル化を削除して、値として Region を返します。また、"CustomerID" を "id" に変更します。この例は、FOR JSON EXPLICIT モードによって細かいフォーマット制御を行い、RAW モードと AUTO モードの間のような結果を生成する方法を示しています。

```
SELECT
    1                 AS TAG,
    NULL              AS PARENT,
    Empl.EmployeeID   AS [!1!EmployeeID],
    NULL              AS [SalesO!2!id],
    NULL              AS [!2!]
FROM Employees AS Empl
    WHERE Empl.EmployeeID <= 195
UNION ALL
SELECT
    2                 AS TAG,
    1                 AS PARENT,
    Empl.EmployeeID,
    SalesO.CustomerID,
    SalesO.Region
FROM Employees AS Empl
    KEY JOIN SalesOrders AS SalesO
WHERE Empl.EmployeeID <= 195
ORDER BY 3, 1
FOR JSON EXPLICIT;
```

クエリの結果で、現在は SalesO がオブジェクトの配列ではなく、2 次元配列になっています。

```
[
    {"EmployeeID":102},{"EmployeeID":105},{"EmployeeID":129,
        "SalesO":[
            [{"id":101},"Eastern"],
            ...
            [{"id":205},"Eastern"]
        ]
    },
    {"EmployeeID":148},
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    {"EmployeeID":160},
    {"EmployeeID":184},
    {"EmployeeID":191},
    {"EmployeeID":195,
        "SalesO":[
            [{"id":101},"Eastern"],
            ...
            [{"id":209},"Western"]
        ]
    }
]
```

次の例は、FOR JSON RAW を使用した場合に似ていますが、EmployeeID、CustomerID、Region が名前/値のペアとしてではなく、値として出力されます。

```
SELECT
    1                    AS TAG,
    NULL                 AS PARENT,
    Empl.EmployeeID,
    SalesO.CustomerID,
    SalesO.Region
FROM Employees AS Empl KEY
    JOIN SalesOrders AS SalesO
WHERE Empl.EmployeeID <= 195
ORDER BY 3
FOR JSON EXPLICIT;
```

このクエリは以下のような結果を返します。EmployeeID、CustomerID、Region で構成される 2 次元配列が生成されています。

```
[
    [129,107,"Eastern"],
    ...
    [195,176,"Eastern"]
]
```

1.7 データのインポートとエクスポート

バルクオペレーションという用語は、データのインポートやエクスポートのプロセスを説明するために使用されます。

バルクオペレーションは一般的なエンドユーザアプリケーションの一部ではなく、実行するには特別な権限が必要です。バルクオペレーションは同時実行性とトランザクションログに影響する可能性があるため、ユーザがデータベースに接続していないときに実行する必要があります。

データをインポートおよびエクスポートする際の、一般的な状況を次に示します。

- 新しいデータベースに最初のデータセットをインポートします
- データベースの構造を修正した場合などに、新しいデータベースを構築します
- スプレッドシートなど、他のアプリケーションで使うためにデータベースからデータをエクスポートします
- レプリケーションまたは同期に使用するデータベースの抽出を作成します
- 破損したデータベースを修復します
- データベースを再構築してパフォーマンスを向上させます
- 新しいバージョンのデータベースソフトウェアを入手し、ソフトウェアアップグレードを完了します

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

バルクオペレーションのパフォーマンス面 [595 ページ]
バルクオペレーションのパフォーマンスは、オペレーションがデータベースサーバの内部と外部のどちらに対するものかなどの要因によって決まります。

バルクオペレーションのデータリカバリの問題 [596 ページ]
-b サーバオプションを使用して、バルクオペレーションモードでデータベースサーバを実行できます。

データインポート [596 ページ]
データのインポートは、バルクオペレーションとしてデータベースへのデータの読み込みを行います。

データエクスポート [614 ページ]
データのエクスポートでは、データはデータベースから書き出されます。

クライアントコンピュータ上のデータへのアクセス [631 ページ]
ファイルをデータベースサーバコンピュータにコピーしたり、データベースサーバコンピュータからファイルをコピーすることなく、SQL の文と関数を使用して、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したり、ファイルにデータをアンロードすることができます。

データベースの再構築 [633 ページ]
データベースの再構築は、データベースのアンロードと再ロードを伴うインポートとエクスポートの一種です。

データベース抽出 [650 ページ]
SQL Anywhere の統合データベースから SQL Anywhere のリモートデータベースを抽出します。

SQL Anywhere へのデータベース移行 [650 ページ]
sa_migrate システムプロシージャまたはデータベース移行ウィザードを使用して、複数のソースからテーブルをインポートします。

SQL スクリプトファイル [656 ページ]
SQL スクリプトファイルは、SQL 文を含むテキストファイルであり、同じ SQL 文を繰り返し実行する場合に便利です。

Adaptive Server Enterprise の互換性 [661 ページ]
BCP FORMAT 句を使用して、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できます。

1.7.1 バルクオペレーションのパフォーマンス面

バルクオペレーションのパフォーマンスは、オペレーションがデータベースサーバの内部と外部のどちらに対するものかなどの要因によって決まります。

内部バルクオペレーション

内部バルクオペレーションは、LOAD TABLE および UNLOAD 文を使用してデータベースサーバによって実行されるインポートおよびエクスポート操作であり、サーバ側バルクオペレーションとも呼ばれます。

内部バルクオペレーションを実行する場合は、ASCII テキストファイルまたは Adaptive Server Enterprise の BCP ファイルからロードしたり、これらのファイルにアンロードすることができます。これらのファイルは、データベース サーバと同じコンピュ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ータに存在することも、クライアントコンピュータに存在することもできます。書き込みまたは読み込み対象のファイルのパスは、データベースサーバからの相対パスとして指定します。内部バルクオペレーションは、データベースのデータのインポートおよびエクスポート方法としては最速です。

外部バルクオペレーション

外部バルクオペレーションは、INPUT および OUTPUT 文を使用して Interactive SQL などのクライアントによって実行されるインポートおよびエクスポート操作であり、クライアント側バルクオペレーションとも呼ばれます。クライアントが INPUT 文を発行すると、INPUT 文で指定されたファイルの処理時に読み込まれた各ローに対して、トランザクションログに INSERT 文が記録されます。このため、クライアント側ロードはサーバ側ロードよりかなり遅くなります。また、INPUT 中に INSERT トリガが起動されます。

OUTPUT 文により、SELECT 文の結果セットを複数の異なるファイルフォーマットで書き出すことが可能になります。

外部バルクオペレーションの場合、読み込みまたは書き込み対象のファイルのパスは、クライアントアプリケーションが実行されているコンピュータからの相対パスとして指定します。

1.7.2 バルクオペレーションのデータリカバリの問題

-b サーバオプションを使用して、バルクオペレーションモードでデータベースサーバを実行できます。

このオプションを使用した場合、データベースサーバでは一部の重要な機能を実行しません。具体的には、次のとおりです。

関数	影響
トランザクションログの管理	変更の記録がありません。COMMIT を実行するたびにチェックポイントが設定されます。
レコードのロック	重大な影響はありません。

また、バルクロードしたデータをリカバリ時にも使用できるようにします。そのためには、元のデータソースを元の場所にそのまま残します。また、LOAD TABLE 文の一部のロギングオプションを使用して、バルクロードしたデータをトランザクションログに記録することもできます。

警告

バルクオペレーションモードでは、メディア障害に対してデータベースが防御されないので、このモードの使用前後には、データベースのバックアップを行ってください。

1.7.3 データインポート

データのインポートは、バルクオペレーションとしてデータベースへのデータの読み込みを行います。

次のことが可能です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- テキストファイルからテーブル全体またはテーブルの一部をインポートします
- 変数からデータをインポートします
- スクリプトでインポート手順を自動化して、複数のテーブルを連続的にインポートします
- テーブルにデータを挿入または追加します
- テーブル内のデータを置換します
- インポートの前またはインポート中にテーブルを作成します
- 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ロードします
- BCP 形式の句を使用して、SQL Anywhere と Adaptive Server Enterprise の間でファイルを転送します

まったく新しいデータベースを作成する場合、パフォーマンスを最適化するには、LOAD TABLE を使用してデータをロードしてください。

このセクションの内容:

データのインポートのパフォーマンスに関するヒント [598 ページ]
大量のデータをインポートすると時間がかかる場合がありますが、時間を節約することができるオプションがあります。

INPUT 文を使用したデータのインポート [598 ページ]
INPUT 文を使用して、異なるファイルフォーマットのデータを既存のテーブルや新しいテーブルにインポートします。

ファイルからのデータのインポート [Interactive SQL] [599 ページ]
Interactive SQL を使用して、データをテキストファイル、Microsoft Excel ファイル、またはカンマ区切り (CSV) ファイルからデータベースにインポートします。

インポートウィザード (Interactive SQL) を使用したデータのインポート [601 ページ]
Interactive SQL のインポートウィザードは、データのソース、フォーマット、インポート先テーブルを選択するために使用します。

LOAD TABLE 文を使用したデータのインポート [603 ページ]
LOAD TABLE 文は、データベースサーバまたはクライアントコンピュータにあるデータを、テキストフォーマットや ASCII フォーマットで既存のテーブルにインポートするために使用します。

INSERT 文を使用したデータのインポート [604 ページ]
INSERT 文は、データベースにローを追加するために使用します。

MERGE 文を使用したデータのインポート [605 ページ]
MERGE 文は、更新操作を実行し、大量のテーブルデータを更新するために使用します。

プロキシテーブルを使用したデータのインポートに関するヒント [610 ページ]
プロキシテーブルを使用して、別のデータベースからのデータなどのリモートデータをインポートします。

インポート中の変換エラー [611 ページ]
外部ソースからロードされたデータは、エラーを含む場合があります。

テーブルのインポート (LOAD TABLE 文) [611 ページ]
データをテキストファイル、任意のデータベースの別のテーブル、またはシェイプファイルから、データベース内のテーブルにインポートします。

インポート用のテーブル構造 [612 ページ]
ソースデータの構造は、インポート先テーブル自体の構造と一致する必要はありません。

異なるテーブル構造のマージ [613 ページ]
INSERT 文とグローバルテンポラリテーブルを使用して、インポートデータをテーブルに合うように並べ替えます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

バルクオペレーションのパフォーマンス面 [595 ページ]
クライアントコンピュータ上のデータへのアクセス [631 ページ]
データベースの再構築 [633 ページ]

1.7.3.1 データのインポートのパフォーマンスに関するヒント

大量のデータをインポートすると時間がかかる場合がありますが、時間を節約することができるオプションがあります。

- データファイルとデータベースは、物理的に別のディスクドライブに置きます。ロード中のディスクヘッドの動きを削減できます。
- データベースのサイズを拡張します。ALTER DBSPACE 文を使うと、領域が必要になったときに少しずつサイズ拡張する代わりに、領域が必要になる前の段階でデータベースを大幅に拡張できます。また、大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の断片化を防ぐことができます。
- データのロードにテンポラリテーブルを使用します。ローカルまたはグローバルのテンポラリテーブルは、データセットを繰り返しロードする必要がある場合、または異なる構造を持つテーブルをマージする必要がある場合に役立ちます。
- LOAD TABLE 文を使用している場合は、-b オプション (バルクオペレーションモード) を指定せずにデータベースサーバを起動します。
- INPUT 文または OUTPUT 文を使用している場合は、データベースサーバと同じコンピュータ上で Interactive SQL またはクライアントアプリケーションを実行します。ネットワークを介してデータをロードすると、通信のために余分な負荷がかかります。新しいデータは、データベースサーバがビジー状態ではないときに徐々にロードしてください。

1.7.3.2 INPUT 文を使用したデータのインポート

INPUT 文を使用して、異なるファイルフォーマットのデータを既存のテーブルや新しいテーブルにインポートします。

データベースの ODBC ドライバが存在する場合は、USING 句を使用して、異なる種類のデータベースからデータをインポートします。

デフォルトの入力フォーマットを使用して、INPUT 文ごとにファイルフォーマットを指定することができます。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。

マテリアライズドビューに関する考慮事項

即時ビューでは、基本となるテーブルにデータをバルクロードしようとするとエラーが返されます。最初にビューのデータをトランケートしてから、バルクロードオペレーションを実行します。

手動ビューでは、基本となるテーブルにデータをバルクロードできます。ただし、ビュー内のデータは再表示されるまで古いままです。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブルへのバルクロードオペレーション (INPUT など) を実行する際は、先に従属するマテリアライズドビューのデータをトランケートすることを検討してください。データをロードしたら、ビューを再表示します。

テキストインデックスに関する考慮事項

即時テキストインデックスでは、基本となるテーブルで INPUT などのバルクロードオペレーションを実行してからテキストインデックスを更新すると、自動更新でも時間がかかる場合があります。手動テキストインデックスでは、再表示にも時間がかかる場合があります。

テーブルへのバルクロードオペレーション (INPUT など) を実行する際は、先に従属するテキストインデックスを削除することを検討してください。データをロードしたら、テキストインデックスを再作成します。

データベースに対する影響

INPUT 文を使用すると、変更内容はトランザクションログに記録されます。メディア障害が発生した場合は、詳細な変更の記録があります。ただし、この方法ではすべてのローがトランザクションログに書き込まれるので、この方法で大量のデータをインポートすると、パフォーマンスに影響が及びます。

対照的に、LOAD TABLE 文では各ローがトランザクションログに保存されないため、INPUT 文よりも高速な可能性があります。ただし、INPUT 文ではより多くのデータベースとファイルフォーマットがサポートされています。

1.7.3.3 ファイルからのデータのインポート [Interactive SQL]

Interactive SQL を使用して、データをテキストファイル、Microsoft Excel ファイル、またはカンマ区切り (CSV) ファイルからデータベースにインポートします。

前提条件

テーブルの所有者であるか、次の権限を持っている必要があります。

- テーブルに対する INSERT 権限、または INSERT ANY TABLE システム権限
- テーブルに対する SELECT 権限、または SELECT ANY TABLE システム権限

Microsoft Excel ワークブックファイルからデータをインポートする場合、互換性のある ODBC ドライバがインストールされている必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

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。

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。

手順

1. Interactive SQL を開き、データベースに接続します。
2. 次のオプションのうちの 1 つを選択してください。

オプション	アクション
INPUT 文を使用して、**TEXT** ファイルからデータをインポートします。	次のクエリを実行します。 `INPUT INTO TableName FROM 'filepath' FORMAT TEXT SKIP 1;` この文では、インポート先テーブルの名前は Products で、newSwimwear.csv はデータファイルの名前です。カラム名が含まれているファイルの最初の行はスキップされます。このファイルはクライアントコンピュータに関連して配置されます。
INPUT 文を使用して、**Microsoft Excel** ファイルからデータをインポートします。	次のクエリを実行します。 `INPUT INTO TableName FROM 'filepath' FORMAT EXCEL WORKSHEET 'Book2'` WORKSHEET 句で、インポート元となる Microsoft Excel ファイル内のシートを指定します。この句に値が指定されない場合、ファイルの最初のシートからデータがインポートされます。ワークリストの最初のローには、カラム名が含まれると見なされます。
インポートウィザードを使用したデータのインポート	1. データ > インポート をクリックします。 2. インポートウィザードの指示に従います。

例

INPUT 文を使用して拡張子が .xls の Microsoft Excel ファイルからデータを入力するには、次の手順を実行します。

1. Microsoft Excel で、データを XLS ファイルに保存します。たとえば、そのファイルに newSales.xls という名前を付けます。
2. Interactive SQL で、SQL Anywhere サンプルデータベースなどのデータベースに接続します。
3. imported_sales という名前のテーブルを作成します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

4. INPUT 文を実行します。

```
INPUT INTO imported_sales
FROM 'C:¥¥LocalTemp¥¥newSales.xls'
FORMAT EXCEL
WORKSHEET 'Book2'
```

関連情報

データエクスポート [614 ページ]
データエクスポート [614 ページ]

1.7.3.4 インポートウィザード (Interactive SQL) を使用したデータのインポート

Interactive SQL のインポートウィザードは、データのソース、フォーマット、インポート先テーブルを選択するために使用します。

前提条件

データを既存のテーブルにインポートする場合は、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 権限と INSERT 権限を持っているか、SELECT ANY TABLE システム権限と INSERT ANY TABLE システム権限を持っている必要があります。

データを新しいテーブルにインポートする場合、CREATE TABLE、CREATE ANY TABLE、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。

コンテキスト

データはテキストファイル、Microsoft Excel ファイル、固定フォーマットファイル、シェイプファイルから、既存のテーブルや新しいテーブルにインポートできます。

インポートウィザードを使用して、異なるタイプや異なるバージョンのデータベース間でデータをインポートします。

次の場合は、Interactive SQL のインポートウィザードを使用します。

- データのインポートと同時にテーブルを作成する場合
- テキスト以外のフォーマットでのデータのインポートにポイントアンドクリックインタフェースを使用する場合

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. Interactive SQL で、データ > インポートをクリックします。
2. ファイルタイプを指定し、次へをクリックします。
3. ファイル名フィールドで、参照をクリックしてファイルを追加します。
4. データのインポート先とするテーブルを指定します。新しいテーブルの場合、テーブル名に入力します。
5. 次へをクリックします。
6. テキストファイルの場合は、そのファイルを読み込む方法を指定して次へをクリックします。
7. カラム名やデータ型の変更を行って、インポートをクリックします。
8. 閉じるをクリックします。

例

データを SQL Anywhere サンプルデータベースから Ultra Light データベースにインポートするには、次の手順を実行します。

1. *C:¥Users¥Public¥Documents¥SQL Anywhere 17¥Samples*`¥UltraLite¥CustDB¥custdb.udb` などの Ultra Light データベースに接続します。
2. Interactive SQL で、データ > インポートをクリックします。
3. データベースをクリックします。次へをクリックします。
4. データベースタイプリストで、*SQL Anywhere* をクリックします。
5. アクションドロップダウンリストで、*ODBC* データソースを使用した接続をクリックします。
6. *ODBC* データソース名をクリックし、その下のボックスに *SQL Anywhere 17 Demo* と入力して、パスワードを **`sql`** と指定します。
7. 次へをクリックします。
8. テーブル名リストで、*Customers* をクリックします。次へをクリックします。
9. 新しいテーブルをクリックします。
10. テーブル名フィールドに **`SQLAnyCustomers`** と入力します。
11. インポートをクリックします。
12. 閉じるをクリックします。
13. 生成された SQL 文を表示するには、*SQL* > 前の *SQL* をクリックします。
    インポートウィザードによって生成された INPUT 文が *SQL* 文ウィンドウ枠に次のように表示されます。

```
--  Generated by the Import Wizard
input using 'dsn=SQL Anywhere 17 Demo;PWD=sql;CON='''''
    from "GROUPO.Customers" into "SQLAnyCustomers"
    create table on
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.3.5 LOAD TABLE 文を使用したデータのインポート

LOAD TABLE 文は、データベースサーバまたはクライアントコンピュータにあるデータを、テキストフォーマットや ASCII フォーマットで既存のテーブルにインポートするために使用します。

また、LOAD TABLE 文を使用すると、別のテーブルのカラムや値の式 (関数やシステムプロシージャの結果など) からデータをインポートすることもできます。データをいくつかのビューにインポートすることもできます。

LOAD TABLE 文はテーブルにローを追加します。置き換えることはしません。

LOAD TABLE 文 (WITH ROW LOGGING および WITH CONTENT LOGGING オプションなし) を使用すると、INPUT 文を使用するよりもずっと早くデータをロードできます。

LOAD TABLE 文を使用してロードされたデータに対してはトリガは起動しません。

マテリアライズドビューに関する考慮事項

即時ビューでは、基本となるテーブルにデータをバルクロードしようとするとエラーが返されます。最初にビューのデータをトランケートしてから、バルクロードオペレーションを実行します。

手動ビューでは、基本となるテーブルにデータをバルクロードできます。ただし、ビュー内のデータは再表示されるまで古いままです。

テーブルへのバルクロードオペレーション (LOAD TABLE など) を実行する際は、先に従属するマテリアライズドビューのデータをトランケートすることを検討してください。データをロードしたら、ビューを再表示します。

テキストインデックスに関する考慮事項

即時テキストインデックスでは、基本となるテーブルで LOAD TABLE などのバルクロードオペレーションを実行してからテキストインデックスを更新すると、自動更新にもかかわらず時間がかかる場合があります。手動テキストインデックスでは、再表示にも時間がかかる場合があります。

テーブルへのバルクロードオペレーション (LOAD TABLE など) を実行する際は、先に従属するテキストインデックスを削除することを検討してください。データをロードしたら、テキストインデックスを再作成します。

データベースリカバリと同期の考慮事項

デフォルトでは、データをファイルからロードする場合 (`LOAD TABLE table-name FROM filename;` など)、LOAD TABLE 文のみがトランザクションログに記録され、ロードされる実際のデータのローは記録されません。このため、元のロードファイルの変更、移動、削除後にトランザクションログを使用してデータベースのリカバリを行おうとすると、問題が発生します。また、同期やレプリケーションを行うデータベースが新しいデータを取得できません。

リカバリや同期の考慮事項に対応するため、LOAD TABLE 文では 2 つのロギングオプションを使用できます。WITH ROW LOGGING を使用すると、ローをロードするごとに、トランザクションログに INSERT 文を作成します。WITH CONTENT

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

LOGGING を使用すると、ロードしたローをチャンクにグループ化して、そのチャンクをトランザクションログに記録します。これらのオプションにより、ロードデータのソースがなくなっても、ロード操作を繰り返すことができます。

データベースのミラーリングの考慮事項

データベースでミラーリングを行っている場合は、LOAD TABLE 文の使用に注意してください。たとえば、ファイルからデータをロードする場合、ミラーサーバでファイルをロードできるかどうかや、ミラーデータベースがロードを処理するまでにロード元のソースのデータが変更されるかどうかに注意してください。これらのいずれかのリスクが存在する場合は、LOAD TABLE 文のロギングレベルとして WITH ROW LOGGING または WITH CONTENT LOGGING のいずれかを指定してください。これにより、ミラーデータベースにロードされるデータが、ミラーリングされているデータベースにロードされたデータと同一になります。

関連情報

クライアントコンピュータ上のデータへのアクセス [631 ページ]

1.7.3.6 INSERT 文を使用したデータのインポート

INSERT 文は、データベースにローを追加するために使用します。

INSERT 文にはインポート先テーブルにインポートするデータが含まれているため、この文は対話型入力と見なされます。リモートデータアクセスに INSERT 文を使用して、ファイルではなく別のデータベースからデータをインポートすることもできます。

次の場合は、INSERT 文を使用してデータをインポートします。

- 1 つのテーブルに少量のデータをインポートする場合
- ファイルフォーマットが柔軟な場合
- ファイルではなく外部データベースからリモートデータをインポートする場合

INSERT 文では、ON EXISTING 句を使用して、挿入するローがすでに挿入先のテーブルに存在する場合に実行するアクションを指定できます。ただし、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。

マテリアライズドビューに関する考慮事項

即時ビューでは、基本となるテーブルにデータをバルクロードしようとするとエラーが返されます。最初にビューのデータをトランケートしてから、バルクロードオペレーションを実行します。

手動ビューでは、基本となるテーブルにデータをバルクロードできます。ただし、ビュー内のデータは再表示されるまで古いままです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブルへのバルクロードオペレーション (INSERT など) を実行する際は、先に従属するマテリアライズドビューのデータをトランケートすることを検討してください。データをロードしたら、ビューを再表示します。

テキストインデックスに関する考慮事項

即時テキストインデックスでは、基本となるテーブルで INSERT などのバルクロードオペレーションを実行してからテキストインデックスを更新すると、自動更新にもかかわらず時間がかかる場合があります。手動テキストインデックスでは、再表示にも時間がかかる場合があります。

テーブルへのバルクロードオペレーション (INSERT など) を実行する際は、先に従属するテキストインデックスを削除することを検討してください。データをロードしたら、テキストインデックスを再作成します。

データベースに対する影響

INSERT 文を使用すると、変更内容はトランザクションログに記録されます。データベースファイルに関するメディア障害が発生した場合は、変更内容に関する情報をトランザクションログからリカバリできます。

1.7.3.7 MERGE 文を使用したデータのインポート

MERGE 文は、更新操作を実行し、大量のテーブルデータを更新するために使用します。

データのマージ時に、ソースデータのローがターゲットデータ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指定できます。

マージ動作の定義

MERGE 文の構文の省略形を以下に示します。

```
MERGE INTO target-object
USING source-object
ON merge-search-condition
{ WHEN MATCHED | WHEN NOT MATCHED } [...]
```

データベースによってマージ操作が実行されると、source-object のローと target-object のローが比較され、ON 句に含まれる定義に基づいて一致する行または一致しない行が検索されます。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。

source-object は、ベーステーブル、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の結果のいずれかです。target-object には、これらのオブジェクトのうち、マテリアライズドビューとプロシージャ以外の任意のオブジェクトを指定できます。

ANSI/ISO SQL 標準では、マージ操作中に target-object のローを source-object の複数のローで更新することは許可されません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

source-object のローが一致または不一致と見なされると、それぞれ一致の場合と不一致の場合の WHEN 句 (WHEN MATCHED または WHEN NOT MATCHED) に対して評価されます。WHEN MATCHED 句では、target-object のローに対して実行するアクションを定義します (たとえば、WHEN MATCHED ... UPDATE は、target-object のローを更新することを指定します)。WHEN NOT MATCHED 句では、source-object の一致しないローを使用して、target-object に対して実行するアクションを定義します。

WHEN 句は無制限に指定でき、指定した順序で処理されます。また、WHEN 句内で AND 句を使用し、ローのサブセットに対するアクションを指定することもできます。たとえば、次の WHEN 句では、一致したローの Quantity カラムの値に応じて、異なるアクションを実行するよう定義しています。

```
WHEN MATCHED AND myTargetTable.Quantity<=500 THEN SKIP
WHEN MATCHED AND myTargetTable.Quantity>500 THEN UPDATE SET
myTargetTable.Quantity=500
```

マージ操作における分岐

一致するローおよび一致しないローをアクションごとに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。分岐は、単一の WHEN MATCHED 句または WHEN NOT MATCHED 句と同等です。たとえば、ある分岐に source-object の一致しないローのセットが含まれている場合は、それらのローを挿入する必要があります。分岐アクションの実行は、すべての分岐アクティビティを完了してから (source-object のすべてのローを評価してから) 開始されます。データベースサーバは、WHEN 句が指定された順序に従って、分岐アクションの実行を開始します。

source-object の一致しないロー、または source-object と target-object の一致するローのペアが分岐に入ると、後続の分岐に対して評価されません。したがって、WHEN 句を指定する順序は重要です。

一致または不一致と見なされる source-object のローのうち、いずれの分岐にも属さない (つまり、いずれの WHEN 句も満たさない) ものは無視されます。これは、WHEN 句に AND 句が含まれていて、ローがいずれの AND 句の条件も満たさない場合に発生します。この場合、アクションが定義されていないため、ローは無視されます。

データを変更するアクションは、個々の INSERT、UPDATE、DELETE 文としてトランザクションログに記録されます。

ターゲットテーブルに定義されたトリガ

通常、マージ操作中に各 INSERT、UPDATE、DELETE 文を実行すると、トリガが起動されます。たとえば、UPDATE アクションが定義された分岐の処理時に、データベースサーバは次の内容を実行します。

1. すべての BEFORE UPDATE トリガを起動します。
2. ローの候補セットに対して UPDATE 文を実行すると同時に、すべてのローレベルの UPDATE トリガを起動します。
3. AFTER UPDATE トリガを起動します。

target-table に対してトリガを起動すると、別の分岐で更新されるローに影響が及ぶ場合、マージ操作で競合が発生する可能性があります。たとえば、ロー B を削除するトリガを起動するアクションをロー A に対して実行するとします。しかし、ロー B には独自のアクションが定義されており、まだ実行されていません。ローに対してアクションを実行できないと、マージ操作は失敗し、すべての変更がロールバックされ、エラーが返されます。

複数のトリガアクションが定義されたトリガは、同じトリガに各トリガアクションを 1 つずつ指定したものと見なされます (つまり、各トリガに 1 つのトリガアクションを指定して、別個のトリガを定義したのと同等になります)。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

即時マテリアライズドビューに関する考慮事項

MERGE 文によって多数のローを更新すると、データベースサーバのパフォーマンスに影響する場合があります。多数のローを更新する場合は、従属する即時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 MERGE 文を実行することを検討してください。MERGE 文を実行したら、REFRESH MATERIALIZED VIEW 文を実行します。

テキストインデックスに関する考慮事項

MERGE 文によって多数のローを更新すると、データベースサーバのパフォーマンスに影響する場合があります。テーブルで MERGE 文を実行する際は、先に従属するテキストインデックスを削除することを検討してください。MERGE 文を実行したら、テキストインデックスを再作成します。

例

例 1

ジャケットとセーターを販売する小さい会社を経営しているとします。ジャケットの素材の価格が 5% 上昇したため、それに合わせて価格を調整したいとします。次の CREATE TABLE 文を使用して、販売するジャケットとセーターの現在の価格情報を保持する myProducts という小さいテーブルを作成します。その後の INSERT 文で、myProducts にデータを入力します。この例では、CREATE TABLE 権限が必要です。

```
CREATE TABLE myProducts (
   product_id    NUMERIC(10),
   product_name  CHAR(20),
   product_size  CHAR(20),
   product_price NUMERIC(14,2));
INSERT INTO myProducts VALUES (1, 'Jacket', 'Small', 29.99);
INSERT INTO myProducts VALUES (2, 'Jacket', 'Medium', 29.99);
INSERT INTO myProducts VALUES (3, 'Jacket', 'Large', 39.99);
INSERT INTO myProducts VALUES (4, 'Sweater', 'Small', 18.99);
INSERT INTO myProducts VALUES (5, 'Sweater', 'Medium', 18.99);
INSERT INTO myProducts VALUES (6, 'Sweater', 'Large', 19.99);
SELECT * FROM myProducts;
```

product_id	product_name	product_size	product_price
1	Jacket	Small	29.99
2	Jacket	Medium	29.99
3	Jacket	Large	39.99
4	Sweater	Small	18.99
5	Sweater	Medium	18.99
6	Sweater	Large	19.99

さらに、次の文を使用して、ジャケットの価格変更に関する情報を保持する myPrices という別のテーブルを作成します。マージ操作を実行する前に myPrices テーブルの内容を確認できるように、末尾に SELECT 文を追加します。

```
CREATE TABLE myPrices (
   product_id    NUMERIC(10),
   product_name  CHAR(20),
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    product_size  CHAR(20),
    product_price NUMERIC(14,2),
    new_price     NUMERIC(14,2));
INSERT INTO myPrices (product_id) VALUES (1);
INSERT INTO myPrices (product_id) VALUES (2);
INSERT INTO myPrices (product_id) VALUES (3);
INSERT INTO myPrices (product_id) VALUES (4);
INSERT INTO myPrices (product_id) VALUES (5);
INSERT INTO myPrices (product_id) VALUES (6);
SELECT * FROM myPrices;
```

product_id	product_name	product_size	product_price	new_price
1	(NULL)	(NULL)	(NULL)	(NULL)
2	(NULL)	(NULL)	(NULL)	(NULL)
3	(NULL)	(NULL)	(NULL)	(NULL)
4	(NULL)	(NULL)	(NULL)	(NULL)
5	(NULL)	(NULL)	(NULL)	(NULL)
6	(NULL)	(NULL)	(NULL)	(NULL)

次の MERGE 文を使用して、myProducts テーブルのデータを myPrices テーブルにマージします。`source-object` は、product_name が Jacket のローのみを含むようフィルタリングされた派生テーブルです。また、ON 句では、`target-object` と `source-object` の product_id カラムの値が一致する場合にローが一致するよう指定しています。

```
MERGE INTO myPrices p
USING ( SELECT
      product_id,
      product_name,
      product_size,
      product_price
  FROM myProducts
  WHERE product_name='Jacket') pp
ON (p.product_id = pp.product_id)
WHEN MATCHED THEN
  UPDATE SET
    p.product_id=pp.product_id,
    p.product_name=pp.product_name,
    p.product_size=pp.product_size,
    p.product_price=pp.product_price,
    p.new_price=pp.product_price * 1.05;
SELECT * FROM myPrices;
```

product_id	product_name	product_size	product_price	new_price
1	Jacket	Small	29.99	31.49
2	Jacket	Medium	29.99	31.49
3	Jacket	Large	39.99	41.99
4	(NULL)	(NULL)	(NULL)	(NULL)
5	(NULL)	(NULL)	(NULL)	(NULL)
6	(NULL)	(NULL)	(NULL)	(NULL)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

product_id が 4、5、6 のカラムの値は、NULL のままになります。これは、myProducts テーブルで製品が (`product_name='Jacket'`) であるいずれのローとも一致しなかったためです。

例 2

次の例では、myTargetTable のプライマリキー値を使用してローを一致させ、mySourceTable テーブルと myTargetTable テーブルのローをマージします。mySourceTable のローに、myTargetTable のプライマリキーカラムと同じ値が含まれている場合、ローは一致していると見なされます。

```
MERGE INTO myTargetTable
   USING mySourceTable ON PRIMARY KEY
   WHEN NOT MATCHED THEN INSERT
   WHEN MATCHED THEN UPDATE;
```

WHEN NOT MATCHED THEN INSERT 句は、mySourceTable にはあって myTargetTable にはないローを、myTargetTable に追加することを指定します。WHEN MATCHED THEN UPDATE 句は、myTargetTable の一致するローを mySourceTable の値に更新することを指定します。

次の構文は前述の構文と同等です。ここでは、myTargetTable にはカラム (I1, I2, ..In) があり、プライマリキーがカラム (I1, I2) に定義されていることを前提としています。mySourceTable にはカラム (U1, U2, .. Un) があります。

```
MERGE INTO myTargetTable ( I1, I2, .. ., In )
   USING mySourceTable ON myTargetTable.I1 = mySourceTable.U1
      AND myTargetTable.I2 = mySourceTable.U2
   WHEN NOT MATCHED
      THEN INSERT ( I1, I2, .. In )
         VALUES ( mySourceTable.U1, mySourceTable.U2, ..., mySourceTable.Un )
   WHEN MATCHED
      THEN UPDATE SET
      myTargetTable.I1 = mySourceTable.U1,
      myTargetTable.I2 = mySourceTable.U2,
      ...
      myTargetTable.In = mySourceTable.Un;
```

RAISERROR アクションの使用

一致または不一致のアクションに指定できるアクションの 1 つが、RAISERROR です。RAISERROR を使用すると、WHEN 句の条件を満たした場合に、マージ操作を失敗させることができます。

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 の指定は、後でエラーの発生した状況を特定する場合に便利です。

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。

次の文で、SQLCODE のカスタマイズによって返される内容にどのような影響が出るのかを簡単に示します。この例では、CREATE TABLE 権限が必要です。

次のように targetTable を作成します。

```
CREATE TABLE targetTable( c1 int );
INSERT INTO targetTable VALUES( 1 );
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次の文は、SQLSTATE = '23510' および SQLCODE = -1254 のエラーを返します。

```
MERGE INTO targetTable
   USING (SELECT 1 c1 ) AS sourceData
   ON targetTable.c1 = sourceData.c1
   WHEN MATCHED THEN RAISERROR;
SELECT sqlstate, sqlcode;
```

次の文は、SQLSTATE = '23510' および SQLCODE = -17001 のエラーを返します。

```
MERGE INTO targetTable
   USING (SELECT 1 c1 ) AS sourceData
   ON targetTable.c1 = sourceData.c1
   WHEN MATCHED THEN RAISERROR 17001
   WHEN NOT MATCHED THEN RAISERROR 17002;
SELECT sqlstate, sqlcode;
```

次の文は、SQLSTATE = '23510' および SQLCODE = -17002 のエラーを返します。

```
MERGE INTO targetTable
   USING (SELECT 2 c1 ) AS sourceData
   ON targetTable.c1 = sourceData.c1
   WHEN MATCHED THEN RAISERROR 17001
   WHEN NOT MATCHED THEN RAISERROR 17002;
SELECT sqlstate, sqlcode;
```

1.7.3.8 プロキシテーブルを使用したデータのインポートに関するヒント

プロキシテーブルを使用して、別のデータベースからのデータなどのリモートデータをインポートします。

プロキシテーブルは、メタデータを含むローカルテーブルです。リモートデータベースサーバのテーブルに、ローカルテーブルであるかのようにアクセスするときに使用します。

データベースに対する影響

プロキシテーブルを使用すると、変更内容はトランザクションログに記録されます。データベースファイルに関するメディア障害が発生した場合は、変更内容に関する情報をトランザクションログからリカバリできます。

プロキシテーブルの使用方法

プロキシテーブルを作成し、SELECT 句を指定した INSERT 文を使用してリモートデータベースからデータベース内の永久テーブルにデータを挿入します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

リモートデータアクセス [661 ページ]

1.7.3.9 インポート中の変換エラー

外部ソースからロードされたデータは、エラーを含む場合があります。

たとえば、無効な日付や数字が含まれている可能性があります。conversion_error データベースオプションを使用すると、変換エラーを無視し、無効な値を NULL 値に変換できます。

1.7.3.10 テーブルのインポート (LOAD TABLE 文)

データをテキストファイル、任意のデータベースの別のテーブル、またはシェイプファイルから、データベース内のテーブルにインポートします。

前提条件

ユーザ本人が所有するテーブルを作成するには、CREATE TABLE 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。

データのインポート (ロード) に必要な権限は、-gl データベースオプションの設定値とデータのインポート元によって異なります。データのロードに必要な権限についての詳細は、LOAD TABLE 文の項を参照してください。

手順

1. CREATE TABLE 文を使用してインポート先テーブルを作成します。例:

```
CREATE TABLE Departments (
DepartmentID           integer NOT NULL,
DepartmentName         char(40) NOT NULL,
DepartmentHeadID       integer NULL,
CONSTRAINT DepartmentsKey PRIMARY KEY (DepartmentID) );
```

2. LOAD TABLE 文を実行します。例:

```
LOAD TABLE Departments
FROM 'C:¥¥ServerTemp¥¥Departments.csv';
```

3. 値の中の後続ブランクをそのままにするには、LOAD TABLE 文中で STRIP OFF 句を使用します。デフォルトの設定 (STRIP RTRIM) では、値は後続ブランクが取り除かれてから挿入されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

LOAD TABLE 文はファイルの内容を、テーブルの既存のローに追加します。既存のローを置き換えるわけではありません。テーブルからすべてのローを削除するには、TRUNCATE TABLE 文を使用します。

FROM 句は、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。

TRUNCATE TABLE 文と LOAD TABLE 文は、カスケード型削除のようなトリガの起動や参照整合性に関わるアクションの実行を行うことはありません。

結果

データが指定されたテーブルにインポートされます。

1.7.3.11 インポート用のテーブル構造

ソースデータの構造は、インポート先テーブル自体の構造と一致する必要はありません。

たとえば、カラムのデータ型が異なる、順序が異なる、またはロード先のテーブルのカラム数を超えるインポートデータの値があるなどです。

テーブルまたはデータの並べ替え

インポートするデータの構造がインポート先テーブルの構造と一致しないことがわかっている場合は、次の操作を行うことができます。

- LOAD TABLE 文でロードするカラム名のリストを入力することができます。
- INSERT 文の一種とグローバルテンポラリテーブルを使用して、インポートデータをテーブルに合うように並べ替えることができます。
- INPUT 文を使用して、カラムの特定のセットまたは順序を指定できます。

カラムに **NULL** 値を入力できるようにする

インポート中のファイルにテーブルのカラムのサブセット用のデータがある場合、またはカラムの順序が異なる場合は、LOAD TABLE 文の DEFAULTS オプションを使用して、ブランクを埋めて一致しないテーブル構造をマージすることもできます。

- DEFAULTS オプションが 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 オプションが OFF で、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 オプションが ON で、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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば、Customers テーブルの City カラムにデフォルト値を定義してから、次のような LOAD TABLE 文を使用して、データベースサーバコンピュータ上の `C:¥ServerTemp` ディレクトリにある `newCustomers.csv` というファイルから、Customers テーブルに新しいローをロードできます。

```
ALTER TABLE Customers
ALTER City DEFAULT 'Waterloo';
LOAD TABLE Customers ( Surname, GivenName, Street, State, Phone )
FROM 'C:¥¥ServerTemp¥¥newCustomers.csv'
DEFAULTS ON;
```

City カラムには値が入力されていないため、デフォルト値が入力されます。`DEFAULTS OFF` が指定されている場合は、City カラムには空の文字列が割り当てられます。

1.7.3.12 異なるテーブル構造のマージ

INSERT 文とグローバルテンポラリテーブルを使用して、インポートデータをテーブルに合うように並べ替えます。

前提条件

グローバルテンポラリテーブルを作成するには、次のいずれかのシステム権限を持っている必要があります。

- CREATE TABLE
- CREATE ANY TABLE
- CREATE ANY OBJECT

データのインポート (ロード) に必要な権限は、-gl データベースオプションの設定値とデータのインポート元によって異なります。データのロードに必要な権限についての詳細は、LOAD TABLE 文の項を参照してください。

INSERT 文を使用するには、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。

- INSERT ANY TABLE システム権限
- そのテーブルに対する INSERT 権限

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、テーブルに対する UPDATE 権限を持っている必要があります。

手順

1. *SQL 文*ウィンドウ枠で、入力ファイルと構造が一致するグローバルテンポラリテーブルを作成します。

   CREATE TABLE 文を使用して、グローバルテンポラリテーブルを作成します。
2. LOAD TABLE 文を使用して、作成したグローバルテンポラリテーブルにデータをロードします。

   データベース接続を閉じると、グローバルテンポラリテーブル内のデータは消去されます。ただし、テーブル定義は残ります。この定義は、次にデータベースに接続するときに使用します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

3. INSERT 文と SELECT 句を使用し、テンポラリテーブルからデータを抽出して要約し、データベースの 1 つまたは複数の永久テーブルにコピーします。

結果

データが、永久データベーステーブルにロードされます。

例

次は前述した手順の例です。

```
CREATE GLOBAL TEMPORARY TABLE TempProducts
(
    ID                      integer NOT NULL,
    Name                    char(15) NOT NULL,
    Description             char(30) NOT NULL,
    Size                    char(18) NOT NULL,
    Color                   char(18) NOT NULL,
    Quantity                integer NOT NULL,
    UnitPrice               numeric(15,2) NOT NULL,
    CONSTRAINT ProductsKey PRIMARY KEY (ID)
)
ON COMMIT PRESERVE ROWS;
LOAD TABLE TempProducts
FROM 'C:¥¥ServerTemp¥¥newProducts.csv'
SKIP 1;
INSERT INTO Products WITH AUTO NAME
    (SELECT Name, Description, ID, Size, Color, Quantity,
            UnitPrice * 1.25 AS UnitPrice
    FROM TempProducts);
```

1.7.4 データエクスポート

データのエクスポートでは、データはデータベースから書き出されます。

データのエクスポートは、データベースの大部分を共有する必要がある場合、または特定の基準に従ってデータベースの一部を抽出する必要がある場合に便利です。次のことが可能です。

- 個々のテーブル、クエリ結果、またはテーブルスキーマをエクスポートします。
- 複数のテーブルを連続してエクスポートできるようにするために、エクスポートを自動化するスクリプトを作成します。
- 多くの異なるファイルフォーマットにエクスポートします。
- クライアントコンピュータにあるファイルにデータをエクスポートします。
- BCP FORMAT 句を使用して、SQL Anywhere と Adaptive Server Enterprise の間でファイルをエクスポートします。

データをエクスポートする前に、所有しているリソースの種類、およびデータベースからエクスポートする情報の種類を判断します。

データベース全体をエクスポートする場合は、パフォーマンスを考慮して、データをエクスポートするのではなくデータベースをアンロードしてください。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

エクスポートの制限事項

Microsoft Excel ODBC ドライバを使用して、SQL Anywhere データベースから Microsoft Excel データベースにデータをエクスポートすると、以下のデータ型の変更が発生することがあります。

- CHAR、LONG VARCHAR、NCHAR、NVARCHAR、または LONG NVARCHAR データ型として格納されているデータをエクスポートする場合、データは VARCHAR (Microsoft Excel ドライバによってサポートされる最も近い型) として格納されます。
  Microsoft Excel ODBC ドライバがサポートするテキストカラムの幅は、255 文字までです。
- MONEY データ型と SMALLMONEY データ型として格納されているデータは、CURRENCY データ型にエクスポートされます。それ以外の場合、数値データは数値としてエクスポートされます。

このセクションの内容:

エクスポートウィザードを使用したデータのエクスポート [616 ページ]
Interactive SQL の**エクスポートウィザード**は、特定のフォーマットのクエリ結果をファイルやデータベースにエクスポートするために使用します。

OUTPUT 文を使用したデータのエクスポートに関するヒント [617 ページ]
OUTPUT 文を使用して、データベースからクエリ結果、テーブル、またはビューをエクスポートします。

UNLOAD TABLE 文を使用したデータのエクスポートに関するヒント [618 ページ]
UNLOAD TABLE 文を使用すると、テキストフォーマットだけでデータを効率的にエクスポートできます。

UNLOAD 文を使用したデータのエクスポートに関するヒント [619 ページ]
UNLOAD 文は、クエリ結果をファイルにエクスポートするという点で OUTPUT 文に似ています。

アンロードユーティリティ (dbunload) を使用したデータのエクスポートに関するヒント [620 ページ]
アンロードユーティリティ (dbunload) を使用して、1 つ、複数、またはすべてのデータベーステーブルをエクスポートします。

データベースアンロードウィザードを使用したデータのエクスポートに関するヒント [620 ページ]
データベースアンロードウィザードは、データベースを新しいデータベースにアンロードするために使用します。

CSV または Microsoft Excel スプレッドシートファイルへのクエリ結果のエクスポート [Interactive SQL] [622 ページ]
OUTPUT 文を使用して、クエリ結果を Microsoft Excel ワークブックファイルまたは CSV ファイルにエクスポートします。

データのアンロードウィンドウを使用したデータのエクスポート [624 ページ]
データのアンロードウィンドウを使用して SQL Central のテーブルをアンロードします。

UNLOAD 文を使用したクエリ結果のエクスポート [625 ページ]
UNLOAD 文を使用して、Interactive SQL のクエリ結果をエクスポートします。

Interactive SQL での NULL 値の処理設定 [626 ページ]
Interactive SQL の**結果**ウィンドウ枠を設定することにより、OUTPUT 文を使用したときに NULL 値をどのように表すのかを指定します。

データベースのエクスポート (SQL Central の場合) [627 ページ]
SQL Central の**データベースアンロードウィザード**を使用して、データをデータベースから再ロードファイル、新規データベース、または既存のデータベースにアンロードします。

データベースのエクスポート (コマンドラインの場合) [628 ページ]
コマンドラインのアンロードユーティリティ (dbunload) を使用して、データをデータベースから再ロードファイル、新規データベース、または既存のデータベースにアンロードします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブルのエクスポート (SQL の場合) [629 ページ]
Interactive SQL から UNLOAD TABLE 文を実行して、テーブルをエクスポートします。

テーブルのエクスポート (コマンドラインの場合) [630 ページ]
コマンドラインのアンロードユーティリティ (dbunload) を実行して、テーブルをエクスポートします。

関連情報

バルクオペレーションのパフォーマンス面 [595 ページ]
クライアントコンピュータ上のデータへのアクセス [631 ページ]
データベースの再構築 [633 ページ]

1.7.4.1 エクスポートウィザードを使用したデータのエクスポート

Interactive SQL のエクスポートウィザードは、特定のフォーマットのクエリ結果をファイルやデータベースにエクスポートするために使用します。

前提条件

問い合わせる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。

手順

1. クエリを実行します。
2. Interactive SQL で、データ > エクスポート をクリックします。
3. エクスポートウィザードの指示に従います。

結果

クエリ結果は、指定されたファイルまたはデータベースにエクスポートされます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

1. サンプルデータベースに接続している状態で、次のクエリを実行します。Employees テーブルに対する SELECT 権限を持っているか、SELECT ANY TABLE システム権限を持っている必要があります。

```
SELECT * FROM Employees WHERE State = 'GA';
```

2. 結果セットには、ジョージア州に住むすべての従業員のリストが含まれます。
3. データ > エクスポート をクリックします。
4. データベースをクリックしてから、次へをクリックします。
5. データベースタイプリストで、*Ultra Light* をクリックします。
6. ユーザ *ID* フィールドに、**DBA** と入力します。
7. パスワード項目に、**sql** と入力します。
8. データベースファイル項目に、*C:¥Users¥Public¥Documents¥SQL Anywhere 17¥Samples*¥UltraLite¥CustDB¥custdb.udb と入力します。
9. 次へをクリックします。
10. 新しいテーブルを作成をクリックします。
11. テーブル名フィールドに **GAEmployees** と入力します。
12. エクスポートをクリックします。
13. 閉じるをクリックします。
14. *SQL* > 前の *SQL* をクリックします。
エクスポートウィザードによって作成および使用された OUTPUT USING 文が *SQL* 文ウィンドウ枠に表示されます。

```
--  Generated by the Export Wizard
output using 'driver=UltraLite 17;UID=DBA;PWD=***;
DBF=C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥¥Samples¥¥UltraLite¥¥CustDB¥
¥custdb.udb'
    into "GAEmployees"
    create table on
```

1.7.4.2 OUTPUT 文を使用したデータのエクスポートに関するヒント

OUTPUT 文を使用して、データベースからクエリ結果、テーブル、またはビューをエクスポートします。

OUTPUT 文は、SELECT 文の結果セットを複数の異なるファイルフォーマットで書き出すことができるため、互換性が重要な場合に役立ちます。デフォルトの出力フォーマットを使用したり、OUTPUT 文ごとにファイルフォーマットを指定することができます。Interactive SQL では、複数の OUTPUT 文が含まれた SQL スクリプトファイルを実行できます。

Interactive SQL のデフォルトの出力フォーマットは、*Interactive SQL* のオプションウィンドウ (Interactive SQL で ツール > オプション をクリック) のインポート/エクスポートタブで指定します。

次の場合は、Interactive SQL の OUTPUT 文を使用します。

- テキスト以外のフォーマットでテーブルまたはビューのすべてまたは一部をエクスポートする場合
- SQL スクリプトファイルを使用してエクスポート処理を自動化する場合

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベースに対する影響

OUTPUT 文、UNLOAD 文、UNLOAD TABLE 文のいずれかを選択できる場合は、パフォーマンスを考慮して UNLOAD TABLE 文を選択します。

OUTPUT 文を使用して大量のデータをエクスポートすると、パフォーマンスに影響が及びます。可能であれば OUTPUT 文はサーバと同じコンピュータ上で使用して、ネットワークを介してデータを大量に送信しないようにしてください。

1.7.4.3 UNLOAD TABLE 文を使用したデータのエクスポートに関するヒント

UNLOAD TABLE 文を使用すると、テキストフォーマットだけでデータを効率的にエクスポートできます。

UNLOAD TABLE 文では、1 行に 1 ローずつ、値をカンマで区切ってエクスポートします。再ロードを速くするために、データはプライマリキー値の順にエクスポートされます。

次の場合は、UNLOAD TABLE 文を使用します。

- テキストフォーマットでテーブル全体をエクスポートする場合
- データベースパフォーマンスを考慮する場合
- クライアントコンピュータにあるファイルにデータをエクスポートする場合

UNLOAD TABLE 文を使用するには、適切な権限を持っている必要があります。たとえば、-gl データベースサーバオプションの設定が NONE でない限り、ほとんどの場合、SELECT ANY TABLE システム権限があれば十分です。

-gl データベースサーバオプションでは、UNLOAD TABLE 文を使用できるユーザを管理します。

データベースに対する影響

OUTPUT 文、UNLOAD 文、UNLOAD TABLE 文のいずれかを選択できる場合は、パフォーマンスを考慮して UNLOAD TABLE 文を選択します。

UNLOAD TABLE 文は、アンロード中にテーブル全体に排他ロックを配置します。

例

SQL Anywhere のサンプルデータベースを使用し、次のコマンドを実行して Employees テーブルをテキストファイル `Employees.csv` にアンロードできます。

```
UNLOAD TABLE Employees TO 'C:¥¥ServerTemp¥¥Employees.csv';
```

この形式の UNLOAD TABLE 文を使用すると、ファイルパスはデータベースサーバコンピュータに対する相対パスとなります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

クライアントコンピュータ上のデータへのアクセス [631 ページ]

1.7.4.4 UNLOAD 文を使用したデータのエクスポートに関するヒント

UNLOAD 文は、クエリ結果をファイルにエクスポートするという点で OUTPUT 文に似ています。

ただし、UNLOAD 文はテキストフォーマットでより効率的にデータをエクスポートします。UNLOAD 文は、1 行に 1 ローずつ、値をカンマで区切ってエクスポートします。

次の場合は、UNLOAD 文を使用してデータをアンロードします。

- パフォーマンスが問題で、クエリ結果をエクスポートする場合
- テキストフォーマットで出力データを格納する場合
- アプリケーションにエクスポート文を埋め込む場合
- クライアントコンピュータにあるファイルにデータをエクスポートする場合

UNLOAD 文を SELECT とともに使用するには、適切な権限を持っている必要があります。たとえば、-gl データベースサーバオプションの設定が NONE でない限り、ほとんどの場合、SELECT ANY TABLE システム権限があれば十分です。少なくとも、UNLOAD 文で指定されたテーブルに対して SELECT を実行できる権限を持っている必要があります。

-gl データベースサーバオプションでは、UNLOAD 文を使用できるユーザを管理します。

データベースに対する影響

OUTPUT 文、UNLOAD 文、UNLOAD TABLE 文のいずれかを選択できる場合は、パフォーマンスを考慮して UNLOAD TABLE 文を選択します。

UNLOAD 文と SELECT は、現在の独立性レベルで実行されます。

例

SQL Anywhere のサンプルデータベースを使用し、次のコマンドを実行して Employees テーブルのサブセットをテキストファイル `GAEmployees.csv` にアンロードできます。

```
UNLOAD
SELECT * FROM Employees
WHERE State = 'GA'
TO 'C:¥¥ServerTemp¥¥GAEmployees.csv'
QUOTE '"';
```

この形式の UNLOAD TABLE 文を使用すると、ファイルパスはデータベースサーバコンピュータに対する相対パスとなります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

クライアントコンピュータ上のデータへのアクセス [631 ページ]

1.7.4.5 アンロードユーティリティ (dbunload) を使用したデータのエクスポートに関するヒント

アンロードユーティリティ (dbunload) を使用して、1 つ、複数、またはすべてのデータベーステーブルをエクスポートします。

テーブルデータとテーブルスキーマをエクスポートできます。データベースのテーブルを配置し直す場合にも dbunload を使用でき、必要な SQL スクリプトファイルを作成して必要に応じて修正します。コマンドファイルを使用すると、異なるデータベースに同じテーブルを作成できます。テーブルは、構造のみ、データのみ、または構造とデータの両方をアンロードできます。-ac オプションを使用して、既存のデータベースに直接アンロードすることもできます。

以下の目的で dbunload を使用します。

- データベースを再構築または抽出します
- 異なるファイルフォーマットにデータをエクスポートします
- 大量のデータをすばやく処理します

注記

アンロードユーティリティ (dbunload) と SQL Central のデータベースアンロードウィザードは、機能的に同じものです。どちらを使っても同じ結果が得られます。

1.7.4.6 データベースアンロードウィザードを使用したデータのエクスポートに関するヒント

データベースアンロードウィザードは、データベースを新しいデータベースにアンロードするために使用します。

データベースアンロードウィザードを使用してデータベースをアンロードする際、データベース内のすべてのオブジェクトをアンロードするのか、またはデータベースからテーブルのサブセットのみをアンロードするのかを選択できます。所有者フィルタの設定ウィンドウ内で選択したユーザ用のテーブルのみが、データベースアンロードウィザードに表示されます。特定のデータベースユーザに属するテーブルを表示する場合、アンロードするデータベースを右クリックし、所有者フィルタの設定 をクリックしてから、表示されるウィンドウでユーザを選択します。

データベースアンロードウィザードを使用すると、データベース全体をカンマ区切りのテキストフォーマットでアンロードし、データベース全体の再作成に必要な SQL スクリプトファイルを作成することもできます。これは、SQL Remote を抽出するときや、同一または少しだけ修正した構造を持つデータベースを新しく作成するときに便利です。データベースアンロードウィザードは、SQL Anywhere 内での再使用を目的として SQL Anywhere ファイルをエクスポートするときに便利です。

データベースアンロードウィザードでは、再ロードファイルではなく、既存のデータベースまたは新しいデータベースへ再ロードするオプションもあります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

> **注記**
>
> アンロードユーティリティ (dbunload) とデータベースアンロードウィザードは、機能的に同じものです。どちらを使っても同じ結果が得られます。

このセクションの内容:

データベースファイルまたは稼働しているデータベースのアンロード [621 ページ]
SQL Central で、データベースアンロードウィザードを使用して、停止または実行中のデータベースをアンロードします。

1.7.4.6.1 データベースファイルまたは稼働しているデータベースのアンロード

SQL Central で、データベースアンロードウィザードを使用して、停止または実行中のデータベースをアンロードします。

前提条件

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。

コンテキスト

> **注記**
>
> テーブルだけをアンロードするときには、テーブルを所有するユーザ ID はアンロードされません。テーブルを所有するユーザ ID を新しいデータベースに作成してからテーブルを再ロードする必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. ツール > *SQL Anywhere17* > データベースのアンロード をクリックします。
2. データベースアンロードウィザードの指示に従います。

結果

指定されたデータベースがアンロードされます。

1.7.4.7 CSV または Microsoft Excel スプレッドシートファイルへのクエリ結果のエクスポート [Interactive SQL]

OUTPUT 文を使用して、クエリ結果を Microsoft Excel ワークブックファイルまたは CSV ファイルにエクスポートします。

前提条件

問い合わせる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。

Microsoft Excel ワークブックファイルにデータをエクスポートする場合、互換性のある 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。

コンテキスト

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。Microsoft Excel ワークブックファイルの場合、WORKSHEET 句でデータのエクスポート先となるワークシートの名前を指定します。この句を省略した場合、データはファイルの最初のシートにエクスポートされます。ファイルが存在しない場合、新しいファイルが作成され、データはデフォルトのワークシートにエクスポートされます。

手順

1. Interactive SQL の *SQL* 文ウィンドウ枠にクエリを入力します。
2. クエリの最後で、次のオプションのうちいずれかを選択します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプション	OUTPUT 文の指定

テーブル全体をエクスポートする	例:
	`SELECT * FROM TableName;`
	`OUTPUT TO 'filepath';`

クエリ結果をエクスポートして別のファイルに結果を追加する	APPEND 句を次のように指定します。
	`SELECT * FROM TableName;`
	`OUTPUT TO 'filepath'`
	`APPEND;`

クエリ結果をエクスポートしてメッセージをインクルードする	VERBOSE 句を次のように指定します。
	`SELECT * FROM TableName;`
	`OUTPUT TO 'filepath'`
	`VERBOSE;`

結果とメッセージの両方を追加する	APPEND 句および VERBOSE 句を次のように指定します。
	`SELECT * FROM TableName;`
	`OUTPUT TO 'filepath'`
	`APPEND VERBOSE;`

ファイルの最初の行にカラム名を付けてクエリ結果をエクスポートする
注記
Microsoft Excel ファイルをエクスポートする場合、文は、最初の行にカラム名が含まれているものと判断します。
	WITH COLUMN NAMES 句を次のように指定します。
	`SELECT * FROM TableName;`
	`OUTPUT TO 'filepath'`
	`FORMAT TEXT`
	`QUOTE '"'`
	`WITH COLUMN NAMES;`

Microsoft Excel スプレッドシートにクエリ結果をエクスポートする	FORMAT EXCEL 句を次のように指定します。
	`SELECT * FROM TableName;`
	`OUTPUT TO 'filepath' FORMAT EXCEL`

3. SQL > 実行 をクリックします。

結果

エクスポートに成功すると、履歴タブに、クエリ結果セットのエクスポートに要した時間、エクスポートされたデータのファイル名とパス、書き込まれたローの数が表示されます。エクスポートに失敗した場合は、エクスポートに失敗したことを示すメッセージが表示されます。

例

次の文は、Customers テーブルの内容をサンプルデータベースから `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
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

Adaptive Server Enterprise の互換性 [661 ページ]
データインポート [596 ページ]

1.7.4.8 データのアンロードウィンドウを使用したデータのエクスポート

データのアンロードウィンドウを使用して SQL Central のテーブルをアンロードします。

前提条件

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。

コンテキスト

SQL Central のデータのアンロードウィンドウを使用して、データベース内の 1 つ以上のテーブルをアンロードします。この機能は、データベースアンロードウィザードまたはアンロードユーティリティ (dbunload) でも使用できますが、このウィンドウを使用すると、データベースアンロードウィザード全体を実行する代わりに 1 ステップでテーブルをアンロードできます。

手順

1. SQL Central で、*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. テーブルをダブルクリックします。
3. データをエクスポートするテーブルを右クリックして、*[データのアンロード]* をクリックします。
4. データのアンロードウィンドウの入力を完了します。*OK* をクリックします。

結果

データは、指定されたファイルに保存されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.4.9 UNLOAD 文を使用したクエリ結果のエクスポート

UNLOAD 文を使用して、Interactive SQL のクエリ結果をエクスポートします。

前提条件

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。

コンテキスト

BCP FORMAT 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。

手順

SQL 文ウィンドウ枠で、UNLOAD 文を実行します。例:

```
UNLOAD
SELECT * FROM Employees
TO 'C:¥¥ServerTemp¥¥Employees.csv';
```

エクスポートに成功すると、Interactive SQL の履歴タブに、クエリ結果セットのエクスポートに要した時間、エクスポートされたデータのファイル名とパス、書き込まれたローの数が表示されます。エクスポートに失敗した場合は、エクスポートに失敗したことを示すメッセージが表示されます。

この形式の UNLOAD TABLE 文を使用すると、ファイルパスはデータベースサーバコンピュータに対する相対パスとなります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

クエリ結果は、指定されたロケーションにエクスポートされます。

関連情報

Adaptive Server Enterprise の互換性 [661 ページ]

1.7.4.10 Interactive SQL での NULL 値の処理設定

Interactive SQL の結果ウィンドウ枠を設定することにより、OUTPUT 文を使用したときに NULL 値をどのように表すのかを指定します。

手順

1. Interactive SQL で、ツール > オプション をクリックします。
2. *SQL Anywhere* をクリックします。
3. *[結果]* タブをクリックします。
4. *Null 値の代替文字*フィールドに、NULL に使用する値を入力します。
5. *OK* をクリックします。

結果

NULL 値の代わりに表示される値が変更されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.4.11 データベースのエクスポート (SQL Central の場合)

SQL Central のデータベースアンロードウィザードを使用して、データをデータベースから再ロードファイル、新規データベース、または既存のデータベースにアンロードします。

前提条件

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。

手順

1. ツール > *SQL Anywhere17* > データベースのアンロード をクリックします。
2. データベースアンロードウィザードの指示に従います。

結果

データは、指定されたロケーションにアンロードされます。

関連情報

データベースのエクスポート (コマンドラインの場合) [628 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.4.12 データベースのエクスポート (コマンドラインの場合)

コマンドラインのアンロードユーティリティ (dbunload) を使用して、データをデータベースから再ロードファイル、新規データベース、または既存のデータベースにアンロードします。

前提条件

再ロードを伴わないアンロードの場合は、SELECT ANY TABLE システム権限が必要です。再ロードを伴うアンロードの場合は、SELECT ANY TABLE システム権限と SERVER OPERATOR システム権限が必要です。

手順

アンロード (dbunload) ユーティリティを実行し、-c オプションを使用して接続パラメータを指定します。

オプション	アクション
データベース全体のアンロード	データベース全体をサーバコンピュータ上のディレクトリ `C:¥ServerTemp¥DataFiles` にアンロードするには、次の手順に従います。 `dbunload -c "DBN=demo;UID=DBA;PWD=sql" C:¥ServerTemp¥DataFiles`
データのみエクスポート	-d オプションと -ss オプションを使用します。次に例を示します。 `dbunload -c "DBN=demo;UID=DBA;PWD=sql" -d -ss C:¥ServerTemp ¥DataFiles`
スキーマのみエクスポート	-n オプションを使用します。次に例を示します。 `dbunload -c "DBN=demo;UID=DBA;PWD=sql" -n`

スキーマの再作成とテーブルの再ロードに必要な文は、クライアントのカレントディレクトリの `reload.sql` に記述されています。

結果

データは、指定されたロケーションにアンロードされます。

関連情報

データベースのエクスポート (SQL Central の場合) [627 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.4.13 テーブルのエクスポート (SQL の場合)

Interactive SQL から UNLOAD TABLE 文を実行して、テーブルをエクスポートします。

前提条件

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。

コンテキスト

テーブル内のすべてのデータを選択して、クエリ結果をエクスポートすることにより、テーブルをエクスポートします。

手順

UNLOAD TABLE 文を実行します。例:

```
UNLOAD TABLE Departments
TO 'C:¥¥ServerTemp¥¥Departments.csv';
```

この文は、SQL Anywhere サンプルデータベースの Departments テーブルを、クライアントコンピュータではなくデータベースサーバコンピュータのディレクトリにある `Departments.csv` ファイルにアンロードします。このファイルのパスは SQL リテラルで指定されるため、円記号は 2 つ重ねることによりエスケープされ、'¥n' または '¥x' などのエスケープシーケンスの変換が回避されます。

テーブルの各ローは出力ファイルの 1 行に書き出されます。また、カラム名はエクスポートされません。各カラムはカンマで区切られます。デリミタに使う文字は、DELIMITED BY 句で変更できます。フィールドは固定幅ではありません。エクスポートされるのは、各エントリの文字だけで、カラム幅全体ではありません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

データが指定されたファイルにエクスポートされます。

関連情報

エクスポートウィザードを使用したデータのエクスポート [616 ページ]

1.7.4.14 テーブルのエクスポート (コマンドラインの場合)

コマンドラインのアンロードユーティリティ (dbunload) を実行して、テーブルをエクスポートします。

前提条件

再ロードを伴わないアンロードの場合は、SELECT ANY TABLE システム権限が必要です。再ロードを伴うアンロードの場合は、SELECT ANY TABLE システム権限と SERVER OPERATOR システム権限が必要です。

コンテキスト

カンマ (,) をデリミタとしてテーブル名を区切り、複数のテーブルをアンロードします。

手順

次のコマンドを実行します。

```
dbunload -c "DBN=demo;UID=DBA;PWD=sql" -t Employees C:¥ServerTemp¥DataFiles
```

このコマンドの -c ではデータベース接続パラメータを指定し、-t ではエクスポートする 1 つまたは複数のテーブルの名前を指定します。この dbunload コマンドは、データを SQL Anywhere サンプルデータベース (デフォルトのデータベースサーバ上で実行されていると仮定) からサーバコンピュータ上の `C:¥ServerTemp¥DataFiles` ディレクトリのファイルセットにアンロードします。データファイルからテーブルを再構築するのに必要な SQL スクリプトファイルは、デフォルト名 `reload.sql` としてクライアントのカレントディレクトリに作成されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

データは、指定されたロケーションにエクスポートされます。

1.7.5 クライアントコンピュータ上のデータへのアクセス

ファイルをデータベースサーバコンピュータにコピーしたり、データベースサーバコンピュータからファイルをコピーすることなく、SQL の文と関数を使用して、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したり、ファイルにデータをアンロードすることができます。

これを実行するために、データベースサーバは Command Sequence 通信プロトコル (CmdSeq) ファイルハンドラを使用して転送を開始します。CmdSeq ファイルハンドラは、データベースサーバがクライアントコンピュータとのデータの転送を必要としているクライアントアプリケーションから要求を受信した後、応答を送信する前に呼び出されます。このファイルハンドラは、クライアントの複数のファイルの同時転送およびインターリーブ転送を任意の時点でサポートしています。たとえば、クライアントアプリケーションにより実行された文で複数のファイルの同時転送が必要な場合、データベースサーバはそれを開始することができます。

CmdSeq ファイルハンドラを使用してクライアントデータの転送を実行することにより、アプリケーションは特殊なコードを使用する必要がなく、次に示す SQL コンポーネントを使用した機能をすぐに利用することができます。

READ_CLIENT_FILE 関数

READ_CLIENT_FILE 関数は、クライアントコンピュータ上の指定したファイルからデータを読み込み、ファイルの内容を表す LONG BINARY 値を返します。この関数は、BLOB が使用可能な SQL コードの任意の位置で使用できます。実体化の実行を文で明示的に指定していない限り、READ_CLIENT_FILE 関数によって返されるデータはメモリ内で実体化されません。たとえば、LOAD TABLE 文は実体化を行わずにクライアントファイルからデータをストリーミングします。READ_CLIENT_FILE 関数によって返された値を接続変数に割り当てると、データベースサーバはクライアントファイルの内容を取得して実体化します。

WRITE_CLIENT_FILE 関数

WRITE_CLIENT_FILE 関数は、クライアントコンピュータにある、指定したファイルにデータを書き込みます。

READ CLIENT FILE システム権限

READ CLIENT FILE システム権限によって、クライアントコンピュータのファイルを読み込むことができます。

WRITE CLIENT FILE システム権限

WRITE CLIENT FILE システム権限によって、クライアントコンピュータのファイルに書き込むことができます。

LOAD TABLE ...USING CLIENT FILE 句

USING CLIENT FILE 句を使用すると、クライアントコンピュータにあるファイル内のデータを使用してテーブルをロードできます。たとえば、`LOAD TABLE ... USING CLIENT FILE 'my-file.txt';` と指定すると、クライアントコンピュータからファイル `my-file.txt` がロードされます。

LOAD TABLE ...USING VALUE 句

USING VALUE 句を使用すると、BLOB 式を値として指定できます。BLOB 式では、READ_CLIENT_FILE 関数を使用してクライアントコンピュータ上のファイルから BLOB をロードできます。たとえば、`LOAD TABLE ... USING VALUE READ_CLIENT_FILE( 'my-file' )` と指定します。`my-file` はクライアントコンピュータ上のファイルです。

UNLOAD TABLE ...INTO CLIENT FILE 句

INTO CLIENT FILE 句を使用すると、データのアンロード先のクライアントコンピュータにあるファイルを指定できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

UNLOAD TABLE ...INTO VARIABLE 句

INTO VARIABLE 句を使用すると、データのアンロード先として変数を指定できます。

read_client_file および write_client_file セキュリティ機能

read_client_file および write_client_file セキュリティ機能は、クライアントファイルの読み込みまたは書き込みを行う文の使用を制御します。

プロシージャからのクライアントファイルの読み込みと書き込みを許可するには、関数またはその他の間接文、コールバック関数を登録します。コールバック関数が呼び出され、直接要求していないクライアント転送をアプリケーションが許可することが確認されます。

このセクションの内容:

クライアント側データセキュリティ [632 ページ]
クライアントファイルの転送に関して、データベースサーバコンピュータとは異なる場所に存在することの多いクライアントコンピュータ上のデータを不正に転送できないようにする複数のメカニズムが用意されています。

クライアント側データのロード時のリカバリ [633 ページ]
トランザクションログから LOAD TABLE 文のリカバリが必要になった場合、データのロードに使用したクライアントコンピュータ上のファイルはすでに使用できないか、変更されている可能性が高いため、元のデータは使用できなくなっています。

1.7.5.1 クライアント側データセキュリティ

クライアントファイルの転送に関して、データベースサーバコンピュータとは異なる場所に存在することの多いクライアントコンピュータ上のデータを不正に転送できないようにする複数のメカニズムが用意されています。

そのために、データベースサーバは各文の実行元を追跡し、文がクライアントアプリケーションから直接受信されたものかどうかを判断します。クライアントからの新しいファイルの転送を開始する際に、データベースサーバは文の実行元に関する情報を付加します。文がクライアントアプリケーションから直接送信された場合は、CmdSeq ファイルハンドラによってファイルの転送が許可されます。文がクライアントアプリケーションから直接送信されたものではない場合、アプリケーションは検証のコールバックを登録する必要があります。コールバックが登録されないと、転送が拒否されて文が失敗し、エラーが発生します。

また、接続が正常に確立されるまで、クライアントデータの転送は許可されません。この制限により、接続文字列やログインプロシージャを使用した不正アクセスを防止できます。

許可されたユーザを装ってシステムへのアクセスを試みるユーザからの保護を実現するには、転送データの暗号化を行ってください。

また、SQL Anywhere では、さまざまなレベルでアクセスを制御する次のセキュリティメカニズムも提供しています。

サーバレベルのセキュリティ

read_client_file および write_client_file セキュリティ機能により、サーバワイドでクライアント側の転送をすべて無効にできます。

アプリケーションレベルおよび **DBA** レベルのセキュリティ

allow_read_client_file および allow_write_client_file データベースオプションにより、データベース、ユーザ、接続レベルでのアクセス制御を実現します。たとえば、アプリケーションの接続後にこのデータベースオプションを OFF に設定することにより、アプリケーションがクライアント側の転送に使用されるのを防止できます。

ユーザレベルのセキュリティ

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

READ CLIENT FILE および WRITE CLIENT FILE システム権限は、それぞれクライアントコンピュータからのデータの読み込みと、クライアントコンピュータへのデータの書き込みに関して、ユーザレベルのアクセス制御を提供します。

1.7.5.2 クライアント側データのロード時のリカバリ

トランザクションログから LOAD TABLE 文のリカバリが必要になった場合、データのロードに使用したクライアントコンピュータ上のファイルはすでに使用できないか、変更されている可能性が高いため、元のデータは使用できなくなっています。

このような状況が発生するのを防ぐため、ロギングがオフになっていないことを確認してください。また、データのロード時に WITH ROW LOGGING または WITH CONTENT LOGGING 句のいずれかを指定してください。これらの句を指定することにより、ロードしているデータがトランザクションログに記録されるため、リカバリ時にトランザクションログのリプレイが可能になります。

WITH ROW LOGGING では、挿入された各ローがトランザクションログに INSERT 文として記録されます。WITH CONTENT LOGGING では、データベースサーバがリカバリ時に処理できるように、挿入されたデータはチャンク単位でトランザクションログに記録されます。どちらの方法も、リカバリ時にクライアント側データをロードできるようにするには適しています。ただし、同期を行っているデータベースにデータをロードする場合は、WITH CONTENT LOGGING は使用できません。

次のいずれかの LOAD TABLE 文を指定する際に、ロギングレベルが指定されていない場合は、WITH CONTENT LOGGING がデフォルトの動作となります。

- LOAD TABLE...USING CLIENT FILE `client-filename-expression`
- LOAD TABLE...USING VALUE `value-expression`
- LOAD TABLE...USING COLUMN `column-expression`

1.7.6 データベースの再構築

データベースの再構築は、データベースのアンロードと再ロードを伴うインポートとエクスポートの一種です。

再構築 (アンロード/ロード) ツールと抽出ツールを使用して、データベースの再構築、既存のデータベースの一部からの新しいデータベースの作成、未使用のページの排除を行います。

データベースは、SQL Central から再構築したり、dbunload を使用して再構築したりすることができます。

> **注記**
>
> データベースを再構築する場合、特に元のデータベースを再構築したデータベースに置き換える場合は、再構築を実行する前にデータベースのバックアップを作成するようにしてください。

インポートとエクスポートの場合、データの送信先はデータベース内またはデータベース外になります。インポートでは、データはデータベースに読み込まれます。エクスポートでは、データはデータベースから書き出されます。情報を、SQL Anywhere 以外の別のデータベースから受け取ったり、別のデータベースに送信したりすることがよくあります。

暗号化オプション -ek、-ep または -et を指定する場合は、`reload.sql` ファイルの LOAD TABLE 文に暗号化キーを含めます。ハードコーディングされたキーはセキュリティの低下を招くため、`reload.sql` ファイルのパラメータには暗号化キーが指定されます。Interactive SQL 内で `reload.sql` ファイルを実行する際に、パラメータとして暗号化キーを指定してください。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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ロードとアンロードでは、SQL Anywhere のデータベースからデータとスキーマを取り出し、それらを別の SQL Anywhere データベースに入れ直します。アンロードプロシージャでは、データファイルと、テーブルを正確に再作成するために必要なテーブル定義を含む `reload.sql` ファイルが生成されます。`reload.sql` スクリプトを実行すると、テーブルが再作成され、そこに元のデータがロードされます。

データベースの再構築には時間がかかる可能性があり、大量のディスク領域が必要になることがあります。また、データベースのアンロードと再ロード中は、そのデータベースを使用できません。このため、明確な目的がないかぎり、運用環境でデータベースを再構築しないでください。

特定の **SQL Anywhere** データベースから別の **SQL Anywhere** データベースへ

再構築では、通常 SQL Anywhere データベースからデータをコピーし、それを別の SQL Anywhere データベースに再ロードします。アンロードと再ロードは関連しています。通常はどちらか一方ではなく、両方を行います。

再構築とエクスポート

再構築がエクスポートとは異なる点は、再構築では、データの他にテーブル定義とスキーマのエクスポートとインポートが行われることです。再構築処理のアンロード部分では、テキストフォーマットのデータファイルと、テーブルとその他の定義が含まれる `reload.sql` ファイルが生成されます。`reload.sql` スクリプトを実行すると、テーブルが再作成され、そこにデータがロードされます。

SQL Remote または Mobile Link を使用している場合は、(古いデータベースから新しいデータベースを作成して) データベースを抽出することを検討します。

レプリケートするデータベースの再構築

データベース再構築のプロシージャは、データベースがレプリケーションに関連するかしないかによって異なります。データベースがレプリケーションに関連する場合は、操作中はトランザクションログのオフセットを保存しておいてください。これは、Message Agent がこの情報を必要とするためです。データベースがレプリケーションに関連しない場合、処理はもっと簡単です。

このセクションの内容:

データベースを再構築する理由 [635 ページ]

さまざまな理由で、データベースの再構築を検討します。

アンロードユーティリティを使用したデータベースの再構築に関するヒント [636 ページ]

アンロードユーティリティ (dbunload) を使用して、データベース全体をカンマ区切りのテキストフォーマットでアンロードし、データベース全体の再作成に必要な SQL スクリプトファイルを作成することもできます。

dbunload -ao を使用した最短ダウン時間でのデータベースの再構築 [637 ページ]

`dbunload -ao` を使用して、最短ダウン時間で運用データベースを再構築します。

dbunload -aob を使用した最短ダウン時間でのデータベースの再構築 [638 ページ]

`dbunload -aob` を使用して、最短ダウン時間で運用データベースを再構築します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

高可用性システムを使用した最短ダウン時間でのデータベースの再構築 [640 ページ]
動作中の高可用性システムを使用し、再構築されたデータベースに切り替えます。

同期やレプリケーションに関連しないデータベースの再構築 [642 ページ]
アンロードユーティリティ (dbunload) を使用して、データベースをアンロードして新規データベースに再構築したり、既存のデータベースに再ロードしたり、既存のデータベースを置き換えたりします。

同期やレプリケーションに関連するデータベースの再構築 (dbunload の場合) [643 ページ]
dbunload -ar オプションを使用して、同期やレプリケーションに関連するデータベースを再構築します。その際は、同期やレプリケーションに干渉しない方法でデータベースがアンロードおよび再ロードされます。

同期やレプリケーションに関連するデータベースの再構築 (手動の場合) [644 ページ]
同期やレプリケーションに関連するデータベースを再構築します。

UNLOAD TABLE 文を使用したデータベースの再構築に関するヒント [645 ページ]
UNLOAD TABLE 文を使用すると、特定の文字コードでデータを効率的にエクスポートできます。

テーブルデータのエクスポート [646 ページ]
アンロードユーティリティ (dbunload) を使用して、テーブルデータをアンロードします。

テーブルスキーマのエクスポート [647 ページ]
アンロードユーティリティ (dbunload) を使用して、テーブルスキーマをアンロードします。

データベースの再ロード [648 ページ]
データベースは、コマンドラインから再ロードします。

データベース再構築時のダウン時間の最短化 [649 ページ]
バックアップユーティリティ (dbbackup) およびログトランザクションユーティリティ (dbtran) を使用して、データベースを再構築する際のダウン時間を最短にします。

関連情報

データベース抽出 [650 ページ]
同期やレプリケーションに関連しないデータベースの再構築 [642 ページ]
マテリアライズドビューの手動での再表示 [74 ページ]

1.7.6.1 データベースを再構築する理由

さまざまな理由で、データベースの再構築を検討します。

次の処理を行う場合は、データベースを再構築します。

データベースのファイルフォーマットのアップグレード

アップグレードユーティリティを適用すると一部の新機能が使用可能になります。ただし、データベースのファイルフォーマットのアップグレードを必要とする機能もあります。ファイルフォーマットのアップグレードとは、データベースをアンロードして再ロードすることです。

新しいバージョンの SQL Anywhere データベースサーバは、データベースをアップグレードしないで使用できます。新しいシステムテーブルまたはデータベースオプションにアクセスする必要がある新しいバージョンの機能を使用する場合

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

は、アップグレードユーティリティを使用してデータベースをアップグレードしてください。アップグレードユーティリティでは、データをアンロードまたは再ロードしません。

データベースファイルフォーマットの変更を伴う新しいバージョンの SQL Anywhere を使用する場合は、データベースをアンロードして再ロードしてください。データベースをバックアップしてから再構築してください。

> **注記**
>
> バージョン 9 より前からアップグレードする場合は、データベースファイルを再構築する必要があります。バージョン 10.0.0 以降からアップグレードする場合は、アップグレードユーティリティを使用するか、データベースを再構築します。

ディスク領域を再利用する場合

データを削除しても、データベースは縮小されません。代わりに、空のページが、再使用できるように空き領域としてマーク付けされます。データベースを再構築しないかぎり、空のページがデータベースから削除されることはありません。データベースからデータを大量に削除し、データをそれ以上追加しない場合は、データベースを再構築するとディスク領域を再利用できます。

データベースパフォーマンスを向上させる場合

データベースを再構築すると、パフォーマンスが向上する場合があります。プライマリキーの順にデータベースをアンロードして再ロードできるので、関連するローが同じページまたは周辺のページに表示されるため、関連情報に速くアクセスできます。

> **注記**
>
> テーブルが極端に断片化されているためにパフォーマンスが低下していることが判明した場合は、テーブルを再編成します。

1.7.6.2 アンロードユーティリティを使用したデータベースの再構築に関するヒント

アンロードユーティリティ (dbunload) を使用して、データベース全体をカンマ区切りのテキストフォーマットでアンロードし、データベース全体の再作成に必要な SQL スクリプトファイルを作成することもできます。

たとえば、SQL Remote 抽出を作成したり、同一または少しだけ修正した構造を持つデータベースを新しく作成するために、これらのファイルを使用できます。

以下の目的で、アンロードユーティリティ (dbunload) を使用します。

- データベースを再構築する、またはデータベースからデータを抽出します
- 複数のファイルフォーマットでエクスポートします
- 大量のデータをすばやく処理します

> **注記**
>
> アンロードユーティリティ (dbunload) とデータベースアンロードウィザードは、機能的に同じものです。どちらを使っても同じ結果が得られます。Interactive SQL の SQL OUTPUT 文または SQL UNLOAD 文を使用することによっても、データベースをアンロードできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

同期やレプリケーションに関連するデータベースの再構築 (dbunload の場合) [643 ページ]

1.7.6.3 dbunload -ao を使用した最短ダウン時間でのデータベースの再構築

`dbunload -ao` を使用して、最短ダウン時間で運用データベースを再構築します。

前提条件

次の条件を満たす必要があります。

- 元のデータベースが SQL Anywhere バージョン 17 で作成されています。
- 再構築されたデータベースのページサイズ、暗号化アルゴリズム、暗号化キーが元のデータベースと同じです。
- 中間ファイルが必要なため、再構築を実行するコンピュータに、元のデータベースのデータベース、DB 領域、ログファイルの合計の 2 倍のサイズを保持するための十分な空き領域があります。
- 現在のデータベースで使用されている DB 領域がある場合、その DB 領域ファイルはデータベースファイルと同じディレクトリに存在します。また、絶対パスを使用していません。
- データベースが高可用性機能を使用していません。

以下の条件を満たすことができない場合、dbunload -aob を使用したデータベースの再構築を検討します。

- トランザクションが失われないように、`dbbackup -wa` を使用して最初のバックアップを取る必要があるため、運用サーバに未処理のトランザクションがない状態が一定時間なくてはなりません。
- 運用サーバへの複数回のバックアップや、運用サーバでのトランザクションログの名前変更は許容される必要があります。

手順

1. 次のコマンドと同様のコマンドを実行し、`filename.db` という名前の再構築データベースを作成します。

   ```
   dbunload -c connection-string-to-production-database -ao filename.db
   ```

   必要に応じて -aot および -dt オプションを指定し、それぞれ、ダウン時間をさらに短くしたり、テンポラリディレクトリを設定したりすることができます。

   このコマンドの実行中、運用データベースでは、バックアップやトランザクションログの名前変更が多数発生します。
2. 運用コンピュータ以外のコンピュータで再構築を実行した場合、再構築されたデータベースファイルを現在の運用データベースから別のディレクトリにコピーするのではなく、運用コンピュータにコピーします。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

3. データベースがトランザクションログに基づく同期 (Mobile Link、SQL Remote、データベースミラーリング、または DBLTM) に関連している場合、名前が変更されたトランザクションログすべてを、運用データベースから再構築されたデータベースと同じディレクトリにコピーします。
4. 運用データベースを停止します。
5. 再構築されたデータベースと同じ運用コンピュータ上のディレクトリに、現在の運用データベースのトランザクションログをコピーします。
6. 再構築されたデータベースファイルを運用データベースとして再起動し、今後運用サーバを再起動するときに再構築されたデータベースが使用されるようにします。必要に応じて、運用サーバの起動に使用される任意のスクリプトまたはサービスを変更し、元の運用データベースファイルの代わりに再構築されたデータベースファイルを参照するようにします。

結果

運用データベースが最短ダウン時間で再構築されます。

1.7.6.4 dbunload -aob を使用した最短ダウン時間でのデータベースの再構築

`dbunload -aob` を使用して、最短ダウン時間で運用データベースを再構築します。

前提条件

次の条件を満たす必要があります。

- 元のデータベースが SQL Anywhere バージョン 17 によって作成されています。
- 再構築されたデータベースのページサイズ、暗号化アルゴリズム、暗号化キーが元のデータベースと同じです。
- 中間ファイルが必要なため、再構築を実行するコンピュータに、元のデータベースのデータベース、DB 領域、トランザクションログファイルの合計の 2 倍のサイズを保持するための十分な空き領域があります。
- 現在のデータベースで使用されている DB 領域がある場合、その DB 領域ファイルはデータベースファイルと同じディレクトリに存在します。また、絶対パスを使用しません。
- データベースが高可用性機能を使用していません。

手順

1. バックアップデータベースを作成し、最短ダウン時間でデータベースを再構築する際のソースデータベースとして使用します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

オプション	アクション
Interactive SQL でサーバ側のバックアップを作成する場合	WAIT AFTER END 句を使用して BACKUP DATABASE 文を実行します。
コマンドラインを使用してバックアップを作成する場合	-ws および -wa オプションを使用して dbbackup ユーティリティを実行します。
運用データベースに常に **1** つ以上の未処理トランザクションがある場合 **(**バックアップでは未処理トランザクションを無期限に待機します**)**	1. 運用データベースを停止します。データベースは完全に停止する必要がありますが、サーバプロセスは終了する必要はありません。 2. 別のディレクトリに運用データベースとトランザクションログをコピーします。これは、`dbunload -aob` 実行時に使用されるバックアップです。 3. 運用トランザクションログファイルの名前を変更します。 4. 運用データベースを再起動します。データベースを再起動すると、新しいトランザクションログファイルが作成されます。

`dbunload -aob` を実行するか、トランザクションログの終了オフセットが変更されるまで、バックアップデータベースを起動しないでください。現在のトランザクションログの名前変更後、終了オフセットは開始オフセットと完全に一致する必要があります。

2. 次のコマンドを実行し、バックアップ `path-to-backupsource.db` から `filename.db` という名前の再構築されたデータベースを作成します。

```
dbunload -aob filename.db -c ...;DBF=path-to-backupsource.db
```

接続文字列には、データベースファイル (DBF) の接続パラメータが含まれている必要があります。

3. ダウン時間低減のため、トランザクションログの名前を変更して、運用データベースのインクリメンタルバックアップを実行します。例:

```
dbbackup -c connection-string -t -r -n directory
```

4. 次のコマンドを実行して、再構築されたデータベースにインクリメンタルバックアップを適用します。ここで、`filename.log` はインクリメンタルバックアップにより作成されたものです。

```
dbeng17 filename.db -a filename.log
```

5. (オプション) 手順 3 と 4 を数回繰り返すと、さらにダウン時間が低減されます。
6. 運用データベースと同じコンピュータ上の別のディレクトリに、再構築されたデータベースをコピーします。
7. データベースがトランザクションログに基づく同期 (Mobile Link、SQL Remote、またはデータベースミラーリング) に関連している場合、名前が変更されたトランザクションログすべてを、運用データベースから再構築されたデータベースと同じディレクトリにコピーします。
8. 運用データベースを停止します。
9. 再構築されたデータベースと同じディレクトリに、現在の運用データベースのトランザクションログをコピーします。
10. 次のコマンドを実行して、再構築されたデータベースにトランザクションログのコピーを適用します。

```
dbeng17 filename.db -a filename.log
```

11. 再構築されたデータベースファイルを運用データベースとして再起動し、今後運用サーバを再起動するときに再構築されたデータベースが使用されるようにします。必要に応じて、運用サーバの起動に使用される任意のスクリプトまたはサービスを変更し、古い運用データベースファイルの代わりに再構築されたデータベースファイルを参照するようにします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

運用データベースが最短ダウン時間で再構築されます。

次のステップ

運用データベースが再構築されたデータベースに正常に置き換わるまで、-a データベースを使用せずに再構築されたデータベースを起動しないでください。-a データベースオプションを使用せずに再構築されたデータベースを起動すると、少なくとも、再構築されたデータベースでチェックポイント操作が実行され、運用データベースから再構築されたデータベースにトランザクションログを適用することができなくなります。

1.7.6.5 高可用性システムを使用した最短ダウン時間でのデータベースの再構築

動作中の高可用性システムを使用し、再構築されたデータベースに切り替えます。

前提条件

次の条件を満たす必要があります。

- 元のデータベースが SQL Anywhere バージョン 17 以降によって作成されています。
- 再構築されたデータベースのページサイズ、暗号化アルゴリズム、暗号化キーが元のデータベースと同じです。
- 中間ファイルが必要なため、再構築を実行するコンピュータに、元のデータベースのデータベース、DB 領域、トランザクションログファイルの合計の 2 倍のサイズを保持するための十分な空き領域があります。
- 現在のデータベースで使用されている DB 領域がある場合、その DB 領域ファイルはデータベースファイルと同じディレクトリに存在します。また、絶対パスを使用しません。

以下の条件を満たすことができない場合、dbunload -aob を使用したデータベースの再構築を検討します。

- トランザクションが失われないように、`dbbackup -wa` を使用して最初のバックアップを取る必要があるため、運用サーバに未処理のトランザクションがない状態が一定時間なくてはなりません。
- 運用サーバへの複数回のバックアップや、運用サーバでのトランザクションログの名前変更は許容される必要があります。

手順

1. 次のコマンドを実行します。ここで `filename.db` とは再構築するデータベースの名前です。

```
dbunload -c connection-string-to-production-primary-database -ao filename.db
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

必要に応じて、-dt オプションを使用してテンポラリディレクトリを指定します。

このコマンドの実行中、運用データベースでは、バックアップやトランザクションログの名前変更がいくつか発生します。

2. プライマリサーバへの接続時に次の文を実行して、プライマリサーバが監視サーバに接続されるようにします。

```
SELECT DB_PROPERTY ( 'ArbiterState' );
```

プライマリサーバが監視サーバに接続されない場合、ミラーデータベースが停止するとプライマリデータベースも停止します。

3. 次の文を実行し、同期された結果を返すようにします。

```
SELECT DB_PROPERTY ( 'MirrorState' );
```

ミラーが同期されると、少なくとも現在のプライマリトランザクションログファイルの一部がミラーで使用されます。

4. ミラーサーバ以外のコンピュータで手順 1 を実行した場合、filename.db をミラーサーバコンピュータにコピーします。filename.db は現在のミラーデータベース以外のディレクトリに配置する必要があるためです。
5. ミラーサーバで動作しているデータベースを停止します。
6. 現在のトランザクションログを、ミラーから filename.db と同じディレクトリにコピーします。
7. ユーティリティデータベースに接続し、次の文を実行して、ミラーサーバ上の再構築されたデータベースを停止します。

```
START DATABASE database-file MIRROR ON AUTOSTOP OFF;
```

8. ミラーサーバがプライマリサーバからの変更をすべて適用し、同期するのを待機します。たとえば、次の文を実行し、同期された結果を返すようにします。

```
SELECT DB_PROPERTY ( 'MirrorState' );
```

9. ダウン時間が許容範囲である場合、プライマリサーバへの接続時に次の文を実行することで、再構築されたデータベースのパートナサーバがプライマリサーバとなります。

```
ALTER DATABASE SET PARTNER FAILOVER;
```

ダウン時間は通常 1 分以内です。

10. 再構築されたデータベースのパートナサーバがプライマリサーバを引き継ぐと、新しいミラーパートナサーバで、新しいミラーサーバから新しいプライマリデータベース (再構築および同期されたもの) のバックアップを作成します。
11. 新しいミラーサーバの元のデータベースを停止します。
12. ユーティリティデータベースに接続し、次の文を実行して、新しいミラーサーバの再構築されたデータベースのバックアップを停止します。

```
START DATABASE database-file MIRROR ON AUTOSTOP OFF;
```

13. パートナサーバの起動に使用される任意のスクリプトまたはサービスを変更し、両方のパートナで、古い運用データベースではなく再構築されたデータベースを参照するようにします。

結果

運用データベースを再構築し、高可用性システムでフェイルオーバー機能を使用して、再構築されたデータベースを新しいプライマリデータベースとして最短のダウン時間で設定します。

 © 2016 SAP SE or an SAP affiliate company. All rights reserved.

> **i 注記**
>
> ミラーデータベースが再構築されたデータベースに代わることで、高可用性システムですべてのオペレーションを適用することができます。これにより再構築にかかるダウン時間を低減する一方、ミラーがプライマリに同期およびオペレーションを適用している間は高可用性システムは使用されません。これは、プライマリはミラーが同期しない限りフェイルオーバーできないためです。

1.7.6.6 同期やレプリケーションに関連しないデータベースの再構築

アンロードユーティリティ (dbunload) を使用して、データベースをアンロードして新規データベースに再構築したり、既存のデータベースに再ロードしたり、既存のデータベースを置き換えたりします。

前提条件

同期やレプリケーションに関連しないデータベースの場合にのみ、次の手順を使用してください。

SELECT ANY TABLE および SERVER OPERATOR システム権限が必要です。

コンテキスト

-an オプションと -ar オプションは、パーソナルサーバへの接続、または共有メモリ経由によるネットワークサーバへの接続にのみ適用されます。-ar や -an オプションを指定すると、SQL Central でデータベースアンロードウィザードを使用した場合より高速で処理できますが、-ac を指定すると処理はデータベースアンロードウィザードよりも遅くなります。

dbunload の他のオプションを使用して、実行中のデータベースまたは実行されていないデータベースや、データベースパラメータを指定します。

手順

1. 次のいずれかのオプションを指定してアンロードユーティリティ (dbunload) を実行します。

オプション	アクション
新規データベースへの再構築	-an
既存のデータベースへの再ロード	-ac
既存のデータベースの置換	-ar

これらのオプションの 1 つを使用した場合、ディスク上にデータの中間コピーが作成されないため、コマンドラインではアンロードディレクトリを指定する必要なありません。このため、データのセキュリティが向上します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. 再ロードしたデータベースを使用する前に、データベースを停止し、トランザクションログを圧縮します。

結果

データベースは、指定されたロケーションにアンロードされます。

1.7.6.7 同期やレプリケーションに関連するデータベースの再構築 (dbunload の場合)

dbunload -ar オプションを使用して、同期やレプリケーションに関連するデータベースを再構築します。その際は、同期やレプリケーションに干渉しない方法でデータベースがアンロードおよび再ロードされます。

前提条件

SELECT ANY TABLE および SERVER OPERATOR システム権限が必要です。

すべてのサブスクリプションを同期してから、Mobile Link 同期に参加するデータベースを再構築する必要があります。

コンテキスト

この作業の内容は、SQL Anywhere の Mobile Link クライアント (dbmlsync を使用するクライアント) と SQL Remote に適用されます。

同期やレプリケーションは、トランザクションログのオフセットに基づいています。データベースを再構築すると、古いトランザクションログのオフセットは新しいログのオフセットとは異なるため、古いログは使用できなくなります。このため、同期やレプリケーションに参加しているときは特に、データベースを確実にバックアップすることが重要です。

> **注記**
>
> dbunload の他のオプションを使用して、実行中のデータベースまたは実行されていないデータベースや、データベースパラメータを指定します。

手順

1. データベースを停止します。
2. データベースファイルとトランザクションログファイルを安全なロケーションにコピーしてオフラインでフルバックアップを実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

3. 次 dbunload のコマンドを実行して、データベースを再構築します。

```
dbunload -c connection-string -ar directory
```

connection-string は適切な権限による接続を表し、directory はレプリケーション環境で使用した古いトランザクションログのディレクトリを表します。データベースへのその他の接続はありません。

-ar オプションは、パーソナルサーバへの接続、または共有メモリ経由によるネットワークサーバへの接続にのみ適用されます。
4. 新しいデータベースを停止してから、データベースの復元後に通常実行する妥当性検査を実行します。
5. 必要な運用オプションを使用してデータベースを起動します。これで、再ロードされたデータベースにアクセスできます。

結果

データベースは再ロードされ、起動します。

1.7.6.8 同期やレプリケーションに関連するデータベースの再構築 (手動の場合)

同期やレプリケーションに関連するデータベースを再構築します。

前提条件

データベースを再構築するには、SELECT ANY TABLE および SERVER OPERATOR システム権限が必要です。

すべてのサブスクリプションを同期してから、Mobile Link 同期に参加するデータベースを再構築する必要があります。

コンテキスト

この作業の内容は、SQL Anywhere の Mobile Link クライアント (dbmlsync を使用するクライアント) と SQL Remote に適用されます。

同期やレプリケーションは、トランザクションログのオフセットに基づいています。データベースを再構築すると、古いトランザクションログのオフセットは新しいログのオフセットとは異なるため、古いログは使用できなくなります。このため、同期やレプリケーションに参加しているときは特に、データベースを確実にバックアップすることが重要です。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. データベースを停止します。
2. データベースファイルとトランザクションログファイルを安全なロケーションにコピーして、オフラインでフルバックアップを実行します。
3. dbtran ユーティリティを実行してデータベースの現在のトランザクションログファイルの開始オフセット、終了オフセット、および現在の予定表を表示します。

   終了オフセットと予定表は手順 8 で使用するために記録しておきます。
4. 現在のトランザクションログファイルの名前を変更して、アンロード処理中に修正されないようにします。このファイルを dbremote のオフラインログディレクトリに置きます。
5. データベースを再構築します。
6. 新しいデータベースを停止します。
7. 新しいデータベースで使用されていたトランザクションログファイルを消去します。
8. 新しいデータベースで dblog を使用します。手順 3 で記録した終了オフセットと予定表を -z オプションに指定し、相対オフセットを 0 に設定します。

   ```
   dblog -x 0 -z 0000698242 -ft timeline -ir -is database-name.db
   ```
9. Message Agent を実行するときは、コマンドラインに元のオフラインディレクトリのパスを入力します。
10. データベースを起動します。これで、再ロードされたデータベースにアクセスできます。

結果

データベースは再ロードされ、起動します。

1.7.6.9 UNLOAD TABLE 文を使用したデータベースの再構築に関するヒント

UNLOAD TABLE 文を使用すると、特定の文字コードでデータを効率的にエクスポートできます。

テキストフォーマットでデータをエクスポートする場合は、UNLOAD TABLE 文を使用してデータベースを再構築してください。

UNLOAD TABLE 文は、テーブル全体に排他ロックを配置します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.6.10 テーブルデータのエクスポート

アンロードユーティリティ (dbunload) を使用して、テーブルデータをアンロードします。

前提条件

問い合わせる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。

コンテキスト

スキーマの再作成と指定したテーブルの再ロードに必要な文は、現在のローカルディレクトリの `reload.sql` に記述されています。

テーブル名をカンマで区切り、複数のテーブルをアンロードします。

手順

dbunload コマンドを実行します。その際には -c オプションを使用して接続パラメータを指定し、-t オプションを使用してデータをエクスポートするテーブルを指定し、-ss オプションを使用してカラム統計を表示しないかどうかを指定し、-d オプションを使用してデータのみをアンロードするかどうかを指定します。

たとえば、Employees テーブルからデータをエクスポートするには、次のコマンドを実行します。

```
dbunload -c "DBN=demo;UID=DBA;PWD=sql" -ss -d -t Employees C:¥ServerTemp¥DataFiles
```

`reload.sql` ファイルはクライアントのカレントディレクトリに書き込まれ、Employees テーブルのデータを再ロードするために必要な LOAD TABLE 文が含まれます。データファイルはサーバディレクトリ `C:¥ServerTemp¥DataFiles` に書き込まれます。

結果

指定されたディレクトリにテーブルデータがエクスポートされます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.6.11 テーブルスキーマのエクスポート

アンロードユーティリティ (dbunload) を使用して、テーブルスキーマをアンロードします。

前提条件

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。

コンテキスト

スキーマの再作成と指定されたテーブルの再ロードに必要な文は、クライアントのカレントディレクトリの `reload.sql` に記述されています。

カンマをデリミタとしてテーブル名を区切り、複数のテーブルをアンロードします。

手順

dbunload コマンドを実行します。その際には -c オプションを使用して接続パラメータを指定し、-t オプションを使用してデータをエクスポートするテーブルを指定し、-n オプションを指定してスキーマのみをアンロードするかどうかを指定します。

たとえば、Employees テーブルからスキーマのみをエクスポートするには、次のコマンドを実行します。

```
dbunload -c "DBN=demo;UID=DBA;PWD=sql" -n -t Employees
```

結果

テーブルスキーマがエクスポートされます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.6.12 データベースの再ロード

データベースは、コマンドラインから再ロードします。

前提条件

既存の `reload.sql` ファイルが必要です。

コンテキスト

再ロードでは、空のデータベースファイルを作成し、既存の `reload.sql` ファイルを使用してスキーマを作成し、別の SQL Anywhere データベースからアンロードしたすべてのデータを、新規に作成したテーブルに挿入します。

手順

1. dbinit ユーティリティを実行して空の新しいデータベースファイルを作成します。
2. 新しいデータベースに接続します。
3. `reload.sql` スクリプトを実行します。

結果

データベースが再ロードされます。

例

次のコマンドは、`mynewdb.db` という名前のファイルを作成します。

```
dbinit -dba DBA,passwd mynewdb.db
```

次のコマンドは、`reload.sql` スクリプトを現在のディレクトリにロードして実行します。

```
dbisql -c "DBF=mynewdb;UID=DBA;PWD=passwd" reload.sql
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.6.13 データベース再構築時のダウン時間の最短化

バックアップユーティリティ (dbbackup) およびログトランザクションユーティリティ (dbtran) を使用して、データベースを再構築する際のダウン時間を最短にします。

前提条件

データベースファイルのバックアップコピーを作成してから、データベースを再構築してください。

トランザクションログ名を変更するようなバックアップが他に予定されていないことを確認します。トランザクションログ名が変更された場合は、再構築されたデータベースに、名前変更されたトランザクションログのトランザクションを適切な順序で適用してください。

BACKUP DATABASE システム権限が必要です。

コンテキスト

> **注記**
>
> データベースが SQL Anywhere 17 で作成されている場合、下記の手順ではなく、`dbunload -ao` または `dbunload -aob` を使用してください。

手順

1. `dbbackup -r -wa` を使用し、データベースとログのバックアップを作成して、アクティブなトランザクションがなくなったらトランザクションログの名前を変更します。このバックアップは、未処理のトランザクションがなくなるまで完了しません。

   > **注記**
   >
   > -wa パラメータを使用して、トランザクションログの名前を変更している間、アクティブなトランザクションが失われることを防ぎます。クライアント側のバックアップでは、dbbackup に指定されている接続文字列のデータベースバージョンは、17 である必要があります。

2. バックアップデータベースを別のコンピュータ上で再構築します。
3. 運用サーバ上で再度 dbbackup -r を実行してトランザクションログの名前を変更してください。
4. 名前を変更したトランザクションログに対して dbtran ユーティリティを実行し、再構築したデータベースにトランザクションを適用します。
5. 運用サーバを停止し、データベースとトランザクションログをコピーします。
6. 再構築したデータベースを運用サーバにコピーします。
7. 手順 5 のトランザクションログに対して dbtran を実行します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

8. 再構築されたデータベースをパーソナルサーバ (dbeng17) で開始し、ユーザが接続できないことを確認します。
9. 手順 7 のトランザクションを適用します。
10. データベースサーバを停止し、データベースをネットワークサーバ (dbsrv17) で開始して、ユーザが接続できるようにします。

結果

データベースを再構築するときのダウン時間が最小限に抑えられます。

1.7.7 データベース抽出

SQL Anywhere の統合データベースから SQL Anywhere のリモートデータベースを抽出します。

データベースの抽出は、SQL Remote で使用します。

SQL Central のデータベース抽出ウィザードまたは抽出ユーティリティを使用して、データベースを抽出できます。SQL Remote レプリケーション用に統合データベースからリモートデータベースを作成する場合は、抽出ユーティリティ (dbxtract) を使用してください。

1.7.8 SQL Anywhere へのデータベース移行

sa_migrate システムプロシージャまたはデータベース移行ウィザードを使用して、複数のソースからテーブルをインポートします。

- SAP SQL Anywhere
- SAP Ultra Light
- SAP Adaptive Server Enterprise
- SAP IQ
- SAP HANA
- SAP Advantage Database Server
- IBM DB2
- Microsoft SQL Server
- Microsoft Access
- Oracle Database
- Oracle MySQL
- リモートサーバに接続する汎用 ODBC ドライバ

データベース移行ウィザードやシステムプロシージャの sa_migrate セットを使用してデータを移行する前に、まずターゲットデータベースを作成してください。ターゲットデータベースとは、データの移行先となるデータベースのことです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

i 注記

SAP HANA テーブルを SQL Anywhere に移行する場合、インデックスはテーブルとともに移行されないため、移行後に手動で作成する必要があります。

このセクションの内容:

データベース移行ウィザードの使用 [651 ページ]
SQL Central でデータベース移行ウィザードを使用して、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、および (必要に応じて) 現在のユーザをリモートデータベースに接続するための外部ログインを作成します。

sa_migrate システムプロシージャ [652 ページ]
sa_migrate システムプロシージャを使用して、リモートデータを移行します。

1.7.8.1 データベース移行ウィザードの使用

SQL Central でデータベース移行ウィザードを使用して、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、および (必要に応じて) 現在のユーザをリモートデータベースに接続するための外部ログインを作成します。

前提条件

事前にリモートサーバを作成しておく必要があります。ターゲットデータベース内にそのテーブルを所有するユーザが必要です。

CREATE PROXY TABLE システム権限と CREATE TABLE システム権限の両方を持っているか、次のシステム権限をすべて持っている必要があります。

- CREATE ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE
- INSERT ANY TABLE
- SELECT ANY TABLE
- CREATE ANY INDEX

手順

1. SQL Central で、*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. ツール > *SQL Anywhere17* > データベースの移行 をクリックします。
3. 次へをクリックします。
4. 対象のデータベースを選択して次へをクリックします。
5. リモートデータベースへの接続に使用するリモートサーバを選択し、次へ をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

リモートサーバ用の外部ログインも作成できます。デフォルトでは、SQL Anywhere は、現在のユーザに代わってリモートサーバに接続する場合に、そのユーザのユーザ ID とパスワードを使用します。ただし、リモートサーバに現在のユーザと同じユーザ ID とパスワードで定義されたユーザがいない場合は、外部ログインを作成する必要があります。外部ログインは、現在のユーザ用に代替ログイン名とパスワードを割り当ててリモートサーバに接続できるようにします。

6. 移行するテーブルを選択し、次へ をクリックします。

   システムテーブルは移行できないので、このリストにはシステムテーブルは表示されません。
7. ターゲットデータベースのテーブルを所有するユーザを選択し、次へ をクリックします。
8. リモートテーブルからデータと外部キーを移行するかどうか、また移行プロセスのために作成されたプロキシテーブルを保持するかどうかを選択し、次へ をクリックします。
9. 完了をクリックします。

結果

指定されたテーブルが移行されます。

関連情報

リモートサーバの作成 (SQL Central の場合) [666 ページ]
外部ログインの作成 (SQL Central の場合) [682 ページ]

1.7.8.2 sa_migrate システムプロシージャ

sa_migrate システムプロシージャを使用して、リモートデータを移行します。

テーブルや外部キーのマッピングを削除する場合は、拡張メソッドを使用します。

このセクションの内容:

sa_migrate システムプロシージャを使用したすべてのテーブルの移行 [653 ページ]
sa_migrate システムプロシージャを使用して、すべてのテーブルを移行します。

データベースマイグレーションシステムプロシージャを使用した個別のテーブルの移行 [654 ページ]
データベースマイグレーションシステムプロシージャを使用して、個別のテーブルを移行します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.8.2.1 sa_migrate システムプロシージャを使用したすべてのテーブルの移行

sa_migrate システムプロシージャを使用して、すべてのテーブルを移行します。

前提条件

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)

ターゲットデータベース内に移行対象のテーブルを所有するユーザが必要です。

外部ログインを作成するには、MANAGE ANY USER システム権限が必要です。

コンテキスト

リモートデータベースで、テーブルの所有者が異なっていても名前が同じ場合、それらのテーブルはターゲットデータベースに移行すると 1 人の所有者に属します。したがって、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。

移行したテーブルすべてを、ターゲットデータベースと同じユーザによって所有されないようにする場合は、ターゲットデータベース上の所有者ごとに、`local-table-owner` 引数と `owner-name` 引数を指定して sa_migrate プロシージャを実行します。

手順

1. Interactive SQL からターゲットデータベースに接続します。
2. リモートサーバを作成してリモートデータベースに接続します。
3. (省略可) リモートデータベースに接続できるように外部ログインを作成します。この手順は、ユーザがターゲットデータベースとリモートデータベースで異なるパスワードを使用している場合、またはターゲットデータベースで使用しているユーザ ID とは異なるユーザ ID でリモートデータベースにログインする場合にのみ必要です。
4. *SQL* 文ウィンドウ枠で、sa_migrate システムプロシージャを実行します。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移行されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

例:

```
CALL sa_migrate( 'local_user1', 'rmt_server1', NULL, 'remote_user1', NULL, 1, 1,
1 );
```

結果

このプロシージャは複数のプロシージャを順番に呼び出し、指定された基準を使用してユーザ remote_user1 に属するリモートテーブルをすべて移行します。

関連情報

リモートサーバとリモートテーブルのマッピング [663 ページ]
外部ログイン [681 ページ]

1.7.8.2.2 データベースマイグレーションシステムプロシージャを使用した個別のテーブルの移行

データベースマイグレーションシステムプロシージャを使用して、個別のテーブルを移行します。

前提条件

次のシステム権限が必要です。

- CREATE ANY TABLE
- CREATE ANY INDEX
- INSERT ANY TABLE
- SELECT ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE

事前にリモートサーバを作成しておく必要があります。ターゲットデータベース内にそのテーブルを所有するユーザが必要です。

外部ログインを作成するには、MANAGE ANY USER システム権限が必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

`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。両方に NULL を指定すると、システムテーブルを含む、データベース内のすべてのテーブルが移行します。また、リモートデータベースで、テーブルの所有者が異なっていても名前が同じ場合、それらのテーブルはターゲットデータベースに移行すると 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。

手順

1. ターゲットデータベースを作成します。
2. Interactive SQL からターゲットデータベースに接続します。
3. (省略可) リモートデータベースに接続できるように外部ログインを作成します。外部ログインは、ユーザがターゲットデータベースとリモートデータベースで異なるパスワードを使用している場合、またはターゲットデータベースで使用しているユーザ ID とは異なるユーザ ID でリモートデータベースにログインする場合にのみ必要です。
4. sa_migrate_create_remote_table_list システムプロシージャを実行します。次に例を示します。

   ```
   CALL sa_migrate_create_remote_table_list( 'rmt_server1',
        NULL, 'remote_user1', 'mydb' );
   ```

   Adaptive Server Enterprise および Microsoft SQL Server データベースのデータベース名を指定してください。

   このプロシージャにより、移行するリモートテーブルのリストが dbo.migrate_remote_table_list テーブルに移植されます。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します。
5. sa_migrate_create_tables システムプロシージャを実行します。次に例を示します。

   ```
   CALL sa_migrate_create_tables( 'local_user1' );
   ```

   このプロシージャは、dbo.migrate_remote_table_list からリモートテーブルのリストを取り出し、リストにあるテーブルごとにプロキシテーブルとベーステーブルを作成します。また、移行したテーブルのプライマリキーインデックスもすべて作成します。
6. リモートテーブルからターゲットデータベース上のベーステーブルにデータを移行する場合は、sa_migrate_data システムプロシージャを実行します。次に例を示します。

   ```
   CALL sa_migrate_data( 'local_user1' );
   ```

   このプロシージャは、各リモートテーブルから、sa_migrate_create_tables プロシージャで作成されたベーステーブルにデータを移行します。

   リモートデータベースから外部キーを移行しない場合は、手順 10 に進みます。
7. sa_migrate_create_remote_fks_list システムプロシージャを実行します。次に例を示します。

   ```
   CALL sa_migrate_create_remote_fks_list( 'rmt_server1' );
   ```

   このプロシージャは、dbo.migrate_remote_table_list にリストされている各リモートテーブルに関連した外部キーのリストを、テーブル dbo.migrate_remote_fks_list に設定します。

   ローカルのベーステーブルに再作成しない外部キーマッピングを削除します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

8. sa_migrate_create_fks システムプロシージャを実行します。次に例を示します。

```
CALL sa_migrate_create_fks( 'local_user1' );
```

このプロシージャは、dbo.migrate_remote_fks_list に定義されている外部キーマッピングをベーステーブル上に作成します。
9. 移行用に作成されたプロキシテーブルを削除する場合は、sa_migrate_drop_proxy_tables システムプロシージャを実行します。次に例を示します。

```
CALL sa_migrate_drop_proxy_tables( 'local_user1' );
```

結果

このプロシージャは、移行用に作成したプロキシテーブルをすべて削除し、移行プロセスを完了します。

関連情報

リモートサーバとリモートテーブルのマッピング [663 ページ]
外部ログイン [681 ページ]

1.7.9 SQL スクリプトファイル

SQL スクリプトファイルは、SQL 文を含むテキストファイルであり、同じ SQL 文を繰り返し実行する場合に便利です。

スクリプトファイルは手動で作成できますが、データベースユーティリティによって自動的に作成することもできます。たとえば、アンロードユーティリティ (dbunload) を使うと、データベースの再構築に必要な SQL 文で構成されたスクリプトファイルを作成できます。

SQL スクリプトファイルの作成

SQL スクリプトファイルの作成には好きなテキストエディタを使用できますが、Interactive SQL の使用をお奨めします。実行される SQL 文にはコメント行を含めることができます。

i 注記

Interactive SQL では、SQL スクリプトファイルをお気に入りから *SQL* 文ウィンドウ枠にロードすることができます。

このセクションの内容:

SQL スクリプトファイルをロードしないで実行 [657 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

Interactive SQL を使用して、SQL スクリプトファイルを *SQL 文*ウィンドウ枠にロードすることなく実行します。

Interactive SQL の READ 文を使用した SQL スクリプトファイルの実行 [658 ページ]
Interactive SQL の READ 文を使用して、*SQL 文*ウィンドウ枠にロードすることなく、SQL スクリプトファイルを実行します。

バッチモードでの SQL スクリプトファイルの実行 (コマンドラインの場合) [658 ページ]
Interactive SQL のコマンドライン引数として SQL スクリプトファイルを指定します。

ファイルから SQL 文ウィンドウ枠への SQL スクリプトのロード [659 ページ]
Interactive SQL を使用して、SQL スクリプトファイルを *SQL 文*ウィンドウ枠にロードし、そこから直接実行します。

ファイルへのデータベース出力の書き込み [660 ページ]
SQL 文の出力をファイルに保存します。

1.7.9.1 SQL スクリプトファイルをロードしないで実行

Interactive SQL を使用して、SQL スクリプトファイルを *SQL 文*ウィンドウ枠にロードすることなく実行します。

前提条件

Interactive SQL が `.sql` ファイルのデフォルトエディタとして設定されていることを確認します。

Interactive SQL で、**ツール** > **オプション** > **一般** をクリックしてから、*Interactive SQL を .SQL ファイルとプランファイルのデフォルトエディタにする*をクリックします。

必要な権限は、実行する文によって異なります。

コンテキスト

スクリプトの実行メニュー項目の機能は、READ 文と同じです。

手順

1. Interactive SQL で、**ファイル** > **スクリプトの実行** をクリックします。
2. ファイルを検索し、*[開く]* をクリックします。

結果

指定されたファイルの内容がすぐに実行されます。実行の進行状況を示す *[ステータス]* ウィンドウが表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.9.2 Interactive SQL の READ 文を使用した SQL スクリプトファイルの実行

Interactive SQL の READ 文を使用して、*SQL 文*ウィンドウ枠にロードすることなく、SQL スクリプトファイルを実行します。

前提条件

必要な権限は、実行する文によって異なります。

手順

*SQL 文*ウィンドウ枠で、次の例のような文を実行します。

```
READ 'C:¥¥LocalTemp¥¥filename.sql';
```

この文の `C:¥LocalTemp¥filename.sql` には、ファイルのパス、名前、拡張子を指定します。パスにスペースが含まれている場合にのみ、一重引用符 (例文参照) が必要です。一重引用符を使用する場合、円記号は 2 つ重ねることによりエスケープされ、'¥n' または '¥x' などのエスケープシーケンスの変換が回避されます。

結果

SQL スクリプトファイルが実行されます。

1.7.9.3 バッチモードでの SQL スクリプトファイルの実行 (コマンドラインの場合)

Interactive SQL のコマンドライン引数として SQL スクリプトファイルを指定します。

前提条件

必要な権限は、実行する文によって異なります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

dbisql ユーティリティを実行し、コマンドライン引数として SQL スクリプトファイルを指定します。

結果

SQL スクリプトファイルが実行されます。

例

次のコマンドは、SQL Anywhere サンプルデータベースに対して SQL スクリプトファイル myscript.sql を実行します。

```
dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql" myscript.sql
```

1.7.9.4 ファイルから SQL 文ウィンドウ枠への SQL スクリプトのロード

Interactive SQL を使用して、SQL スクリプトファイルを *SQL 文*ウィンドウ枠にロードし、そこから直接実行します。

前提条件

Interactive SQL が .sql ファイルのデフォルトエディタとして設定されていることを確認します。

Interactive SQL で、ツール > オプション > 一般 をクリックしてから、*Interactive SQL を .SQL ファイルとプランファイルのデフォルトエディタにする*をクリックします。

必要な権限は、実行する文によって異なります。

手順

1. ファイル > 開く をクリックします。
2. ファイルを検索し、*[開く]* をクリックします。

結果

文が *SQL 文* ウィンドウ枠に表示され、読み込み、編集、実行を行うことができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.9.5 ファイルへのデータベース出力の書き込み

SQL 文の出力をファイルに保存します。

前提条件

必要な権限は、実行する文によって異なります。

コンテキスト

Interactive SQL では、文の実行結果セットデータは、次の文が実行されると結果ウィンドウ枠の結果タブから消えます。

手順

たとえば、2 つの SELECT 文 statement1 と statement2 がある場合、次のようにしてそれぞれの実行結果を file1 と file2 に出力します。

```
statement1; OUTPUT TO file1;statement2; OUTPUT TO file2;
```

結果

各 SQL 文の出力が個別のファイルに保存されます。

例

次の文は、クエリの結果を C:¥LocalTemp ディレクトリ内の Employees.csv という名前のファイルに保存します。

```
SELECT * FROM Employees;
OUTPUT TO 'C:¥¥LocalTemp¥¥Employees.csv';
```

関連情報

UNLOAD 文を使用したデータのエクスポートに関するヒント [619 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.7.10 Adaptive Server Enterprise の互換性

BCP FORMAT 句を使用して、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できます。

Adaptive Server Enterprise で使用するために SQL Anywhere から BLOB データをエクスポートしている場合は、UNLOAD TABLE 文で BCP フォーマット句を使用します。

データを SQL Anywhere にインポートできるように、BCP の out コマンドを使用して Adaptive Server Enterprise からファイルをエクスポートする場合、データはテキストまたは ASCII フォーマットとし、カンマ区切りにしてください。テキスト/ASCII フォーマットでデータをエクスポートするには、BCP out コマンドの -c オプションを使用できます。-t オプションを使用すると、区切り文字 (デフォルトはタブ) を変更できます。区切り文字を変更しない場合は、SQL Anywhere データベースにデータをインポートするときに、LOAD TABLE 文に `DELIMITED BY '¥x09'` を指定してください。

1.8 リモートデータアクセス

リモートデータアクセスによって、他のデータソースのデータや、データベースサーバを実行しているコンピュータにあるファイルにアクセスすることができます。

リモートデータアクセスは、次の機能によりサポートされています。

オプション	説明
リモートサーバ	他のリレーショナルデータベースのデータにアクセスします。
ディレクトリアクセスサーバ	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	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します。

リモートデータアクセスを使用して、次のことを実行できます。

- INSERT および SELECT 文を使用してデータをあるロケーションから別のロケーションに移動するために SQL Anywhere を使用します。
- SAP Adaptive Server Enterprise、SAP HANA、Oracle Database、IBM DB2 などのリレーショナルデータベースのデータにアクセスします。
- Microsoft Excel スプレッドシート、Microsoft Access データベース、Microsoft Visual FoxPro、テキストファイルのデータにアクセスします。
- ODBC インタフェースをサポートするデータソースにアクセスします。
- ローカルデータとリモートデータ間でジョインを実行します。ただしパフォーマンスは、すべてのデータが単一の SQL Anywhere データベース内にある場合よりかなり低速になります。
- 別々の SQL Anywhere データベースのテーブル間でジョインを実行します。パフォーマンスの制限は、他のリモートデータソースの場合と同様です。
- 通常は SQL Anywhere の機能を持たないデータソースに対して、その機能を使用します。たとえば、Oracle データベースに格納されているデータに対して Java の機能を使用したり、スプレッドシートでサブクエリを実行したりできます。データを取り出してから操作することによって、リモートデータソースではサポートされていない機能を SQL Anywhere が補います。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- FORWARD TO 文を使用して、リモートサーバに直接アクセスします。
- 他のサーバへのリモートプロシージャコールを実行します。

次の外部データソースにもアクセスが可能です。

- SQL Anywhere
- SAP Adaptive Server Enterprise
- SAP HANA
- SAP IQ
- SAP Ultra Light
- SAP Advantage Database Server
- IBM DB2
- Microsoft Access
- Microsoft SQL Server
- Oracle MySQL
- Oracle Database
- その他の ODBC データソース

このセクションの内容:

リモートサーバとリモートテーブルのマッピング [663 ページ]

SQL Anywhere は、テーブル内の全データがアプリケーションの接続先データベースに格納されているかのように、クライアントアプリケーションにテーブルを提示します。リモートオブジェクトをローカルのプロキシテーブルにマッピングするには、リモートオブジェクトが置かれるリモートサーバを、あらかじめ定義しておきます。

ディレクトリアクセスサーバの代用としてのストアドプロシージャ [672 ページ]

以下の一連のストアドプロシージャを xp_read_file システムプロシージャおよび xp_write_file システムプロシージャと組み合わせれば、ディレクトリアクセスサーバと同じ機能を提供できますので、リモートサーバや外部ログインを作成する必要がありません。

ディレクトリアクセスサーバ [672 ページ]

ディレクトリアクセスサーバは、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へのアクセスを可能にするリモートサーバです。

外部ログイン [681 ページ]

外部ログインを使用して、リモートサーバとの通信を行い、ディレクトリアクセスサーバへのアクセスを許可します。

プロキシテーブル [684 ページ]

リモートデータベースがプロキシテーブルの候補としてエクスポートする任意のオブジェクト (テーブル、ビュー、マテリアライズドビューを含む) にアクセスするには、プロキシテーブルを使用します。

ネイティブ文とリモートサーバ [693 ページ]

FORWARD TO 文を使用して、1 つまたは複数の文をネイティブ構文でリモートサーバに送信します。

リモートプロシージャコール (RPC) [693 ページ]

リモートサーバへのプロシージャコールを発行できます。

トランザクションの管理とリモートデータ [698 ページ]

トランザクションを使って、複数の SQL 文をグループ化して 1 つの単位として処理するようにできます。(SQL 文の実行結果はすべてデータベースにコミットされるか、1 つもコミットされないかのいずれかになります。)

クエリで実行される内部オペレーション [699 ページ]

ローカルとリモートの両方の、すべてのクエリに対して複数の手順が実行されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

その他の内部オペレーション [700 ページ]
: クエリで実行された内部オペレーションのほかに、データベースサーバで実行される内部オペレーションがいくつかあります。

トラブルシューティング: リモートデータには使用できない機能 [702 ページ]
: いくつかの機能は、リモートデータではサポートしていません。

トラブルシューティング: 大文字と小文字の区別およびリモートデータアクセス [703 ページ]
: SQL Anywhere データベースの大文字と小文字の区別の設定は、アクセス先のリモートサーバの設定に合わせてください。

トラブルシューティング: リモートデータアクセスの接続テスト [703 ページ]
: 短い手順に従えば、リモートサーバへの接続を確認できます。

トラブルシューティング: クエリ上でブロックされるクエリ [703 ページ]
: クエリが実行する個々のタスクをサポートするのに十分なスレッドが使用可能でなければなりません。

トラブルシューティング:ODBC を使用したリモートデータアクセスの接続 [704 ページ]
: ODBC を介してリモートデータベースにアクセスする場合、リモートサーバへの接続には名前が付けられます。

リモートデータアクセスのサーバクラス [704 ページ]
: CREATE SERVER 文に指定するサーバクラスは、リモート接続の動作を決定します。

1.8.1 リモートサーバとリモートテーブルのマッピング

SQL Anywhere は、テーブル内の全データがアプリケーションの接続先データベースに格納されているかのように、クライアントアプリケーションにテーブルを提示します。リモートオブジェクトをローカルのプロキシテーブルにマッピングするには、リモートオブジェクトが置かれるリモートサーバを、あらかじめ定義しておきます。

リモートテーブルに関わるクエリが実行されると、内部では対象となるデータの格納場所を割り出し、外部にあるその場所にアクセスしてデータを取り出します。

リモートテーブルのデータにアクセスするには、次の設定を行う必要があります。

1. リモートオブジェクトが置かれるリモートサーバを定義します。これにはサーバのクラスとリモートサーバの場所が含まれます。CREATE SERVER 文を実行して、リモートサーバを定義します。
2. リモートサーバ上のデータベースへのアクセスに必要なクレデンシャルが、接続先のデータベースへのアクセスに必要なクレデンシャルと異なる場合は、リモートサーバのユーザログイン情報を定義する必要があります。CREATE EXTERNLOGIN 文を実行して、ユーザの外部ログインを作成します。
3. プロキシテーブルの定義を作成します。これによってリモートテーブルに対するローカルプロキシテーブルのマッピングを指定します。リモートテーブルが置かれているサーバ、リモートテーブルのデータベース名、所有者名、テーブル名、カラム名を指定します。CREATE EXISTING TABLE 文を実行して、プロキシテーブルを作成します。リモートサーバに新しいテーブルを作成するには、CREATE TABLE 文を実行します。

警告

Microsoft Access、Microsoft SQL Server、SAP Adaptive Server Enterprise などの一部のリモートサーバでは、複数の COMMIT や ROLLBACK を実行するとカーソルが保存されません。オートコミット機能が無効 (Interactive SQL のデフォルトの動作) になっている場合は、Interactive SQL を使用してプロキシテーブルのデータを表示および編集します。Oracle Database、IBM DB2、SQL Anywhere などのその他の RDBMS では、この制限はありません。

リモートサーバを定義するときは、サーバのクラスを選択します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

サーバクラスとは、リモートサーバとの対話に使用するアクセス方法を指定するものです。異なるタイプのリモートサーバには、異なるアクセス方法が必要です。データベースサーバは、サーバ機能に関する詳細情報をそのサーバクラスから得ます。データベースサーバはこれらの情報に基づいて、リモートサーバとのアクセスを調整します。

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。

このセクションの内容:

リモートサーバの作成 (SQL の場合) [664 ページ]
CREATE SERVER 文を使用して、リモートサーバ定義を設定します。

リモートサーバの作成 (SQL Central の場合) [666 ページ]
SQL Central を使用して、リモートサーバ定義定義を作成します。

リモートサーバの削除 (SQL の場合) [667 ページ]
DROP SERVER 文を使用してリモートサーバを削除します。

リモートサーバの削除 (SQL Central の場合) [668 ページ]
SQL Central でリモートサーバを削除します。

リモートサーバの変更 (SQL の場合) [669 ページ]
Interactive SQL でリモートサーバのプロパティを変更します。

リモートサーバの変更 (SQL Central の場合) [670 ページ]
SQL Central でリモートサーバのプロパティを変更します。

リモートサーバ上のテーブルのリスト (SQL の場合) [671 ページ]
システムプロシージャを使用すると、リモートサーバ上の全テーブルの制限されたリスト、または包括的なリストを表示します。

リモートサーバの機能 [672 ページ]
データベースサーバはリモートサーバの機能情報を使用して、リモートサーバに渡すことができる SQL 文の量を判断します。

関連情報

リモートデータアクセスのサーバクラス [704 ページ]

1.8.1.1 リモートサーバの作成 (SQL の場合)

CREATE SERVER 文を使用して、リモートサーバ定義を設定します。

前提条件

SERVER OPERATOR システム権限が必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

各リモートサーバにアクセスするには ODBC ドライバを使用します。データベースごとにリモートサーバの定義が必要です。

データソースの識別には接続文字列を使用します。UNIX プラットフォームでは、ODBC ドライバも接続文字列で参照される必要があります。

手順

CREATE SERVER 文を使用して、リモートサーバにリンクするリモートデータアクセスサーバを定義します。

たとえば、次の文はリモートサーバ RemoteASE を定義します。SQL Anywhere データベースサーバは、USING 句で指定された ODBC 接続文字列を使用して Adaptive Server Enterprise 16 データベースサーバに接続します。

```
CREATE SERVER RemoteASE
CLASS 'ASEODBC'
USING 'DRIVER=SAP ASE ODBC
Driver;Server=TestASE;Port=5000;Database=testdb;UID=username;PWD=password';
```

以下に、CREATE SERVER 文の各部の分析を説明します。

SERVER

この句を使用して、リモートサーバに名前を付けます。たとえば、RemoteASE はリモートサーバの名前です。

CLASS

この句を使用して、SQL Anywhere データベースサーバがリモートサーバと通信する方法を示します。この例では、ASEODBC はリモートサーバが Adaptive Server Enterprise (ASE) であり、ASE ODBC ドライバを使用して接続が確立されることを示します。

USING

この句は、リモートサーバの ODBC 接続文字列を指定します。この例では、Adaptive Server Enterprise 16 の ODBC ドライバ名が指定されています。

結果

CREATE SERVER 文は、ISYSSERVER システムテーブルにエントリを作成します。

例

次の文は、リモートサーバ RemoteSA を定義します。SQL Anywhere データベースサーバは、USING 句で指定された ODBC データソース名 (DSN) を使用して、SQL Anywhere データベースサーバに接続します。

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'SQL Anywhere 17 CustDB';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のステップ

必要な場合は、外部ログイン情報を作成します。

関連情報

外部ログインの作成 (SQL Central の場合) [682 ページ]
プロキシテーブルの作成 (SQL の場合) [688 ページ]

1.8.1.2 リモートサーバの作成 (SQL Central の場合)

SQL Central を使用して、リモートサーバ定義定義を作成します。

前提条件

MANAGE ANY USER と SERVER OPERATOR のシステム権限が必要です。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。
2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. ファイル > 新規 > リモートサーバ をクリックします。
4. 新しいリモートサーバの名前を指定してください。フィールドにリモートサーバの名前を入力し、次へをクリックします。
5. リモートサーバのタイプを選択して、*[次へ]*をクリックします。
6. 接続タイプを選択し、*[接続情報を指定してください。]* フィールドに次に示す接続情報を入力します。
   - ODBC ベースの接続では、データソース名を指定するか、ODBC Driver パラメータとその他の接続パラメータを指定します。
   - JDBC ベースの接続では、`computer-name:port-number` の形式で URL を指定します。

   データアクセスメソッド (JDBC か ODBC) は、データベースサーバがリモートデータベースへのアクセスに使用するためのものです。SQL Central がデータベースに接続する方法をこれで決めているわけではありません。

   JDBC ベースのリモートサーバアクセスは、現在のリリースではサポートされていません。
7. 次へをクリックします。
8. リモートサーバを読み込み専用にするかどうかを指定し、*[次へ]* をクリックします。
9. 現在のユーザの外部ログインを作成するをクリックし、必須フィールドの入力を完成させます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトでは、データベースサーバは、現在のユーザに代わってリモートサーバに接続する場合に、常にそのユーザのユーザ ID とパスワードを使用します。ただし、リモートサーバに現在のユーザと同じユーザ ID とパスワードで定義されたユーザがいない場合は、外部ログインを作成する必要があります。外部ログインは、現在のユーザ用に代替ログイン名とパスワードを割り当ててリモートサーバに接続できるようにします。

10. 接続テストをクリックしてリモートサーバ接続をテストします。
11. 完了をクリックします。

結果

リモートサーバは、指定された定義で作成されます。

次のステップ

必要な場合は、外部ログイン情報を作成します。

関連情報

外部ログインの作成 (SQL Central の場合) [682 ページ]
プロキシテーブルの作成 (SQL Central の場合) [687 ページ]

1.8.1.3 リモートサーバの削除 (SQL の場合)

DROP SERVER 文を使用してリモートサーバを削除します。

前提条件

SERVER OPERATOR システム権限が必要です。

コンテキスト

リモートサーバを削除する前に、そのリモートサーバに定義されているプロキシテーブルをすべて削除する必要があります。次のクエリを使用すると、リモートサーバ server-name に定義されているプロキシサーバを特定できます。

```
SELECT st.table_name, sp.remote_location, sp.existing_obj
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    FROM SYS.SYSPROXYTAB sp
    JOIN SYS.SYSSERVER ss ON ss.srvid = sp.srvid
    JOIN SYS.SYSTAB st ON sp.table_object_id = st.object_id
    WHERE ss.srvname = 'server-name';
```

手順

1. ホストデータベースに接続します。
2. DROP SERVER 文を実行します。

```
DROP SERVER server-name;
```

結果

リモートサーバが削除されます。

例

次の文は RemoteASE という名前のリモートサーバを削除します。

```
DROP SERVER RemoteASE;
```

関連情報

リモートサーバの削除 (SQL Central の場合) [668 ページ]

1.8.1.4 リモートサーバの削除 (SQL Central の場合)

SQL Central でリモートサーバを削除します。

前提条件

SERVER OPERATOR システム権限が必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

リモートサーバを削除する前に、そのリモートサーバに定義されているプロキシテーブルをすべて削除する必要があります。SQL Central は、リモートサーバに定義されているプロキシテーブルを自動的に特定して、それらを最初に削除します。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。
2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. リモートサーバを選択し、編集 > 削除 をクリックします。

結果

リモートサーバが削除されます。

関連情報

リモートサーバの削除 (SQL の場合) [667 ページ]

1.8.1.5 リモートサーバの変更 (SQL の場合)

Interactive SQL でリモートサーバのプロパティを変更します。

前提条件

SERVER OPERATOR システム権限が必要です。

コンテキスト

ALTER SERVER 文は、サーバの既知の機能の有効化または無効化にも使用できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. ホストデータベースに接続します。
2. ALTER SERVER 文を実行します。

結果

リモートサーバのプロパティが変更されます。

ただし、リモートサーバの設定の変更は、次回リモートサーバに接続する際に有効になります。

例

次の文は、RemoteASE という名前のサーバのサーバクラスを ASEODBC に変更します。

```
ALTER SERVER RemoteASE
CLASS 'ASEODBC';
```

1.8.1.6 リモートサーバの変更 (SQL Central の場合)

SQL Central でリモートサーバのプロパティを変更します。

前提条件

SERVER OPERATOR システム権限が必要です。

コンテキスト

リモートサーバの設定の変更は、次回リモートサーバに接続する際に有効になります。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。
2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. リモートサーバを選択し、ファイル > プロパティをクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

4. リモートサーバの設定を変更し、*OK* をクリックします。

結果

リモートサーバのプロパティが変更されます。

1.8.1.7 リモートサーバ上のテーブルのリスト (SQL の場合)

システムプロシージャを使用すると、リモートサーバ上の全テーブルの制限されたリスト、または包括的なリストを表示します。

手順

sp_remote_tables システムプロシージャを呼び出し、リモートサーバ上のテーブルのリストを返します。

`@table_name` または `@table_owner` を指定すると、テーブルのリストは一致するテーブルにのみ制限されます。

結果

すべてのテーブルのリスト、またはテーブルの制限されたリストが返されます。

例

GROUPO が所有する RemoteSA という名前のリモートサーバで、データベースのすべてのテーブルのリストを取得するには、次の文を実行します。

```
CALL sp_remote_tables('RemoteSA', null, 'GROUPO');
```

RemoteASE という名前の Adaptive Server Enterprise サーバで、運用データベースのすべてのテーブルのリストを取得するには、次の文を実行します。

```
CALL sp_remote_tables('RemoteASE', null, 'Fred', 'Production');
```

Excel という名前のリモートサーバから使用可能な、すべての Microsoft Excel ワークシートのリストを取得するには、次の文を実行します。

```
CALL sp_remote_tables('Excel');
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.1.8 リモートサーバの機能

データベースサーバはリモートサーバの機能情報を使用して、リモートサーバに渡すことができる SQL 文の量を判断します。

リモートサーバの機能を調べるには、sp_servercaps システムプロシージャを使用します。

また、システムビュー SYSCAPABILITY および SYSCAPABILITYNAME を問い合わせると、リモートサーバの機能情報を参照できます。これらのシステムビューは、SQL Anywhere が最初にリモートサーバに接続するまでは空の状態です。

sp_servercaps システムプロシージャを使用するときは、`server-name` には CREATE SERVER 文で使用した `server-name` と同じ名前を指定してください。

次のようにして、sp_servercaps ストアドプロシージャを実行します。

```
CALL sp_servercaps('server-name');
```

1.8.2 ディレクトリアクセスサーバの代用としてのストアドプロシージャ

以下の一連のストアドプロシージャを xp_read_file システムプロシージャおよび xp_write_file システムプロシージャと組み合わせれば、ディレクトリアクセスサーバと同じ機能を提供できますので、リモートサーバや外部ログインを作成する必要がありません。

ディレクトリの内容の一覧表示、ファイルの取り出し、ディレクトリの管理といった簡単なタスクであれば、ストアドプロシージャは高機能なディレクトリアクセスサーバの代用として十分使用できます。ストアドプロシージャは簡単に使用でき、特に設定も必要ありません。システム権限やセキュリティ機能によって制限をかけてください。

ディレクトリプロシージャ	ファイルプロシージャ
dbo.sp_list_directory	
dbo.sp_copy_directory	dbo.sp_copy_file
dbo.sp_move_directory	dbo.sp_move_file
dbo.sp_delete_directory	dbo.sp_delete_file

1.8.3 ディレクトリアクセスサーバ

ディレクトリアクセスサーバは、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へのアクセスを可能にするリモートサーバです。

ディレクトリアクセスサーバを作成すると、以下を制御できます。

- アクセスできるサブフォルダ数
- データベースのユーザによるファイルの変更や作成

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトでは、各ユーザの外部ログインを作成することで、ディレクトリアクセスサーバへのアクセス権を明示的に付与します。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、ディレクトリアクセスサーバに対してデフォルトの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。

代替方法

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もできます。

このセクションの内容:

ディレクトリアクセスサーバの作成 (SQL Central の場合) [673 ページ]
ディレクトリサーバと、ディレクトリサーバに必要なプロキシテーブルを作成します。ディレクトリアクセスサーバにより、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へのアクセスを提供します。

ディレクトリアクセスプロキシテーブルでのクエリ [674 ページ]
ディレクトリアクセスプロキシテーブルを問い合わせるタイミングを検討する際のヒントがいくつかあります。

ディレクトリアクセスサーバの作成 (SQL の場合) [676 ページ]
CREATE SERVER 文を使用して、ディレクトリアクセスサーバを作成します。

チュートリアル:動的なディレクトリアクセスサーバの作成 (SQL の場合) [677 ページ]
CREATE SERVER 文とともにディレクトリアクセスサーバのルートとサブディレクトリのレベルを表す変数を使用して、動的な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の削除 (SQL Central の場合) [679 ページ]
ディレクトリアクセスサーバと、それに関連付けられたプロキシテーブルを削除します。

ディレクトリアクセスサーバの削除 (SQL の場合) [680 ページ]
SQL 文を使用して、ディレクトリアクセスサーバを削除します。

1.8.3.1 ディレクトリアクセスサーバの作成 (SQL Central の場合)

ディレクトリサーバと、ディレクトリサーバに必要なプロキシテーブルを作成します。ディレクトリアクセスサーバにより、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へのアクセスを提供します。

前提条件

MANAGE ANY USER と SERVER OPERATOR 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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

手順

1. データベースに接続します。
2. 左ウィンドウ枠でディレクトリアクセスサーバを右クリックし、 新規 > ディレクトリアクセスサーバ をクリックします。
3. ウィザードの指示に従ってディレクトリアクセスサーバを作成し、そのサーバへのアクセスを制限する方法を指定します。

   デフォルトでは、ユーザがディレクトリアクセスサーバに接続するには、外部ログインを使用する必要があります。このオプションを選択すると、特定のユーザに対して外部ログインを作成するよう要求されます。

   各ユーザにディレクトリアクセスサーバへのアクセス権を付与する別の方法として、すべてのユーザがアクセスできるようにデフォルト外部ログインを作成するというオプションもあります。
4. ディレクトリアクセスサーバのプロキシテーブルを作成します。右側のウィンドウ枠で、プロキシテーブルタブをクリックし、 新規 > プロキシテーブル を右クリックします。
5. ウィザードの指示に従い、プロキシテーブルを作成します。ディレクトリアクセスサーバ 1 つにつき、プロキシテーブルが 1 つ必要です。

   デフォルトでは、プロキシテーブルの項目デリミタはセミコロン (;) です。

結果

ディレクトリアクセスサーバが、プロキシテーブルとともに作成および設定されます。

1.8.3.2 ディレクトリアクセスプロキシテーブルでのクエリ

ディレクトリアクセスプロキシテーブルを問い合わせるタイミングを検討する際のヒントがいくつかあります。

パフォーマンスを向上させるには、テーブルスキャンを発生させるクエリを使用するときに contents カラムを選択しないようにします。

可能であれば、ファイル名を使用してディレクトリアクセスプロキシテーブルの内容を取得します。ファイル名を述部として使用すると、指定したファイルのみがディレクトリアクセスサーバによって読み込まれるため、パフォーマンスは向上します。ファイル名が不明の場合は、最初にファイルのリストを取得するクエリを実行してから、リスト内の各ファイルのクエリを発行してその内容を取得します。

例

例 1

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

次に示すクエリの実行は遅い場合があります (ディレクトリ内のファイルの数とサイズによって異なります)。述部と一致するものを見つけるために、ディレクトリ内のすべてのファイルの内容をディレクトリアクセスサーバで読み込む必要があるためです。

```
SELECT contents FROM DirAccessProxyTable WHERE file_name LIKE 'something%';
```

例 2

次に示すクエリは、ディレクトリスキャンを発生させないで、単一のファイルの内容を返します。

```
SELECT contents FROM DirAccessProxyTable WHERE file_name = 'something';
```

例 3

次に示すクエリの実行も遅い場合があります (ディレクトリ内のファイルの数とサイズによって異なります)。論理和 (OR) があるために、テーブルスキャンをディレクトリアクセスサーバで実行する必要があるためです。

```
SELECT contents FROM DirAccessProxyTable WHERE file_name = 'something' OR size
= 10;
```

例 4

filename をリテラル定数としてクエリ内で指定する代わりに、ファイル名の値を変数に入れて、その変数をクエリで使用できます。

```
DECLARE @filename LONG VARCHAR;
SET @filename = 'something';
SELECT contents FROM DirAccessProxyTable WHERE file_name = @filename;
```

このセクションの内容:

デリミタの一貫性 [675 ページ]
ディレクトリアクセスプロキシテーブルを問い合わせるときは、パス名のデリミタの使用法が一貫している必要があります。

ディレクトリアクセスサーバのプロキシテーブル列 [676 ページ]
ディレクトリアクセスサーバのプロキシテーブルには、同じスキーマ定義が含まれます。

1.8.3.2.1 デリミタの一貫性

ディレクトリアクセスプロキシテーブルを問い合わせるときは、パス名のデリミタの使用法が一貫している必要があります。

プラットフォームのネイティブデリミタを使用することをお奨めします。Windows では ¥ を使用し、UNIX では / を使用します。Windows では / もサーバによってデリミタとして認識されますが、リモートデータアクセスでは、ファイル名は常に一貫したデリミタを使用して返されます。そのため、デリミタが一貫していないクエリではローは返されません。

例

次のクエリではローは返されません。

```
SELECT contents FROM DirAccessProxyTable WHERE filename = 'some/dir¥thing';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.3.2.2 ディレクトリアクセスサーバのプロキシテーブル列

ディレクトリアクセスサーバのプロキシテーブルには、同じスキーマ定義が含まれます。

以下の表に、ディレクトリアクセスサーバのプロキシテーブルにおけるカラムを示します。

カラム名およびデータ型	説明
permissions VARCHAR(10)	POSIX スタイルのパーミッション文字列 (drwxrwxrwx など)。
size BIGINT	ファイルのサイズ (バイト単位)。
access_date_time TIMESTAMP	ファイルが最後にアクセスされた日付と時刻 (たとえば、2010-02-08 11:00:24.000)。
modified_date_time TIMESTAMP	ファイルが最後に変更された日付と時刻 (たとえば、2009-07-28 10:50:11.000)。
create_date_time TIMESTAMP	ファイルが作成された日付と時刻 (たとえば、2008-12-18 10:32:26.000)。
owner VARCHAR(20)	ファイル作成者のユーザ ID (たとえば、Linux の root)。Windows の場合、この値は常に "0" です。
file_name VARCHAR(260)	相対パスを含むファイル名 (たとえば、bin¥perl.exe)。
contents LONG BINARY	このカラムが結果セットで明示的に参照される場合、ファイルの内容。

1.8.3.3 ディレクトリアクセスサーバの作成 (SQL の場合)

CREATE SERVER 文を使用して、ディレクトリアクセスサーバを作成します。

前提条件

SERVER OPERATOR および MANAGE ANY USER 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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. CREATE SERVER 文を使用して、リモートサーバを作成します。例:

```
CREATE SERVER my_dir_server
CLASS 'DIRECTORY'
USING 'ROOT=c:¥Program Files;SUBDIRS=3';
```

   デフォルトでは、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です。この方法でアクセスを制限する場合、CREATE EXTERNLOGIN 文を実行して、それぞれのデータベースユーザに対して外部ログインを作成する必要があります。例:

```
CREATE EXTERNLOGIN DBA TO my_dir_server;
```

   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、CREATE SERVER 文で ALLOW ALL USERS 句を指定して、ディレクトリアクセスサーバに対してデフォルトの外部ログインを作成します。
2. CREATE EXISTING TABLE 文を実行して、ディレクトリアクセスサーバのプロキシテーブルを作成します。例:

```
CREATE EXISTING TABLE my_program_files AT 'my_dir_server;;;.';
```

3. プロキシテーブルのローを表示します。

```
SELECT * FROM my_program_files ORDER BY file_name;
```

4. 省略可能です。sp_remote_tables 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上の `c:¥mydir` にあるすべてのサブディレクトリを表示します。

```
CALL sp_remote_tables( 'my_dir_server' );
```

結果

ディレクトリアクセスサーバは作成され、設定されます。

1.8.3.4 チュートリアル:動的なディレクトリアクセスサーバの作成 (SQL の場合)

CREATE SERVER 文とともにディレクトリアクセスサーバのルートとサブディレクトリのレベルを表す変数を使用して、動的なディレクトリアクセスサーバを作成します。

前提条件

SERVER OPERATOR システム権限が必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

あなたは DBA であり、あるときは server1 という名前のデータベースサーバが起動されるコンピュータ A のデータベース、またはあるときは server2 という名前のデータベースサーバが起動されるコンピュータ B のデータベースを利用しているとします。ここで、コンピュータ A のローカルドライブ `c:¥temp` を参照するディレクトリアクセスサーバと、コンピュータ B のネットワークサーバドライブ `d:¥temp` をセットアップしたいとします。さらに、プロキシテーブルをセットアップして、すべてのユーザがそこから自分のプライベートディレクトリのリストを取得できるようにしたいとします。次の手順に従って、CREATE SERVER 文の USING 句と CREATE EXISTING TABLE 文の AT 句の変数を使用することによって、単一のディレクトリアクセスサーバと単一のプロキシテーブルを作成すれば、そのような要求を満たすことができます。

手順

1. この例では、接続しているサーバの名前が `server1` であるとして、次のディレクトリが存在するとします。

   ```
   c:¥temp¥dba
   c:¥temp¥updater
   c:¥temp¥browser
   ```

   ディレクトリアクセスサーバのルートとサブディレクトリのレベルを表す変数を使用して、ディレクトリアクセスサーバを作成します。

   ```
   CREATE SERVER dir
   CLASS 'DIRECTORY'
   USING 'root={@directory};subdirs={@subdirs}';
   ```

2. ディレクトリアクセスサーバの使用を許可するそれぞれのユーザに明示的な外部ログインを作成します。

   ```
   CREATE EXTERNLOGIN "DBA" TO dir;
   CREATE EXTERNLOGIN "UPDATER" TO dir;
   CREATE EXTERNLOGIN "BROWSER" TO dir;
   ```

3. ディレクトリアクセスサーバとその関連するプロキシテーブルを動的に設定するための変数を作成します。

   ```
   CREATE VARIABLE @directory LONG VARCHAR;
   SET @directory = 'c:¥¥temp';
   CREATE VARIABLE @subdirs VARCHAR(10);
   SET @subdirs = '7';
   CREATE VARIABLE @curuser VARCHAR(128);
   SET @curuser = 'updater';
   CREATE VARIABLE @server VARCHAR(128);
   SET @server = 'dir';
   ```

4. ディレクトリアクセスサーバ `@server` の `@directory¥@curuser` を参照するプロキシテーブルを作成します。

   ```
   CREATE EXISTING TABLE dbo.userdir AT '{@server};;;{@curuser}';
   ```

5. 変数はもう必要ではないため、次の文を実行して削除します。

   ```
   DROP VARIABLE @server;
   DROP VARIABLE @curuser;
   DROP VARIABLE @subdirs;
   DROP VARIABLE @directory;
   ```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

6. ユーザが自分のユーザディレクトリの内容を表示するためのプロシージャを作成します。

```
CREATE OR REPLACE PROCEDURE dbo.listmydir()
SQL SECURITY INVOKER
BEGIN
    DECLARE @directory LONG VARCHAR;
    DECLARE @subdirs VARCHAR(10);
    DECLARE @server VARCHAR(128);
    DECLARE @curuser VARCHAR(128);
    -- for this example we always use the "dir" remote directory access server
    SET @server = 'dir';
    -- the root directory is based on the name of the server the user is
connected to
    SET @directory = if property('name') = 'server1' then 'c:¥¥temp'
                     else 'd:¥¥temp' endif;
    -- the subdir limit is based on the connected user
    SET @curuser = user_name();
    -- all users get a subdir limit of 7 except "browser" who gets a limit of 1
    SET @subdirs = convert( varchar(10), if @curuser = 'browser' then 1 else 7
endif);
    -- with all the variables set above, the proxy table dbo.userdir
    -- now points to @directory¥@curuser and has a subdir limit of @subdirs
    SELECT * FROM dbo.userdir;
    DROP REMOTE CONNECTION TO dir CLOSE CURRENT;
END;
```

この手順の最後のステップでは、リモート接続を終了して、ユーザが (sp_remote_tables システムプロシージャなどを使用して) ディレクトリアクセスサーバのリモートテーブルをリストできないようにします。

7. ストアドプロシージャの一般的な用途に必要な権限を設定します。

```
GRANT SELECT ON dbo.userdir TO PUBLIC;
GRANT EXECUTE ON dbo.listmydir TO PUBLIC;
```

8. データベースサーバから切断して、ユーザ UPDATER (パスワード 'update') またはユーザ BROWSER (パスワード 'browse') として再接続します。次のクエリを実行します。

```
CALL dbo.listmydir()
```

結果

動的なディレクトリアクセスサーバが作成され、設定されます。

1.8.3.5 ディレクトリアクセスサーバの削除 (SQL Central の場合)

ディレクトリアクセスサーバと、それに関連付けられたプロキシテーブルを削除します。

前提条件

SERVER OPERATOR システム権限が必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. データベースに接続します。
2. 左ウィンドウ枠で、ディレクトリアクセスサーバをクリックします。
3. ディレクトリアクセスサーバを選択し、 編集 > 削除 をクリックします。

結果

ディレクトリアクセスサーバと、関連付けられているプロキシテーブルが削除されます。

関連情報

プロキシテーブルの削除 (SQL Central の場合) [689 ページ]

1.8.3.6 ディレクトリアクセスサーバの削除 (SQL の場合)

SQL 文を使用して、ディレクトリアクセスサーバを削除します。

前提条件

SERVER OPERATOR システム権限が必要です。

ディレクトリアクセスサーバを削除する前に、そのディレクトリアクセスサーバに定義されているプロキシテーブルをすべて削除する必要があります。次のクエリを使用すると、ディレクトリアクセスサーバ server-name に定義されているプロキシサーバを特定できます。

```
SELECT st.table_name, sp.remote_location, sp.existing_obj
    FROM SYS.SYSPROXYTAB sp
    JOIN SYS.SYSSERVER ss ON ss.srvid = sp.srvid
    JOIN SYS.SYSTAB st ON sp.table_object_id = st.object_id
    WHERE ss.srvname = 'server-name';
```

手順

1. ホストデータベースに接続します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. ディレクトリアクセスサーバに関連付けられている各プロキシテーブルに対して、DROP TABLE 文を実行します。

```
DROP TABLE my_program_files;
```

3. そのディレクトリアクセスサーバに対して、DROP SERVER 文を実行します。

```
DROP SERVER my_dir_server;
```

結果

ディレクトリアクセスサーバが削除されます。

関連情報

プロキシテーブルの削除 (SQL Central の場合) [689 ページ]

1.8.4 外部ログイン

外部ログインを使用して、リモートサーバとの通信を行い、ディレクトリアクセスサーバへのアクセスを許可します。

リモートサーバ接続

リモートサーバでは、外部ログインによってデータベースユーザをリモートサーバのログイン認証情報にマッピングします。

デフォルトでは、リモートサーバにアクセスするために、それぞれのデータベースユーザにユーザ自身の外部ログインが明示的に割り当てられている必要があります。ただし、データベースユーザすべてが使用できるデフォルトログインにしてリモートサーバを作成することができます。

リモートサーバにデフォルトログインが指定されていても、個別のデータベースユーザに外部ログインを作成することができます。たとえば、リモートサーバで、すべてのデータベースユーザに読み込みアクセスを許可するデフォルトログインを設定し、DBA データベースユーザにリモートサーバへの読み込み/書き込みアクセスを許可する外部ログインを設定することも可能です。

まずデータベース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。ユーザに外部ログインがない場合、リモートサーバのデフォルトログイン認証を使用して接続しようとします。リモートサーバにデフォルトログイン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、現在のユーザ ID とパスワードを使用して接続が試みられます。

ディレクトリアクセスサーバ接続

ディレクトリアクセスサーバでは、外部ログインによってアクセスを制限します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトでは、ディレクトリアクセスサーバにアクセスするために、それぞれのデータベースユーザにユーザ自身の外部ログインが明示的に割り当てられている必要があります。ただし、データベースユーザすべてが使用できるデフォルト外部ログインを設定して、ディレクトリアクセスサーバを作成することができます。どのユーザがディレクトリアクセスサーバにアクセスするか問題にならない場合、またはデータベースユーザ全員がアクセスできるようにする場合、ディレクトリアクセスサーバにデフォルト外部ログインを指定します。

このセクションの内容:

外部ログインの作成 (SQL Central の場合) [682 ページ]
: ユーザの外部ログインを作成し、リモートサーバやディレクトリアクセスサーバとの通信に使用します。

外部ログインの削除 (SQL Central の場合) [683 ページ]
: 不要になったリモートサーバとディレクトリアクセスサーバのユーザから外部ログインを削除します。

関連情報

ディレクトリアクセスサーバ [672 ページ]
リモートサーバとリモートテーブルのマッピング [663 ページ]

1.8.4.1 外部ログインの作成 (SQL Central の場合)

ユーザの外部ログインを作成し、リモートサーバやディレクトリアクセスサーバとの通信に使用します。

前提条件

リモートサーバまたはディレクトリアクセスサーバがデータベースに存在する必要があります。

MANAGE ANY USER システム権限が必要です。

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. 左ウィンドウ枠で、リモートサーバをクリックします。
3. リモートサーバを選択し、右ウィンドウ枠で 外部ログインタブをクリックします。
4. ファイルメニューから、 新規 外部ログイン をクリックします。
5. ウィザードの指示に従います。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

外部ログインが作成されます。

関連情報

ディレクトリアクセスサーバ [672 ページ]
リモートサーバとリモートテーブルのマッピング [663 ページ]

1.8.4.2 外部ログインの削除 (SQL Central の場合)

不要になったリモートサーバとディレクトリアクセスサーバのユーザから外部ログインを削除します。

前提条件

MANAGE ANY USER システム権限が必要です。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。
2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. リモートサーバを選択し、右ウィンドウ枠で *[外部ログイン]* タブをクリックします。
4. 外部ログインを選択し、 編集 > 削除 をクリックします。
5. はいをクリックします。

結果

外部ログインが削除されます。

関連情報

ディレクトリアクセスサーバ [672 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

リモートサーバとリモートテーブルのマッピング [663 ページ]

1.8.5 プロキシテーブル

リモートデータベースがプロキシテーブルの候補としてエクスポートする任意のオブジェクト (テーブル、ビュー、マテリアライズドビューを含む) にアクセスするには、プロキシテーブルを使用します。

プロキシテーブルとリモートサーバ

リモートオブジェクトにマッピングしたローカルのプロキシテーブルを作成すると、ロケーションを意識することなくリモートデータにアクセスできるようになります。プロキシテーブルを作成するには、次のいずれかの文を使用します。

- リモートサーバにすでにテーブルが存在する場合は、CREATE EXISTING TABLE 文を使用します。この文は、リモートサーバの既存のテーブルのプロキシテーブルを定義します。
- リモートサーバにテーブルが存在しない場合は、CREATE TABLE 文を使用します。この文はリモートサーバに新しいテーブルを作成するとともに、そのテーブルのプロキシテーブルを定義します。

> **i 注記**
>
> セーブポイント内では、プロキシテーブルのデータを変更することはできません。
>
> プロキシテーブルでトリガを起動する場合は、プロキシテーブルの所有者の権限ではなく、トリガを起動するユーザの権限を使用します。

プロキシテーブルとディレクトリアクセスサーバ

ディレクトリアクセスサーバ 1 つにつき、プロキシテーブルは 1 つのみ必要です。

このセクションの内容:

プロキシテーブルのロケーション [685 ページ]
CREATE TABLE 文と CREATE EXISTING TABLE 文で AT 句を使用して、既存のオブジェクトのロケーションを定義します。

プロキシテーブルの作成 (SQL Central の場合) [687 ページ]
プロキシテーブルを作成して、リモートデータベースサーバのテーブルに、ローカルテーブルであるかのようにアクセスすることができます。また、ディレクトリアクセスサーバと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できます。

プロキシテーブルの作成 (SQL の場合) [688 ページ]
CREATE TABLE 文または CREATE EXISTING TABLE 文を使用して、プロキシテーブルを作成します。

プロキシテーブルの削除 (SQL Central の場合) [689 ページ]
SQL Central を使用して、リモートサーバに関連付けられているプロキシテーブルを削除します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

リモートテーブルのカラムのリスト [690 ページ]
プロキシテーブルを問い合わせる前に、リモートテーブルで使用できるカラムのリストを取得すると便利な場合があります。

リモートテーブル間のジョイン [691 ページ]
プロキシテーブル間およびリモートテーブル間のジョインを使用できます。

複数のローカルデータベースのテーブル間のジョイン [692 ページ]
データベースサーバでは、同時に複数のローカルデータベースを稼働させることができます。他のローカル SQL Anywhere データベースのテーブルをリモートテーブルとして定義することにより、データベース間のジョインを実行できます。

関連情報

トランザクション内のセーブポイント [762 ページ]

1.8.5.1 プロキシテーブルのロケーション

CREATE TABLE 文と CREATE EXISTING TABLE 文で AT 句を使用して、既存のオブジェクトのロケーションを定義します。

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

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。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。ロケーション文字列はまた、ロケーション文字列中のこれらの区切り文字をエスケープすることもできます。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。

AT 句の構文は次のようになります。

```
... AT 'server.database.owner.table-name'
```

server

現在のデータベースでサーバを識別する名前 (CREATE SERVER 文で指定されるサーバ名) です。このフィールドはすべてのリモートデータソースに必須です。

database

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベースフィールドの意味は、データソースによって異なります。このフィールドは使用しないで、空にしておく場合があります。ただし、その場合でもデリミタは必要です。

データソースが Adaptive Server Enterprise の場合は、`database` によってテーブルが存在するデータベースを指定します。たとえば、master または pubs2 などです。

データソースが SQL Anywhere である場合、このフィールドは適用されないため、空白としておきます。

データソースが Microsoft Excel、Lotus Notes、または Microsoft Access である場合は、テーブルが保存されているファイルの名前を入力します。ファイル名にピリオドが含まれる場合には、区切り文字としてセミコロンを使用してください。

owner

データベースが所有者の概念をサポートしている場合、このフィールドは所有者名を表します。このフィールドは、複数の所有者が同じ名前のテーブルを所有する場合にのみ必要となります。

table-name

このフィールドはテーブルの名前を指定します。Microsoft Excel スプレッドシートの場合、これはブックのシートの名前になります。`table-name` が入力されない場合、リモートテーブル名はローカルのプロキシテーブル名と同じであるとみなされます。

例

以下はロケーション文字列の使用例です。

- SQL Anywhere の場合:

```
'RemoteSA..GROUPO.Employees'
```

- Adaptive Server Enterprise:

```
'RemoteASE.pubs2.dbo.publishers'
```

- Microsoft Excel の場合:

```
'RemoteExcel;d:¥pcdb¥quarter3.xls;;sheet1$'
```

- Microsoft Access の場合:

```
'RemoteAccessDB;¥¥server1¥production¥inventory.mdb;;parts'
```

関連情報

プロキシテーブルの作成 (SQL の場合) [688 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.5.2 プロキシテーブルの作成 (SQL Central の場合)

プロキシテーブルを作成して、リモートデータベースサーバのテーブルに、ローカルテーブルであるかのようにアクセスすることができます。また、ディレクトリアクセスサーバと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できます。

前提条件

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。

コンテキスト

SQL Central は、システムテーブルのプロキシテーブルの作成をサポートしません。ただし、システムテーブルのプロキシテーブルは、CREATE EXISTING TABLE 文を使用して作成できます。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。
2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. 次のオプションのうちの 1 つを選択してください。

オプション	アクション
リモートサーバで使用されるプロキシテーブルを作成する	1. 左ウィンドウ枠で、リモートサーバをクリックします。 2. リモートサーバを選択し、右ウィンドウ枠でプロキシテーブルタブをクリックします。 3. ファイルメニューで、新規 > プロキシテーブル をクリックします。
ディレクトリアクセスサーバで使用されるプロキシテーブルを作成する	1. 左ウィンドウ枠で、ディレクトリアクセスサーバをクリックします。 2. ディレクトリアクセスサーバを選択し、右ウィンドウ枠でプロキシテーブルタブをクリックします。 3. ファイルメニューで、新規 > プロキシテーブル をクリックします。

4. ウィザードの指示に従います。

結果

プロキシテーブルが作成されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.5.3 プロキシテーブルの作成 (SQL の場合)

CREATE TABLE 文または CREATE EXISTING TABLE 文を使用して、プロキシテーブルを作成します。

前提条件

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。

コンテキスト

AT 句とともに CREATE TABLE 文を使用すると、リモートサーバに新しいテーブルを作成し、そのテーブルに対するプロキシテーブルをローカルサーバに作成します。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。その際、区切り文字としてピリオドまたはセミコロンを使用します。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。データは SQL Anywhere により、リモートサーバのネイティブ形式に自動的に変換されます。

CREATE TABLE 文を使用してローカルテーブルとリモートテーブルの両方を作成し、続いて DROP TABLE 文を使用してプロキシテーブルを削除すると、リモートテーブルも削除されます。ただし、CREATE EXISTING TABLE 文を使用して作成したプロキシテーブルを DROP TABLE 文で削除しますが、この場合、リモートテーブルは削除されません。

CREATE EXISTING TABLE 文は、リモートサーバ上にある既存のテーブルにマッピングするプロキシテーブルを作成します。データベースサーバは、リモートロケーションのオブジェクトからカラム属性とインデックス情報を導出します。

手順

1. ホストデータベースに接続します。
2. CREATE EXISTING TABLE 文を実行します。

結果

プロキシテーブルが作成されます。

例

次の文は、RemoteSA というサーバ上のリモートテーブル Employees にマッピングする p_Employees というプロキシテーブルを、ローカルサーバ上に作成します。

```
CREATE EXISTING TABLE p_Employees
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
AT 'RemoteSA..GROUPO.Employees';
```

次の文は、プロキシテーブル a1 を Microsoft Access ファイル `mydbfile.mdb` にマッピングします。AT 句では、セミコロン (;) をデリミタとして使用しています。Microsoft Access 用に定義されているサーバの名前は access です。

```
CREATE EXISTING TABLE a1
AT 'access;d:¥mydbfile.mdb;;a1';
```

次の文は、リモートサーバ RemoteSA に Employees というテーブルを作成し、リモートテーブルにマッピングする Members というプロキシテーブルを作成します。

```
CREATE TABLE Members
( membership_id INTEGER NOT NULL,
member_name CHAR( 30 ) NOT NULL,
office_held CHAR( 20 ) NULL )
AT 'RemoteSA..GROUPO.Employees';
```

関連情報

プロキシテーブルのロケーション [685 ページ]

1.8.5.4 プロキシテーブルの削除 (SQL Central の場合)

SQL Central を使用して、リモートサーバに関連付けられているプロキシテーブルを削除します。

前提条件

所有者であるか、または DROP ANY TABLE と DROP ANY OBJECT のいずれかのシステム権限を持っている必要があります。

コンテキスト

リモートサーバを削除できるようにする前に、そのリモートサーバに関連付けられているすべてのプロキシテーブルを削除する必要があります。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. 右ウィンドウ枠で、プロキシテーブルタブをクリックします。
4. プロキシテーブルを選択し、 編集 > 削除 をクリックします。
5. はいをクリックします。

結果

プロキシテーブルが削除されます。

次のステップ

リモートサーバに関連付けられているすべてのプロキシテーブルが削除されたら、そのリモートサーバを削除できます。

関連情報

リモートサーバの削除 (SQL Central の場合) [668 ページ]

1.8.5.5 リモートテーブルのカラムのリスト

プロキシテーブルを問い合わせる前に、リモートテーブルで使用できるカラムのリストを取得すると便利な場合があります。

sp_remote_columns システムプロシージャは、リモートテーブルのカラムのリストとそれらのデータ型に関する説明を生成します。sp_remote_columns システムプロシージャの構文は次のとおりです。

```
CALL sp_remote_columns( @server_name, @table_name [, @table_owner [,
@table_qualifier ] ] )
```

テーブル名、所有者、またはデータベース名を指定すると、カラムのリストはその指定に当てはまるものだけに限定されます。

たとえば、asetest という名前の Adaptive Server Enterprise サーバの production データベースにある sysobjects テーブルのカラムのリストを取得するには、次のように指定します。

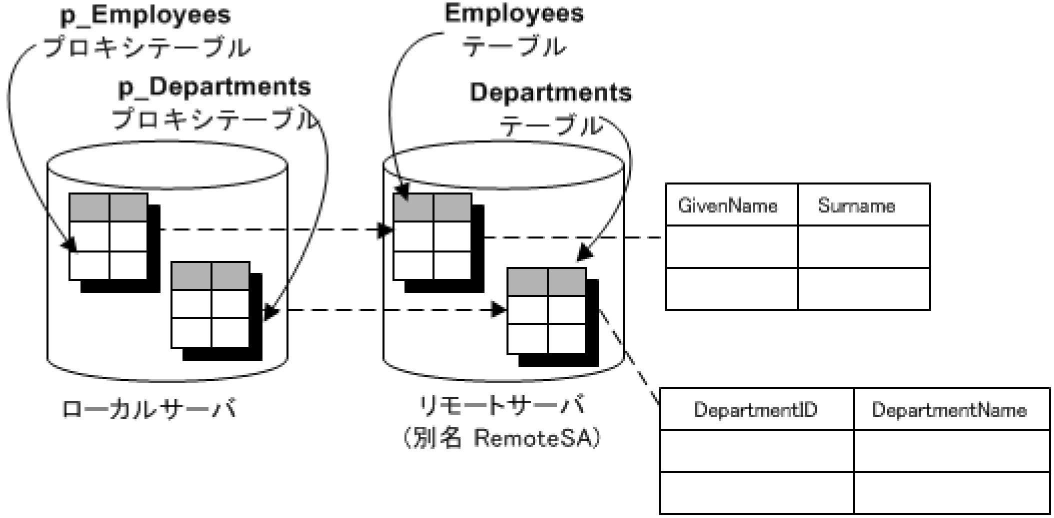
```
CALL sp_remote_columns('asetest, 'sysobjects', null, 'production');
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.5.6 リモートテーブル間のジョイン

プロキシテーブル間およびリモートテーブル間のジョインを使用できます。

次の図は、リモートサーバ RemoteSA にある SQL Anywhere サンプルデータベースのリモートテーブル Employees と Departments にマッピングされている、ローカルデータベースサーバのプロキシテーブルを示しています。



例

次の例では、2 つのリモートテーブル間のジョインを実行します。

1. `empty.db` という名前の新しいデータベースを作成します。
   このデータベースにはデータがありません。このデータベースは、リモートオブジェクトを定義して、SQL Anywhere サンプルデータベースにアクセスするためだけに使用します。
2. `empty.db` を実行するデータベースサーバを起動します。これを行うには、次のコマンドを実行します。

```
dbsrv17 empty
```

3. Interactive SQL から DBA ユーザとして `empty.db` に接続します。
4. 新しいデータベースで、RemoteSA という名前のリモートサーバを作成します。このサーバのサーバクラスは SAODBC で、接続文字列は SQL Anywhere 17 Demo ODBC データソースを参照します。

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'SQL Anywhere 17 Demo;PWD=sql';
```

5. この例では、ローカルデータベースと同じユーザ ID とパスワードをリモートデータベースで使用するので、外部ログインは必要ありません。
   場合によっては、リモートサーバのデータベースに接続するときに、ユーザ ID とパスワードを入力が必要です。新しいデータベースの場合は、リモートサーバへの外部ログインを作成します。例では簡素化するために、ローカルのログイン名とリモートのユーザ ID はどちらも DBA とします。

```
CREATE EXTERNLOGIN DBA
TO RemoteSA
REMOTE LOGIN DBA
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
IDENTIFIED BY sql;
```

6. p_Employees プロキシテーブルを定義します。

```
CREATE EXISTING TABLE p_Employees
AT 'RemoteSA..GROUPO.Employees';
```

7. p_Departments プロキシテーブルを定義します。

```
CREATE EXISTING TABLE p_Departments
AT 'RemoteSA..GROUPO.Departments';
```

8. SELECT 文にプロキシテーブルを使用して、ジョインを実行します。

```
SELECT GivenName, Surname, DepartmentName
FROM p_Employees JOIN p_Departments
ON p_Employees.DepartmentID = p_Departments.DepartmentID
ORDER BY Surname;
```

1.8.5.7 複数のローカルデータベースのテーブル間のジョイン

データベースサーバでは、同時に複数のローカルデータベースを稼働させることができます。他のローカル SQL Anywhere データベースのテーブルをリモートテーブルとして定義することにより、データベース間のジョインを実行できます。

例

データベース db1 を使用しているときに、データベース db2 内のテーブルのデータにアクセスするとします。この場合は、データベース db2 のテーブルを示すプロキシテーブル定義を設定します。たとえば、RemoteSA という名前の SQL Anywhere サーバ上で、db1、db2、db3 の 3 つのデータベースが使用可能であるとします。

1. ODBC を使用している場合、アクセスするデータベースのそれぞれに ODBC データソース名を作成します。
2. ジョインの実行元のデータベースに接続します。たとえば db1 に接続します。
3. アクセスするその他のローカルデータベースのそれぞれに、CREATE SERVER 文を実行します。これによって、SQL Anywhere サーバへのループバック接続が設定されます。

```
CREATE SERVER remote_db2
CLASS 'SAODBC'
USING 'RemoteSA_db2';
CREATE SERVER remote_db3
CLASS 'SAODBC'
USING 'RemoteSA_db3';
```

4. アクセスする他のデータベースにあるテーブルに CREATE EXISTING TABLE 文を実行して、プロキシテーブルの定義を作成します。

```
CREATE EXISTING TABLE Employees
AT 'remote_db2...Employees';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

CREATE SERVER 文の USING 句 [707 ページ]

1.8.6 ネイティブ文とリモートサーバ

FORWARD TO 文を使用して、1 つまたは複数の文をネイティブ構文でリモートサーバに送信します。

この文の使用方法は、次の 2 通りです。

- 1 つの文をリモートサーバに送信します。
- SQL Anywhere 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。

FORWARD TO 文を使用して、サーバが正しく設定されていることを検証できます。リモートサーバに文を送信して、SQL Anywhere がエラーメッセージを返さなければ、リモートサーバは正しく設定されています。

プロシージャ内またはバッチ内では FORWARD TO 文を使用できません。

指定したサーバに接続できない場合、メッセージがユーザに返されます。接続が確立された場合は、クライアントプログラムが認識できるフォームに結果が変換されます。

例

例 1

次の文は、バージョン文字列をセレクトすることによって、RemoteASE というサーバへの接続を検証します。

```
FORWARD TO RemoteASE {SELECT @@version};
```

例 2

次の文は、サーバ RemoteASE とのパススルーセッションを示します。

```
FORWARD TO RemoteASE;
    SELECT * FROM titles;
    SELECT * FROM authors;
FORWARD TO;
```

1.8.7 リモートプロシージャコール (RPC)

リモートサーバへのプロシージャコールを発行できます。

以下のリモートサーバがサポートされています。

- SQL Anywhere
- Adaptive Server Enterprise
- Oracle Database
- IBM DB2

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

複数の結果セットのフェッチなどの、リモートプロシージャからの結果セットのフェッチも可能です。また、リモートファンクションを使用してリモートプロシージャとファンクションから戻り値をフェッチできます。リモートプロシージャは、SELECT 文の FROM 句で使用できます。

リモートプロシージャのデータ型

リモートプロシージャコールのパラメータおよび RETURNS 値には、次のデータ型を使用できます。

- [UNSIGNED] SMALLINT
- [UNSIGNED] INTEGER
- [UNSIGNED] BIGINT
- [UNSIGNED] TINYINT
- TIME
- DATE
- TIMESTAMP
- REAL
- DOUBLE
- CHAR
- BIT
- データ型 LONG VARCHAR、LONG NVARCHAR、LONG BINARY は、IN パラメータには指定できますが、OUT パラメータ、INOUT パラメータ、RETURNS 値には指定できません。
- データ型 NUMERIC と DECIMAL は、IN パラメータには指定できますが、OUT パラメータ、INOUT パラメータ、RETURNS 値には指定できません。

このセクションの内容:

リモートプロシージャの作成 (SQL の場合) [695 ページ]
リモートプロシージャおよびファンクションを作成します。

リモートプロシージャの作成 (SQL Central の場合) [696 ページ]
リモートプロシージャを作成します。

リモートプロシージャの削除 (SQL の場合) [697 ページ]
SQL 文を使用してリモートプロシージャおよびリモートファンクションを削除します。

リモートプロシージャの削除 (SQL Central の場合) [697 ページ]
SQL Central でリモートプロシージャおよびファンクションを削除します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.7.1 リモートプロシージャの作成 (SQL の場合)

リモートプロシージャおよびファンクションを作成します。

前提条件

ユーザ本人が所有するプロシージャおよびファンクションを作成するには、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およびファンクションを作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT の権限が必要です。外部プロシージャとファンクションを作成するには、CREATE EXTERNAL REFERENCE システム権限も必要です。

コンテキスト

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。

手順

1. ホストデータベースに接続します。
2. プロシージャまたはファンクションを定義する文を実行します。

   次に例を示します。

   ```
   CREATE PROCEDURE RemoteProc()
   AT 'bostonase.master.dbo.sp_proc';
   ```

   ```
   CREATE FUNCTION RemoteFunc()
   RETURNS INTEGER
   AT 'bostonasa..dbo.sp_func';
   ```

   構文はローカルプロシージャの定義と似ています。ロケーション文字列は、プロシージャのロケーションを定義します。

結果

リモートプロシージャまたはファンクションが作成されます。

例

リモートプロシージャを呼び出すときにパラメータを指定する例を次に示します。

```
CREATE PROCEDURE RemoteUser ( IN username CHAR( 30 ) )
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
AT 'bostonase.master.dbo.sp_helpuser';
CALL RemoteUser( 'joe' );
```

1.8.7.2 リモートプロシージャの作成 (SQL Central の場合)

リモートプロシージャを作成します。

前提条件

ユーザ本人が所有するプロシージャおよびファンクションを作成するには、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およびファンクションを作成するには、CREATE ANY PROCEDURE または CREATE ANY OBJECT の権限が必要です。外部プロシージャとファンクションを作成するには、CREATE EXTERNAL REFERENCE システム権限も必要です。

コンテキスト

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。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。
2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. リモートサーバを選択し、右ウィンドウ枠で *[リモートプロシージャ]* タブをクリックします。
4. ファイルメニューで、新規 > リモートプロシージャ をクリックします。
5. リモートプロシージャ作成ウィザードの指示に従います。

結果

リモートプロシージャが作成されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.7.3 リモートプロシージャの削除 (SQL の場合)

SQL 文を使用してリモートプロシージャおよびリモートファンクションを削除します。

前提条件

プロシージャまたはファンクションの所有者であるか、または DROP ANY PROCEDURE か DROP ANY OBJECT のいずれかのシステム権限を持っている必要があります。

手順

プロシージャまたはファンクションを削除する文を実行します。

```
DROP PROCEDURE RemoteProc;
```

```
DROP FUNCTION RemoteFunc;
```

結果

リモートプロシージャまたはファンクションが削除されます。

1.8.7.4 リモートプロシージャの削除 (SQL Central の場合)

SQL Central でリモートプロシージャおよびファンクションを削除します。

前提条件

プロシージャまたはファンクションの所有者であるか、または DROP ANY PROCEDURE か DROP ANY OBJECT のいずれかのシステム権限を持っている必要があります。

手順

1. SQL Anywhere17 プラグインを使用して、ホストデータベースに接続します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. 左ウィンドウ枠で、リモートサーバをダブルクリックします。
3. リモートサーバを選択し、右ウィンドウ枠でリモートプロシージャタブをクリックします。
4. リモートプロシージャまたはファンクションを選択し、編集 > 削除をクリックします。
5. はいをクリックします。

結果

リモートプロシージャが削除されます。

1.8.8 トランザクションの管理とリモートデータ

トランザクションを使って、複数の SQL 文をグループ化して 1 つの単位として処理するようにできます。(SQL 文の実行結果はすべてデータベースにコミットされるか、1 つもコミットされないかのいずれかになります。)

リモートテーブルでのトランザクション管理は、SQL Anywhere のローカルテーブルでのトランザクション管理とほとんど同じですが、多少異なる点があります。

このセクションの内容:

リモートトランザクションの管理と制限 [698 ページ]
リモートサーバに関連するトランザクションを管理するには、マルチフェーズコミットプロトコルを使用します。

関連情報

トランザクションと独立性レベル [757 ページ]

1.8.8.1 リモートトランザクションの管理と制限

リモートサーバに関連するトランザクションを管理するには、マルチフェーズコミットプロトコルを使用します。

ただし、1 つのトランザクションで 2 つ以上のリモートサーバが呼び出されるときには、分散した作業単位が未定の状態で残る可能性があり、リカバリ処理は提供されません。

ユーザのトランザクションを管理する通常の論理は、次のようになっています。

1. SQL Anywhere は、BEGIN TRANSACTION 通知でリモートサーバの作業を開始します。
2. トランザクションのコミットの準備が整うと、SQL Anywhere は、トランザクションの一部である各リモートサーバに PREPARE TRANSACTION 通知を送信します。これによって、確実にリモートサーバがトランザクションをコミットできる状態にします。
3. PREPARE TRANSACTION 要求が失敗すると、すべてのリモートサーバは現在のトランザクションをロールバックするよう指示されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

PREPARE TRANSACTION 要求がすべて成功すると、サーバはトランザクションに関わるリモートサーバのそれぞれに、COMMIT TRANSACTION 要求を送信します。

BEGIN TRANSACTION によって開始すればどのようば文でも、トランザクションを開始できます。BEGIN TRANSACTION を明示しない場合は、SQL 文はリモートサーバに送信されて、1 つのリモートの作業単位として実行されます。

トランザクション管理の制限

トランザクション管理には、次のような制限があります。

- セーブポイントはリモートサーバに伝達されません。
- ネストされた BEGIN TRANSACTION 文と COMMIT TRANSACTION 文がリモートサーバに関わるトランザクションに含まれている場合は、一番外側の組の文だけが処理されます。BEGIN TRANSACTION 文と COMMIT TRANSACTION 文を含む一番内側の組は、リモートサーバに転送されません。

1.8.9 クエリで実行される内部オペレーション

ローカルとリモートの両方の、すべてのクエリに対して複数の手順が実行されます。

クエリの解析

文は、クライアントから受信されると、データベースサーバによって解析されます。有効な SQL 文でないと、データベースサーバでエラーが発生します。

クエリの正規化

クエリ内で参照されているオブジェクトが検証され、いくつかのデータ型の互換性が検査されます。

次のクエリを例にとります。

```
SELECT *
FROM t1
WHERE c1 = 10;
```

クエリの正規化の段階で、カラム c1 を持つテーブル t1 がシステムテーブルに存在することを確認します。また、カラム c1 のデータ型が値 10 と合っているかを確認します。たとえば、このカラムのデータ型が TIMESTAMP であった場合、この文は拒否されます。

クエリの前処理

クエリの前処理では、クエリの最適化の準備をします。ここで SQL 文の表現が変更されることもあるため、SQL Anywhere が生成してリモートサーバに引き渡す実際の SQL 文は、セマンティック上は同じであっても構文が元の文と異なる場合があります。

前処理は、ビューによって参照されるテーブルでクエリが操作できるよう、ビューの拡張を行います。処理を効率化するため、式が並べ替えられ、サブクエリが変更される場合があります。たとえば、いくつかのサブクエリがジョインに変換される場合があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.10 その他の内部オペレーション

クエリで実行された内部オペレーションのほかに、データベースサーバで実行される内部オペレーションがいくつかあります。

このセクションの内容:

サーバの機能 [700 ページ]
各リモートサーバに一連の機能が定義されています。

文の完全なパススルー [700 ページ]
効率性を考慮して、SQL Anywhere では、文に含まれるできるだけ多くの要素をリモートサーバに渡します。

文の部分的なパススルー [701 ページ]
ある文に複数のサーバへの参照が含まれている場合、またはリモートサーバでサポートされていない SQL 機能が文で使用されている場合、クエリは複数の単純な部分に分解されます。

1.8.10.1 サーバの機能

各リモートサーバに一連の機能が定義されています。

これらの機能は、ISYSCAPABILITY システムテーブルに格納され、リモートサーバへの最初の接続の間に初期化されます。

以降の手順は、SQL 文の型と、作業に関わるリモートサーバの機能によって異なります。

包括的なサーバクラスである ODBC は、ODBC ドライバから返される情報のみに厳密に基づいてリモートサーバの機能を判別します。DB2ODBC などのその他のサーバクラスには、リモートサーバタイプの機能についてより詳細な情報があり、その情報を使用して、ドライバから返される情報を補います。

ISYSCAPABILITY にサーバが追加されると、以後、そのリモートサーバの機能情報はそのシステムテーブルから取り出されるようになります。

リモートサーバは特定の SQL 文の全機能をサポートしているとは限らないため、データベースサーバでは、クエリをリモートサーバに送信できるようになるまで、文を単純なコンポーネントに分割する必要があります。リモートサーバに渡されない SQL 機能は、データベースサーバ自身によって評価する必要があります。

たとえば、あるクエリに ORDER BY 文があるとします。リモートサーバが ORDER BY を実行できない場合、ORDER BY を除いて、文がリモートサーバに送信されます。返された結果に ORDER BY が実行された後、結果がユーザに返されます。したがって、ユーザはサポートされる全種類の SQL を使用できます。

1.8.10.2 文の完全なパススルー

効率性を考慮して、SQL Anywhere では、文に含まれるできるだけ多くの要素をリモートサーバに渡します。

多くの場合、SQL Anywhere に渡された文がそのままの完全な形でリモートサーバに渡されます。

SQL Anywhere は、次のような場合に完全な文を渡します。

- 文内のすべてのテーブルが同じリモートサーバに存在しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- リモートサーバが文内のすべての構文を処理できます。

まれに、リモートサーバが作業を行うよりも、SQL Anywhere がいくつかの作業を行った方が効率が良い場合があります。たとえば、SQL Anywhere のソートアルゴリズムの方が優れていることがあります。この場合は、ALTER SERVER 文を使用して、リモートサーバの機能の変更を検討します。

1.8.10.3 文の部分的なパススルー

ある文に複数のサーバへの参照が含まれている場合、またはリモートサーバでサポートされていない SQL 機能が文で使用されている場合、クエリは複数の単純な部分に分解されます。

SELECT

引き渡すことのできない部分が取り除かれながら、SELECT 文は分解されていきます。取り除かれた部分の処理は SQL Anywhere によって実行されます。たとえば、次の文内にある ATAN2 関数をリモートサーバが処理できないとします。

```
SELECT a,b,c
WHERE ATAN2( b, 10 ) > 3
AND c = 10;
```

リモートサーバに送信される文は、次のように変換されます。

```
SELECT a,b,c WHERE c = 10;
```

次に、SQL Anywhere がローカルで `WHERE ATAN2( b, 10 ) > 3` を中間結果セットに適用します。

ジョイン

2 つのテーブルがジョインされると、1 つは外部テーブルとなります。外部テーブルは、テーブルに適用する WHERE 条件に基づいてスキャンされます。検出された、条件に合うどのローに対しても、内部テーブルと呼ばれるもう 1 つのテーブルがスキャンされ、ジョイン条件に一致するローが検出されます。

リモートテーブルが参照されるときにも同じアルゴリズムが使用されます。通常は、ローカルテーブルよりリモートテーブルを検索する方がコストがかかるので (ネットワーク I/O のため)、できるかぎりリモートテーブルをジョインの一番外側のテーブルにします。

UPDATE と DELETE

条件に合うローが検出されたとき、SQL Anywhere が UPDATE 文または DELETE 文全体をリモートサーバに渡すことができない場合は、元の WHERE 句のできるだけ多くの部分を持つ SELECT 文に文を変更して、WHERE CURRENT OF `cursor-name` を指定する位置付け UPDATE 文または DELETE 文を後に続ける必要があります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

たとえば、リモートサーバが関数 ATAN2 をサポートしていないとします。

```
UPDATE t1
SET a = atan2( b, 10 )
WHERE b > 5;
```

この文は次のように変換されます。

```
SELECT a,b
FROM t1
WHERE  b > 5;
```

ローが検出されるたびに、SQL Anywhere は a の新しい値を計算して、次の文を実行します。

```
UPDATE t1
SET a = 'new value'
WHERE CURRENT OF CURSOR;
```

新しい値と等しい値がすでに a にある場合、位置付け UPDATE は不要であり、リモートに送信されません。

テーブルスキャンを必要とする UPDATE 文または DELETE 文を処理するには、位置付け UPDATE または DELETE (WHERE CURRENT OF `cursor-name`) を実行する機能をリモートのデータソースがサポートしていなければなりません。データソースによってはこの機能をサポートしていません。

> **i 注記**
>
> テンポラリテーブルは更新できない中間テンポラリテーブルが必要な場合は、UPDATE または DELETE を実行できません。これは ORDER BY を持つクエリと、サブクエリを持つ一部のクエリで発生します。

1.8.11 トラブルシューティング: リモートデータには使用できない機能

いくつかの機能は、リモートデータではサポートしていません。

- リモートテーブルでの ALTER TABLE 文
- プロキシテーブルに定義されたトリガ
- SQL Remote
- リモートテーブルを参照する外部キー
- READTEXT 関数、WRITETEXT 関数、TEXTPTR 関数
- 位置付け UPDATE 文と DELETE 文
- 中間テンポラリテーブルを必要とする UPDATE 文と DELETE 文
- リモートデータに対して開かれたカーソルの後方スクロールフェッチ文は NEXT または RELATIVE 1 でなければなりません。
- プロキシテーブルを参照する式を含む関数の呼び出し
- リモートテーブルのカラムにリモートサーバに関するキーワードがある場合、そのカラムのデータにはアクセスできない。CREATE EXISTING TABLE 文を実行して定義をインポートすることはできますが、そのカラムを選択することはできません。
- 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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.12 トラブルシューティング: 大文字と小文字の区別およびリモートデータアクセス

SQL Anywhere データベースの大文字と小文字の区別の設定は、アクセス先のリモートサーバの設定に合わせてください。

SQL Anywhere データベースは、デフォルトでは大文字と小文字を区別せずに作成されます。この設定では、大文字と小文字を区別するデータベースから選択を行ったときに、予期しない結果が発生することがあります。ORDER BY または文字列比較がリモートサーバに送信されるか、それともローカルの SQL Anywhere によって評価されるかによって、発生する結果が異なります。

1.8.13 トラブルシューティング: リモートデータアクセスの接続テスト

短い手順に従えば、リモートサーバへの接続を確認できます。

- Interactive SQL などのクライアントツールを使用してリモートサーバに接続できることを確認してから、SQL Anywhere を設定します。
- リモートサーバに対して簡単なパススルー文を実行して、接続とリモートログインの設定を確認します。次に例を示します。

```
FORWARD TO RemoteSA {SELECT @@version};
```

- リモートサーバとの対話をトレースするために、リモートトレーシングを有効にします。次に例を示します。

```
SET OPTION cis_option = 7;
```

リモートトレーシングを有効にすると、データベースサーバのメッセージウィンドウにトレーシング情報が表示されます。この出力をファイルに記録するには、データベースサーバの起動時に -o サーバオプションを指定します。

1.8.14 トラブルシューティング: クエリ上でブロックされるクエリ

クエリが実行する個々のタスクをサポートするのに十分なスレッドが使用可能でなければなりません。

必要なタスクの数を提供できないと、クエリはそのクエリ上でブロックされます。

関連情報

トランザクションのブロックとデッドロック [779 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.15 トラブルシューティング:ODBC を使用したリモートデータアクセスの接続

ODBC を介してリモートデータベースにアクセスする場合、リモートサーバへの接続には名前が付けられます。

DROP REMOTE CONNECTION 文を使用すると、リモート要求をキャンセルできます。

接続の名前は、ASACIS_`unique-database-identifier_conn-name` のようになります。`conn-name` は、ローカル接続の接続 ID です。接続 ID は、sa_conn_info ストアドプロシージャから取得できます。

1.8.16 リモートデータアクセスのサーバクラス

CREATE SERVER 文に指定するサーバクラスは、リモート接続の動作を決定します。

サーバクラスは、SQL Anywhere に詳細なサーバ機能情報を提供します。SQL Anywhere は、サーバの機能に合わせて SQL 文をフォーマットします。

サーバクラスはすべて ODBC ベースです。各サーバクラスには各種のユニークな特性があります。リモートデータアクセス用にサーバを設定するには、この特性を知っておく必要があります。サーバクラスのカテゴリ全般についての情報とともに、個々のサーバクラスに固有の情報を参照する必要があります。

サーバのクラスは次のとおりです。

- ADSODBC
- ASEODBC
- DB2ODBC
- HANAODBC
- IQODBC
- MIRROR
- MSACCESSODBC
- MSSODBC
- MYSQLODBC
- ODBC
- ORAODBC
- SAODBC
- ULODBC

> **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。

このセクションの内容:

ODBC 外部サーバ定義 [706 ページ]
ODBC ベースのリモートサーバを定義する最も一般的な方法は、ODBC データソースを基にすることです。これを実行するために、ODBC データソースアドミニストレータを使用して、データソースを作成することができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

CREATE SERVER 文の USING 句 [707 ページ]
アクセスするリモートの SQL Anywhere データベースごとに、個別の CREATE SERVER 文を発行してください。

サーバクラス SAODBC [707 ページ]
サーバクラス SAODBC のリモートサーバは、SQL Anywhere データベースサーバです。

サーバクラス MIRROR [708 ページ]
サーバクラス MIRROR のリモートサーバは、SQL Anywhere データベースサーバです。

サーバクラス ULODBC [708 ページ]
サーバクラスが ULODBC のリモートサーバは Ultra Light データベースサーバです。

サーバクラス ADSODBC [709 ページ]
CREATE TABLE 文を実行すると、SQL Anywhere は、データ型を SAP Advantage Database Server の対応するデータ型に自動的に変換します。次の表に、SQL Anywhere から SAP Advantage Database Server へのデータ型変換を示します。

サーバクラス ASEODBC [710 ページ]
サーバクラスが ASEODBC であるリモートサーバは、Adaptive Server Enterprise (バージョン 10 以降) データベースサーバです。

サーバクラス DB2ODBC [713 ページ]
サーバクラスが DB2ODBC のリモートサーバは IBM DB2 データベースサーバです。

サーバクラス HANAODBC [714 ページ]
サーバクラスが HANAODBC のリモートサーバは SAP HANA データベースサーバです。

サーバクラス IQODBC [716 ページ]
サーバクラスが IQODBC のリモートサーバは、SAP IQ データベースサーバです。

サーバクラス MSACCESSODBC [716 ページ]
Microsoft Access データベースは .mdb ファイルに格納されます。ODBC マネージャを使用して、ODBC データソースを作成し、これらのファイルの 1 つにマッピングします。

サーバクラス MSSODBC [718 ページ]
サーバクラス MSSODBC を使用し、そのいずれかの ODBC ドライバを介して、Microsoft SQL Server にアクセスします。

サーバクラス MYSQLODBC [720 ページ]
CREATE TABLE 文を実行すると、SQL Anywhere は、データ型を Oracle MySQL の対応するデータ型に自動的に変換します。

サーバクラス ODBC [721 ページ]
独自のサーバクラスを持たない ODBC データソースでは、ODBC サーバクラスを使用します。

サーバクラス ORAODBC [725 ページ]
サーバクラスが ORAODBC のリモートサーバは、Oracle Database バージョン 8.0 以降です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.16.1 ODBC 外部サーバ定義

ODBC ベースのリモートサーバを定義する最も一般的な方法は、ODBC データソースを基にすることです。これを実行するために、ODBC データソースアドミニストレータを使用して、データソースを作成することができます。

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。

データソースを定義すると、CREATE SERVER 文の USING 句は ODBC データソース名 (DSN) を参照します。

たとえば、データソース名も mydb2 である mydb2 という名前の IBM DB2 サーバを定義するには、次の文を使用します。

```
CREATE SERVER mydb2
CLASS 'DB2ODBC'
USING 'mydb2';
```

使用するドライバはデータベースサーバのビット設定と一致する必要があります。

Windows では、ビット設定がデータベースサーバと一致するシステムデータソース名 (System DSN) を定義する必要もあります。たとえば、32 ビットのシステム DSN を作成するには、32 ビットの ODBC データソースアドミニストレータを使用します。ユーザ DSN にはビット設定はありません。

データソースの代わりに接続文字列を使用

データソース名を使用しない代わりに、CREATE SERVER 文の USING 句に接続文字列を指定します。これを実行するには、使用している ODBC ドライバの接続パラメータが必要です。たとえば、次は SQL Anywhere データベースサーバへの接続の例です。

```
CREATE SERVER TestSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;HOST=myhost;Server=TestSA;DBN=sample';
```

この例で定義される接続先の TestSA という名前のデータベースサーバは、myhost というコンピュータで実行され、データベースが sample という名前で、使用するプロトコルが TCP/IP です。

関連情報

サーバクラス SAODBC [707 ページ]
サーバクラス ULODBC [708 ページ]
サーバクラス ADSODBC [709 ページ]
サーバクラス ASEODBC [710 ページ]
サーバクラス DB2ODBC [713 ページ]
サーバクラス HANAODBC [714 ページ]
サーバクラス IQODBC [716 ページ]
サーバクラス MSACCESSODBC [716 ページ]
サーバクラス MSSODBC [718 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

サーバクラス MYSQLODBC [720 ページ]
サーバクラス ODBC [721 ページ]
サーバクラス ORAODBC [725 ページ]

1.8.16.2 CREATE SERVER 文の USING 句

アクセスするリモートの SQL Anywhere データベースごとに、個別の CREATE SERVER 文を発行してください。

たとえば、TestSA という名前の SQL Anywhere サーバが Banana というコンピュータで稼働していて、3 つのデータベース (db1、db2、db3) を所有する場合は、次のようなリモートサーバを設定します。

```
CREATE SERVER TestSAdb1
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;HOST=Banana;Server=TestSA;DBN=db1';
CREATE SERVER TestSAdb2
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;HOST=Banana;Server=TestSA;DBN=db2';
CREATE SERVER TestSAdb3
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;HOST=Banana;Server=TestSA;DBN=db3';
```

データベース名を指定しない場合、リモート接続ではリモートの SQL Anywhere サーバのデフォルトデータベースが使用されます。

1.8.16.3 サーバクラス SAODBC

サーバクラス SAODBC のリモートサーバは、SQL Anywhere データベースサーバです。

SQL Anywhere データソースの設定には、特別な要件はありません。

複数のデータベースをサポートする SQL Anywhere データベースサーバにアクセスするには、各データベースへの接続を定義する ODBC データソース名を作成します。これらの ODBC データソース名のそれぞれに対して、CREATE SERVER 文を実行します。

例

CREATE SERVER 文の USING 句で接続文字列を指定して、SQL Anywhere データベースに接続します。

```
CREATE SERVER TestSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 17;HOST=myhost;Server=TestSA;DBN=sample';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.16.4 サーバクラス MIRROR

サーバクラス MIRROR のリモートサーバは、SQL Anywhere データベースサーバです。

MIRROR サーバクラスは、ODBC を介してリモート SQL Anywhere サーバへの接続を行います。ただし、リモートサーバの作成時には、SYS.SYSMIRRORSERVER カタログテーブルにあるミラーサーバ名が USING 句に含まれます。リモートデータアクセスレイヤは、このミラーサーバ名を使用して、リモート SQL Anywhere サーバに対する接続文字列を作成します。

注記

リモートデータアクセスミラーサーバのテーブルにマッピングされたプロキシテーブルを照会すると、リモートデータアクセスレイヤは SYS.SYSMIRRORSERVER カタログテーブルと SYS.SYSMIRRORSERVEROPTION カタログテーブルの両方を参照して、リモートデータアクセスミラーサーバによって指定された SA サーバへの接続を確立するために使用する接続文字列を決定します。

例

MyMirrorServer に接続するためのリモートデータアクセスミラーサーバを設定するには、次のような文を実行します。

```
CREATE SERVER remote_server_name
CLASS 'MIRROR'
USING 'MirrorServer=MyMirrorServer';
```

i 注記

他のリモートデータアクセスサーバのクラスとは異なり、リモートデータミラーアクセスサーバへの接続は、リモート接続が切断されると自動的に再接続されます。

1.8.16.5 サーバクラス ULODBC

サーバクラスが ULODBC のリモートサーバは Ultra Light データベースサーバです。

Ultra Light データベースへの接続を定義する ODBC データソース名を作成します。ODBC データソース名に対して、CREATE SERVER 文を実行します。

Ultra Light は、SQL Anywhere で使用可能なデータ型のサブセットをサポートしているため、Ultra Light のデータ型と SQL Anywhere のデータ型には 1 対 1 のマッピングが存在します。

i 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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

CREATE SERVER 文の USING 句で接続文字列を指定して、Ultra Light データベースに接続します。

```
CREATE SERVER TestUL
CLASS 'ULODBC'
USING 'DRIVER=UltraLite 17;UID=DBA;PWD=sql;DBF=custdb.udb'
```

1.8.16.6 サーバクラス ADSODBC

CREATE TABLE 文を実行すると、SQL Anywhere は、データ型を SAP Advantage Database Server の対応するデータ型に自動的に変換します。次の表に、SQL Anywhere から SAP Advantage Database Server へのデータ型変換を示します。

SQL Anywhere のデータ型	Advantage Database Server のデフォルトデータ型
BIT	Logical
VARBIT(n)	Binary(n)
LONG VARBIT	Binary(2G)
TINYINT	Integer
SMALLINT	Integer
INTEGER	Integer
BIGINT	Numeric(32)
UNSIGNED TINYINT	Numeric(11)
UNSIGNED SMALLINT	Numeric(11)
UNSIGNED INTEGER	Numeric(11)
UNSIGNED BIGINT	Numeric(32)
CHAR(n)	Character(n)
VARCHAR(n)	VarChar(n)
LONG VARCHAR	VarChar(65000)
NCHAR(n)	NChar(n)
NVARCHAR(n)	NVarChar(n)
LONG NVARCHAR	NVarChar(32500)
BINARY(n)	Binary(n)
VARBINARY(n)	Binary(n)
LONG BINARY	Binary(2G)
DECIMAL(precision, scale)	Numeric(precision+3)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	Advantage Database Server のデフォルトデータ型
NUMERIC(precision, scale)	Numeric(precision+3)
SMALLMONEY	Money
MONEY	Money
REAL	Double
DOUBLE	Double
FLOAT(n)	Double
DATE	Date
TIME	Time
TIMESTAMP	TimeStamp
TIMESTAMP WITH TIME ZONE	Char(254)
XML	Binary(2G)
ST_GEOMETRY	Binary(2G)
UNIQUEIDENTIFIER	Binary(2G)

1.8.16.7 サーバクラス ASEODBC

サーバクラスが ASEODBC であるリモートサーバは、Adaptive Server Enterprise (バージョン 10 以降) データベースサーバです。

SQL Anywhere では、クラスが ASEODBC であるリモートの Adaptive Server Enterprise データベースサーバに接続するために、Adaptive Server Enterprise ODBC ドライバと Open Client 接続ライブラリのインストールが必要です。

注記

- Open Client はバージョン 11.1.1、EBF 7886 以降が必要です。Open Client をインストールして Adaptive Server Enterprise サーバへの接続を検証してから、ODBC をインストールして SQL Anywhere を設定してください。
  テスト済みの SAP Adaptive Server Enterprise ODBC ドライバの最新バージョンは、SDK 15.7 SP110 です。
- quoted_identifier オプションのローカル設定は、Adaptive Server Enterprise の引用符付き識別子の使用を制御します。たとえば、quoted_identifier オプションをローカルで Off に設定すると、Adaptive Server Enterprise に対して引用符付き識別子がオフになります。
- *Configuration Manager* でユーザデータソースを次の属性で設定します。

  一般タブ

  Data Source Name に任意の値を入力します。この値は、CREATE SERVER 文の USING 句に使用されます。

  サーバ名は interfaces ファイルにあるサーバ名と一致させてください。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

詳細タブ

Application Using Threads オプションと *Enable Quoted Identifiers* オプションをクリックします。

接続タブ

[charset] フィールドを、SQL Anywhere の文字セットに一致するように設定します。

language フィールドを、エラーメッセージを表示したい言語に設定します。

Performance タブ

Prepare メソッドを **2-Full** に設定します。

最高のパフォーマンスを得るには、*[Fetch Array Size]* をできるだけ大きな値に設定します。これはメモリ内にキャッシュされるローの数なので、この値を大きくすると必要なメモリ量が増大します。Adaptive Server Enterprise では 100 の値を使用することをお奨めします。

Select Method を **0-Cursor** に設定します。

Packet Size をできるだけ大きな値に設定します。Adaptive Server Enterprise では -1 の値を使用することをお奨めします。

Connection Cache を 1 に設定します。

データ型変換: ODBC と Adaptive Server Enterprise

CREATE TABLE 文を実行するときに、SQL Anywhere は、データ型を Adaptive Server Enterprise の対応するデータ型に自動的に変換します。次の表に、SQL Anywhere から Adaptive Server Enterprise へのデータ型変換を示します。

SQL Anywhere のデータ型	Adaptive Server Enterprise のデフォルトデータ型
BIT	bit
VARBIT(n)	if (n <= 255) varbinary(n) else image
LONG VARBIT	image
TINYINT	tinyint
SMALLINT	smallint
INT, INTEGER	int
BIGINT	numeric(20,0)
UNSIGNED TINYINT	tinyint
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	numeric(11,0)
UNSIGNED BIGINT	numeric(20,0)
CHAR(n)	if (n <= 255) char(n) else text
VARCHAR(n)	if (n <= 255) varchar(n) else text
LONG VARCHAR	text
NCHAR(n)	if (n <= 255) nchar(n) else ntext

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	Adaptive Server Enterprise のデフォルトデータ型
NVARCHAR(n)	if (n <= 255) nvarchar(n) else ntext
LONG NVARCHAR	ntext
BINARY(n)	if (n <= 255) binary(n) else image
VARBINARY(n)	if (n <= 255) varbinary(n) else image
LONG BINARY	image
DECIMAL(prec,scale)	decimal(prec,scale)
NUMERIC(prec,scale)	numeric(prec,scale)
SMALLMONEY	numeric(10,4)
MONEY	numeric(19,4)
REAL	real
DOUBLE	float
FLOAT(n)	float(n)
DATE	datetime
TIME	datetime
SMALLDATETIME	smalldatetime
TIMESTAMP	datetime
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	text
ST_GEOMETRY	image
UNIQUEIDENTIFIER	binary(16)

例

CREATE SERVER 文の USING 句で接続文字列を指定して、Adaptive Server Enterprise 16 データベースに接続します。

```
CREATE SERVER TestASE
CLASS 'ASEODBC'
USING 'DRIVER=SAP ASE ODBC
Driver;Server=TestASE;Port=5000;Database=testdb;UID=username;PWD=password'
```

Adaptive Server Enterprise 12 以前のドライバ名は、*Sybase ASE ODBC ドライバ*です。

Adaptive Server Enterprise 15 のドライバ名は、*Adaptive Server Enterprise* です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.16.8 サーバクラス DB2ODBC

サーバクラスが DB2ODBC のリモートサーバは IBM DB2 データベースサーバです。

注記

- SAP は、IBM の DB2 Connect バージョン 5 (修正パック WR09044 付き) の使用を確認しています。この製品の説明に従って、ODBC 構成の設定とテストを実行してください。SQL Anywhere には、IBM DB2 データソースの設定について特別な要件はありません。
- 以下は、mydb2 という ODBC データソースを持つ IBM DB2 サーバの CREATE EXISTING TABLE 文の例です。

```
CREATE EXISTING TABLE ibmcol
AT 'mydb2..sysibm.syscolumns';
```

データ型変換: IBM DB2

CREATE TABLE 文を実行するときに、SQL Anywhere は、データ型を IBM DB2 の対応するデータ型に自動的に変換します。

次の表に、SQL Anywhere から IBM DB2 へのデータ型変換を示します。

SQL Anywhere のデータ型	IBM DB2 のデフォルトデータ型
BIT	smallint
VARBIT(n)	if (n <= 4000) varchar(n) for bit data else long varchar for bit data
LONG VARBIT	long varchar for bit data
TINYINT	smallint
SMALLINT	smallint
INTEGER	int
BIGINT	decimal(20,0)
UNSIGNED TINYINT	int
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	decimal(11,0)
UNSIGNED BIGINT	decimal(20,0)
CHAR(n)	if (n < 255) char(n) else if (n <= 4000) varchar(n) else long varchar
VARCHAR(n)	if (n <= 4000) varchar(n) else long varchar
LONG VARCHAR	long varchar

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	IBM DB2 のデフォルトデータ型
NCHAR(n)	使用不可
NVARCHAR(n)	使用不可
LONG NVARCHAR	使用不可
BINARY(n)	if (n <= 4000) varchar(n) for bit data else long varchar for bit data
VARBINARY(n)	if (n <= 4000) varchar(n) for bit data else long varchar for bit data
LONG BINARY	long varchar for bit data
DECIMAL(prec,scale)	decimal(prec,scale)
NUMERIC(prec,scale)	decimal(prec,scale)
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
REAL	real
DOUBLE	float
FLOAT(n)	float(n)
DATE	date
TIME	time
TIMESTAMP	timestamp
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	long varchar for bit data
ST_GEOMETRY	long varchar for bit data
UNIQUEIDENTIFIER	varchar(16) for bit data

1.8.16.9 サーバクラス HANAODBC

サーバクラスが HANAODBC のリモートサーバは SAP HANA データベースサーバです。

注記

- 以下は、mySAPHANA という ODBC データソースを持つ SAP HANA データベースサーバの CREATE EXISTING TABLE 文の例です。

```
CREATE EXISTING TABLE hanatable
AT 'mySAPHANA..dbo.hanatable';
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

データ型変換:SAP HANA

CREATE TABLE 文を実行するときに、SQL Anywhere は、データ型を SAP HANA の対応するデータ型に自動的に変換します。次の表に、SQL Anywhere から SAP HANA へのデータ型変換を示します。

SQL Anywhere のデータ型	SAP HANA のデフォルトデータ型
BIT	TINYINT
VARBIT(n)	if (n <= 5000) VARBINARY(n) else BLOB
LONG VARBIT	BLOB
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
UNSIGNED TINYINT	TINYINT
UNSIGNED SMALLINT	INTEGER
UNSIGNED INTEGER	BIGINT
UNSIGNED BIGINT	DECIMAL(20,0)
CHAR(n)	if (n <= 5000) VARCHAR(n) else CLOB
VARCHAR(n	if (n <= 5000) VARCHAR(n) else CLOB
LONG VARCHAR	CLOB
NCHAR(n)	if (n <= 5000) NVARCHAR(n) else NCLOB
NVARCHAR(n)	if (n <= 5000) NVARCHAR(n) else NCLOB
LONG NVARCHAR	NCLOB
BINARY(n)	if (n <= 5000) VARBINARY(n) else BLOB
VARBINARY(n)	if (n <= 5000) VARBINARY(n) else BLOB
LONG BINARY	BLOB
DECIMAL(precision, scale)	DECIMAL(precision, scale)
NUMERIC(precision, scale)	DECIMAL(precision, scale)
SMALLMONEY	DECIMAL(13,4)
MONEY	DECIMAL(19,4)
REAL	REAL
DOUBLE	FLOAT
FLOAT(n)	FLOAT
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH TIME ZONE	VARCHAR(254)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	SAP HANA のデフォルトデータ型
XML	BLOB
ST_GEOMETRY	BLOB
UNIQUEIDENTIFIER	VARBINARY(16)

1.8.16.10 サーバクラス IQODBC

サーバクラスが IQODBC のリモートサーバは、SAP IQ データベースサーバです。

SAP IQ データソースの設定には、特別な要件はありません。

複数のデータベースをサポートする SAP IQ データベースサーバにアクセスするには、各データベースへの接続を定義する ODBC データソース名を作成します。これらの ODBC データソース名のそれぞれに対して、CREATE SERVER 文を実行します。

関連情報

CREATE SERVER 文の USING 句 [707 ページ]

1.8.16.11 サーバクラス MSACCESSODBC

Microsoft Access データベースは .mdb ファイルに格納されます。ODBC マネージャを使用して、ODBC データソースを作成し、これらのファイルの 1 つにマッピングします。

新しい .mdb ファイルは、ODBC マネージャを使って作成できます。SQL Anywhere でテーブルを作成するときに別のファイルを指定しないと、このデータベースファイルがデフォルトになります。

ODBC データソースが access という名前であると仮定した場合、次のいずれかの文を使用してデータにアクセスできます。

- ```
  CREATE TABLE tab1 (a int, b char(10))
  AT 'access...tab1';
  ```
- ```
  CREATE TABLE tab1 (a int, b char(10))
  AT 'access;d:¥¥pcdb¥¥data.mdb;;tab1';
  ```
- ```
  CREATE EXISTING TABLE tab1
  AT 'access;d:¥¥pcdb¥¥data.mdb;;tab1';
  ```

Microsoft Access では所有者名の修飾をサポートしないので、これはブランクのままにしてください。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.
  ```

データ型変換:Microsoft Access

SQL Anywhere のデータ型	Microsoft Access のデフォルトデータ型
BIT	TINYINT
VARBIT(n)	if (n <= 4000) BINARY(n) else IMAGE
LONG VARBIT	IMAGE
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	DECIMAL(19,0)
UNSIGNED TINYINT	TINYINT
UNSIGNED SMALLINT	INTEGER
UNSIGNED INTEGER	DECIMAL(11,0)
UNSIGNED BIGINT	DECIMAL(20,0)
CHAR(n)	if (n < 255) CHARACTER(n) else TEXT
VARCHAR(n)	if (n < 255) CHARACTER(n) else TEXT
LONG VARCHAR	TEXT
NCHAR(n)	Not supported
NVARCHAR(n)	Not supported
LONG NVARCHAR	Not supported
BINARY(n)	if (n <= 4000) BINARY(n) else IMAGE
VARBINARY(n)	if (n <= 4000) BINARY(n) else IMAGE
LONG BINARY	IMAGE
DECIMAL(precision, scale)	DECIMAL(precision, scale)
NUMERIC(precision, scale)	DECIMAL(precision, scale)
SMALLMONEY	MONEY
MONEY	MONEY
REAL	REAL
DOUBLE	FLOAT
FLOAT(n)	FLOAT
DATE	DATETIME
TIME	DATETIME
TIMESTAMP	DATETIME
TIMESTAMP WITH TIME ZONE	CHARACTER(254)
XML	XML
ST_GEOMETRY	IMAGE

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	Microsoft Access のデフォルトデータ型
UNIQUEIDENTIFIER	BINARY(16)

1.8.16.12 サーバクラス MSSODBC

サーバクラス MSSODBC を使用し、そのいずれかの ODBC ドライバを介して、Microsoft SQL Server にアクセスします。

注記

- これまでに使用されてきた Microsoft SQL Server ODBC ドライバのバージョンは次のとおりです。
  - Microsoft SQL Server ODBC Driver バージョン 06.01.7601
  - Microsoft SQL Server Native Client バージョン 10.00.1600
- 次に、Microsoft SQL Server の例を示します。

```
CREATE SERVER mysqlserver
CLASS 'MSSODBC'
USING 'DSN=MSSODBC_cli';
CREATE EXISTING TABLE accounts
AT 'mysqlserver.master.dbo.accounts';
```

- quoted_identifier オプションのローカル設定は、Microsoft SQL Server の引用符付き識別子の使用を制御します。たとえば、quoted_identifier オプションをローカルで Off に設定すると、Microsoft SQL Server に対して引用符付き識別子がオフになります。

データ型変換: Microsoft SQL Server

CREATE TABLE 文を実行するときに、SQL Anywhere は、次のデータ型変換を使用して、データ型を Microsoft SQL Server の対応するデータ型に自動的に変換します。

SQL Anywhere のデータ型	Microsoft SQLServer のデフォルトデータ型
BIT	bit
VARBIT(n)	if (n <= 255) varbinary(n) else image
LONG VARBIT	image
TINYINT	tinyint
SMALLINT	smallint
INTEGER	int
BIGINT	numeric(20,0)
UNSIGNED TINYINT	tinyint

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	Microsoft SQLServer のデフォルトデータ型
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	numeric(11,0)
UNSIGNED BIGINT	numeric(20,0)
CHAR(n)	if (n <= 255) char(n) else text
VARCHAR(n)	if (n <= 255) varchar(n) else text
LONG VARCHAR	text
NCHAR(n)	if (n <= 4000) nchar(n) else ntext
NVARCHAR(n)	if (n <= 4000) nvarchar(n) else ntext
LONG NVARCHAR	ntext
BINARY(n)	if (n <= 255) binary(n) else image
VARBINARY(n)	if (n <= 255) varbinary(n) else image
LONG BINARY	image
DECIMAL(precision, scale)	decimal(precision, scale)
NUMERIC(precision, scale)	numeric(precision, scale)
SMALLMONEY	smallmoney
MONEY	money
REAL	real
DOUBLE	float
FLOAT(n)	float(n)
DATE	datetime
TIME	datetime
SMALLDATETIME	smalldatetime
DATETIME	datetime
TIMESTAMP	datetime
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	xml
ST_GEOMETRY	image
UNIQUEIDENTIFIER	binary(16)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.16.13 サーバクラス MYSQLODBC

CREATE TABLE 文を実行すると、SQL Anywhere は、データ型を Oracle MySQL の対応するデータ型に自動的に変換します。

SQL Anywhere のデータ型	MySQL のデフォルトデータ型
BIT	bit(1)
VARBIT(n)	if (n <= 4000) varbinary(n) else longblob
LONG VARBIT	longblob
TINYINT	tinyint unsigned
SMALLINT	smallint
INTEGER	int
BIGINT	bigint
UNSIGNED TINYINT	tinyint unsigned
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	bigint
UNSIGNED BIGINT	decimal(20,0)
CHAR(n)	if (n < 255) char(n) else if (n <= 4000) varchar(n) else longtext
VARCHAR(n)	if (n <= 4000) varchar(n) else longtext
LONG VARCHAR	longtext
NCHAR(n)	if (n < 255) national character(n) else if (n <= 4000) national character varying(n) else longtext
NVARCHAR(n)	if (n <= 4000) national character varying(n) else longtext
LONG NVARCHAR	longtext
BINARY(n)	if (n <= 4000) varbinary(n) else longblob
VARBINARY(n)	if (n <= 4000) varbinary(n) else longblob
LONG BINARY	longblob
DECIMAL(precision, scale)	decimal(precision, scale)
NUMERIC(precision, scale)	decimal(precision, scale)

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	MySQL のデフォルトデータ型
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
REAL	real
DOUBLE	float
FLOAT(n)	float(n)
DATE	date
TIME	time
TIMESTAMP	datetime
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	longblob
ST_GEOMETRY	longblob
UNIQUEIDENTIFIER	varbinary(16)

例

CREATE SERVER 文の USING 句で接続文字列を指定して、Oracle MySQL データベースに接続します。

```
CREATE SERVER TestMySQL
CLASS 'MYSQLODBC'
USING 'DRIVER=MySQL ODBC 5.1
Driver;DATABASE=mydatabase;SERVER=mySQLHost;UID=me;PWD=secret'
```

1.8.16.14 サーバクラス ODBC

独自のサーバクラスを持たない ODBC データソースでは、ODBC サーバクラスを使用します。

任意の ODBC ドライバを使用できます。SAP は次の ODBC データソースの使用を確認しています。

- Microsoft Excel (Microsoft 3.51.171300)
- Microsoft Visual FoxPro (Microsoft 3.51.171300)
- Lotus Notes SQL

最新バージョンの Microsoft ODBC ドライバは、Microsoft Data Access Components (MDAC) とともに、Microsoft ダウンロードセンタからダウンロードできます。MDAC 2.0 には、上に示すバージョンの Microsoft ドライバが含まれています。

このセクションの内容:

Microsoft Excel (Microsoft 3.51.171300) [722 ページ]
Microsoft Excel では、それぞれの Microsoft Excel ブックはいくつかのテーブルを持つデータベースであると、論理的に考えられます。

Microsoft Visual FoxPro (Microsoft 3.51.171300) [723 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Microsoft Visual FoxPro の複数のテーブルを 1 つの Microsoft Visual FoxPro データベースファイル (.dbc) にまとめて格納することもできますし、各テーブルを個別の .dbf ファイルに格納することもできます。

Lotus Notes SQL [723 ページ]
SQL Anywhere テーブルを Notes フォームに簡単にマッピングし、Lotus Notes の連絡先にアクセスするよう SQL Anywhere を設定することができます。

1.8.16.14.1 Microsoft Excel (Microsoft 3.51.171300)

Microsoft Excel では、それぞれの Microsoft Excel ブックはいくつかのテーブルを持つデータベースであると、論理的に考えられます。

テーブルはブック内のシートにマッピングされます。ODBC ドライバマネージャで ODBC データソース名を設定する場合は、そのデータソースに関連付けられたデフォルトのブック名を指定します。ただし、CREATE TABLE 文を実行する場合には、デフォルトを無効にしてロケーションの文字列にブック名を指定できます。これによって、1 つの ODBC DSN を使用してすべての Microsoft Excel ブックにアクセスできます。

Microsoft Excel ODBC ドライバに接続する excel という名前のリモートサーバを作成します。

```
CREATE SERVER excel
CLASS 'ODBC'
USING 'DRIVER=Microsoft Excel Driver (*.xls);DBQ=d:¥
¥work1.xls;READONLY=0;DriverID=790'
```

mywork というシート (テーブル) を持つ work1.xls というブックを作成するには、次のようにします。

```
CREATE TABLE mywork (a int, b char(20))
AT 'excel;d:¥¥work1.xls;;mywork';
```

2 番目のシート (テーブル) を作成するには、次のような文を実行します。

```
CREATE TABLE mywork2 (x float, y int)
AT 'excel;d:¥¥work1.xls;;mywork2';
```

CREATE EXISTING 文を使用して、既存のシートを SQL Anywhere にインポートできます。ここでは、シートの最初のローには、カラム名が入ることを前提としています。

```
CREATE EXISTING TABLE mywork
AT'excel;d:¥¥work1;;mywork';
```

SQL Anywhere によって、テーブルが見つからないと表示された場合は、マッピングするカラムとローの範囲を明示的に指定する必要があります。例:

```
CREATE EXISTING TABLE mywork
AT 'excel;d:¥¥work1;;mywork$';
```

シート名に $ を追加すると、ワークシート全体が選択されることを示します。

AT で指定するロケーションの文字列で、フィールドセパレータとしてピリオドの代わりにセミコロンが使用されていることに注意してください。これは、ファイル名にピリオドが使用されるためです。Microsoft Excel では所有者名フィールドをサポートしないので、これはブランクのままにしてください。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

削除はサポートされていません。また、Microsoft Excel ドライバは位置付け更新をサポートしないため、更新も可能でない場合があります。

例

次の文は、ODBC DSN を使用して Microsoft Excel ワークブック LogFile.xlsx にアクセスし、そのシートを SQL Anywhere にインポートする、TestExcel という名前のデータベースサーバを作成します。

```
CREATE SERVER TestExcel
CLASS 'ODBC'
USING 'DRIVER=Microsoft Excel Driver (*.xls);DBQ=c:¥¥temp¥
¥LogFile.xlsx;READONLY=0;DriverID=790'
CREATE EXISTING TABLE MyWorkbook
AT 'TestExcel;c:¥¥temp¥¥LogFile.xlsx;;Logfile$';
SELECT * FROM MyWorkbook;
```

1.8.16.14.2 Microsoft Visual FoxPro (Microsoft 3.51.171300)

Microsoft Visual FoxPro の複数のテーブルを 1 つの Microsoft Visual FoxPro データベースファイル (.dbc) にまとめて格納することもできますし、各テーブルを個別の .dbf ファイルに格納することもできます。

.dbf ファイルを使用するときは、ファイル名がロケーションの文字列に入っていることを確認してください。入っていない場合は、SQL Anywhere が起動されたディレクトリが使用されます。

```
CREATE TABLE fox1 (a int, b char(20))
AT 'foxpro;d:¥¥pcdb;;fox1';
```

この文は、ODBC ドライバマネージャで *Free Table Directory* オプションが選択されている場合、d:¥pcdb¥fox1.dbf というファイルを作成します。

1.8.16.14.3 Lotus Notes SQL

SQL Anywhere テーブルを Notes フォームに簡単にマッピングし、Lotus Notes の連絡先にアクセスするよう SQL Anywhere を設定することができます。

前提条件

Lotus Notes SQL ドライバを取得する必要があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. Lotus Notes プログラムフォルダがパスに含まれていることを確認します (`C:¥Program Files (x86)¥IBM ¥Lotus¥Notes` など)。
2. NotesSQL ODBC ドライバを使用して 32 ビットの ODBC データソースを作成します。この例では `names.nsf` データベースを使用します。*[特殊文字のマッピング]* オプションをオンにしてください。この例では、データソース名は `my_notes_dsn` です。
3. 32 ビットのデータベースサーバに接続された Interactive SQL を使用して、リモートデータアクセスサーバを作成します。

結果

これで、Lotus Notes の連絡先にアクセスするよう SQL Anywhere が設定されました。

例

- リモートデータアクセスサーバを作成します。

```
CREATE SERVER NotesContacts
CLASS 'ODBC'
USING 'my_notes_dsn';
```

- Lotus Notes サーバ用の外部ログインを作成します。

```
CREATE EXTERNLOGIN "DBA" TO "NotesContacts"
REMOTE LOGIN 'John Doe/SAP' IDENTIFIED BY 'MyNotesPassword';
```

- Person フォームの一部のカラムを SQL Anywhere テーブルにマッピングします。

```
CREATE EXISTING TABLE PersonDetails
( DisplayName CHAR(254),
  DisplayMailAddress CHAR(254),
  JobTitle CHAR(254),
  CompanyName CHAR(254),
  Department CHAR(254),
  Location CHAR(254),
  OfficePhoneNumber CHAR(254) )
AT 'NotesContacts...Person';
```

- テーブルをクエリします。

```
SELECT * FROM PersonDetails
WHERE Location LIKE 'Waterloo%';
```

関連情報

IBM Lotus NotesSQL

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.8.16.15 サーバクラス ORAODBC

サーバクラスが ORAODBC のリモートサーバは、Oracle Database バージョン 8.0 以降です。

注記

- SAP は、Oracle Database バージョン 8.0.03 の ODBC ドライバの使用を確認しています。この製品の説明に従って、ODBC 構成の設定とテストを実行してください。
- 以下は、myora という Oracle Database サーバの CREATE EXISTING TABLE 文の例です。

```
CREATE EXISTING TABLE employees
AT 'myora.database.owner.employees';
```

データ型変換: Oracle Database

CREATE TABLE 文を実行するときに、SQL Anywhere は、次のデータ型変換を使用して、データ型を Oracle Database の対応するデータ型に自動的に変換します。

SQL Anywhere のデータ型	Oracle Database のデータ型
BIT	number(1,0)
VARBIT(n)	if (n <= 255) raw(n) else long raw
LONG VARBIT	long raw
TINYINT	number(3,0)
SMALLINT	number(5,0)
INTEGER	number(11,0)
BIGINT	number(20,0)
UNSIGNED TINYINT	number(3,0)
UNSIGNED SMALLINT	number(5,0)
UNSIGNED INTEGER	number(11,0)
UNSIGNED BIGINT	number(20,0)
CHAR(n)	if (n <= 255) char(n) else long
VARCHAR(n)	if (n <= 2000) varchar(n) else long
LONG VARCHAR	long
NCHAR(n)	if (n <= 255) nchar(n) else nclob

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

SQL Anywhere のデータ型	Oracle Database のデータ型
NVARCHAR(n)	if (n <= 2000) nvarchar(n) else nclob
LONG NVARCHAR	nclob
BINARY(n)	if (n > 255) long raw else raw(n)
VARBINARY(n)	if (n > 255) long raw else raw(n)
LONG BINARY	long raw
DECIMAL(precision, scale)	number(precision, scale)
NUMERIC(precision, scale)	number(precision, scale)
SMALLMONEY	numeric(13,4)
MONEY	number(19,4)
REAL	real
DOUBLE	float
FLOAT(n)	float
DATE	date
TIME	date
TIMESTAMP	date
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	long raw
ST_GEOMETRY	long raw
UNIQUEIDENTIFIER	raw(16)

例

CREATE SERVER 文の USING 句で接続文字列を指定して、Oracle データベースに接続します。

```
CREATE SERVER TestOracle
CLASS 'ORAODBC'
USING 'DRIVER=Oracle ODBC Driver;DBQ=mydatabase;UID=username;PWD=password'
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9 データ整合性

データに整合性があるということは、データが有効、つまり適切であり正確で、データベースの関係構造が保たれていることを意味します。

参照整合性制約を使うと、データベースの関係構造を確保できます。また、この規則によって、各テーブル間でデータの一貫性を保つことができます。データベースに整合性制約を構築すると、データの一貫性を確実に保つことができます。

参照整合性を確保する方法はいくつかあります。たとえば、テーブルとカラムに制約と検査制約を課すことで、各エントリが正しいことを保証できます。また、適切なデータ型を使ってカラムのプロパティを設定したり、特別なデフォルト値を設定したりすることもできます。

SQL Anywhere では、データベースへのデータの入力方法を細かく規定するためのストアドプロシージャをサポートしています。また、トリガも作成できます。トリガは特殊なストアドプロシージャで、特定のカラムの更新など、特定のアクションが行われると自動的に実行されます。

このセクションの内容:

データが有効でなくなる状況 [728 ページ]
　適切な検査が行われないと、データベース内のデータが有効でなくなる可能性があります。

整合性制約 [728 ページ]
　データベースのデータの有効性を確保するには、データが有効かどうかを判断するための検査と、データが従う規則 (ビジネスルール) を作成します。

データの整合性を維持するためのツール [729 ページ]
　データ整合性を確保するため、デフォルト値、データ制約、データベースの参照構造を保つための制約を使用します。

整合性制約を実装するための SQL 文 [730 ページ]
　SQL 文により、いくつかの方法で整合性制約を実装します。

カラムデフォルト [730 ページ]
　カラムデフォルトを設定すると、データベーステーブルに新しいローが追加された場合、特定のカラムに指定された値が設定されます。

テーブルとカラム制約 [738 ページ]
　CREATE TABLE 文と ALTER TABLE 文を使用して、データの精度と整合性を制御できるテーブル属性を指定することができます。

ドメインを使用したデータの整合性の向上方法 [743 ページ]
　ドメインとは、値の許容範囲を制限したりデフォルト値を設定したりできるユーザ定義データ型のことです。

エンティティ整合性と参照整合性 [746 ページ]
　データベースの関係構造によって、データベースサーバがデータベース内の情報を識別し、各テーブル内のすべてのローで (データベーススキーマに定義されている) テーブル間の関係が維持されていることを保証できます。

システムテーブルの整合性ルール [756 ページ]
　データベースの整合性検査と規則に関する情報は、すべてカタログに格納されています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 [87 ページ]

1.9.1 データが有効でなくなる状況

適切な検査が行われないと、データベース内のデータが有効でなくなる可能性があります。

それぞれのケースは、以下の機能を使用して回避できます。

正しくない情報

- オペレータが売り上げトランザクションの日付を間違って入力しました。
- オペレータが一桁間違えて入力し、社員の給料が 10 分の 1 になりました。

データの重複

- 組織のデータベースの Departments テーブルに、2 人の異なる従業員が同じ部署 (DepartmentID は 200) を新しく追加します。

失われた外部キー関係

- 300 という DepartmentID で識別される部署が閉鎖になったのに、1 人の従業員レコードだけが他の部署へ移されていません。

1.9.2 整合性制約

データベースのデータの有効性を確保するには、データが有効かどうかを判断するための検査と、データが従う規則 (ビジネスルール) を作成します。

通常、ビジネスルールは、検査制約、ユーザ定義データ型、適切なトランザクションを使用することによって実装されます。

データベースに組み込まれる制約は、クライアントアプリケーション内に組み込まれる制約や、データベースのユーザに指示される制約よりも高い信頼性を備えています。データベースに組み込まれた制約はデータベースの定義の一部であり、すべてのアプリケーションに対して常に適用されるからです。データベース内に制約が設定されると、それ以降に発生するすべてのデータベースとの対話に対してその制約が適用されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

これに対して、クライアントアプリケーションに組み込まれた制約は、アプリケーションが変更されるたびに無効となる可能性があります。また、複数のアプリケーションに組み込んだり、1 つのアプリケーションの複数箇所に組み込んだりする必要があります。

1.9.3 データの整合性を維持するためのツール

データ整合性を確保するため、デフォルト値、データ制約、データベースの参照構造を保つための制約を使用します。

デフォルト値

各エントリのデータの信頼性を高めるために、カラムにデフォルト値を設定できます。次に例を示します。

- すべてのユーザ、またはクライアントアプリケーションによるトランザクション日を記録するデフォルト値を、CURRENT DATE にします。
- 新しいローの追加以外で、特にユーザが操作しなくてもカラムのデフォルト値を自動的に 1 ずつ大きくしていきます。このようにすると、(発注書などの) 項の値をユニークに、また連続の値にできます。

プライマリキー

プライマリキーによって、指定されたテーブルのすべてのローをテーブル内でユニークに識別できることが保証されます。

テーブルとカラム制約

以下の制約によって、データベース内のデータ構造が維持され、リレーショナルデータベースのテーブル間の関係が定義されます。

参照整合性制約

データ整合性は **RI** 制約 (参照整合性制約の場合) とも呼ばれ、参照整合性制約を使用することによって維持することもできます。RI 制約は、カラムやテーブルに設定されるデータルールであり、データの形式を制御します。RI 制約は、リレーショナルデータベースのテーブル間の関係を定義します。

NOT NULL 制約

NOT NULL 値制約を適用すると、カラムに NULL 値が入力されるのを防ぐことができます。

検査制約

検査制約をカラムに適用すると、カラム内のすべての項目が一定の条件を満たすようにすることができます。たとえば、Salary (給与) カラムに上限と下限を設定して入力エラーを防げます。

検査制約を使って、それぞれのカラム値の差を検査できます。たとえば、図書館データベースで、DateReturned (返却日) エントリが DateBorrowed (貸出日) エントリよりも必ず後になるように指定できます。

カラム制約はドメインから継承できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

高度な整合性規則のためのトリガ

トリガはデータベースに格納されたプロシージャの一種で、特定のテーブル情報が修正されると自動的に実行されます。トリガはデータベース管理者や開発者にとって、データの信頼性を維持するための強力な手段です。データ整合性の確保にはトリガも使用できます。トリガを使うと検査条件をより高度な形で適用できます。

関連情報

カラムデフォルト [730 ページ]
プライマリキー [25 ページ]
エンティティ整合性と参照整合性 [746 ページ]
テーブルとカラム制約 [738 ページ]
ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 [87 ページ]

1.9.4 整合性制約を実装するための SQL 文

SQL 文により、いくつかの方法で整合性制約を実装します。

CREATE TABLE 文

テーブルの作成時の整合性制約を実装します。

ALTER TABLE 文

既存のテーブルに対して、整合性制約の追加や制約の修正を行います。

CREATE TRIGGER 文

より複雑なビジネスルールを確保するトリガを作成します。

CREATE DOMAIN 文

ユーザ定義のデータ型を作成します。データ型の定義には制約を含めることができます。

1.9.5 カラムデフォルト

カラムデフォルトを設定すると、データベーステーブルに新しいローが追加された場合、特定のカラムに指定された値が設定されます。

デフォルト値の設定には、クライアントアプリケーション側からの処理は特に必要ありません。一方、クライアントアプリケーションがカラムに値を設定すると、その値がデフォルト値に上書きされます。

カラムデフォルトは、ローが挿入された日付や時刻、入力するユーザのユーザ ID などの情報を、カラムにすばやく自動的に入力する場合に便利です。カラムデフォルトはデータの整合性を向上させますが、確実なものではありません。クライアントアプリケーションはデフォルト値を自由に上書きできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

@ で始まる変数を使用してデフォルト値が定義された場合、デフォルトとして使用される値は、DML 文または LOAD 文が実行されるときの変数の値です。

サポートされているデフォルト値

SQL では、次のデフォルト値をサポートしています。

- CREATE TABLE 文または ALTER TABLE 文で指定した文字列。
- CREATE TABLE 文または ALTER TABLE 文で指定した数値。
- AUTOINCREMENT: 自動的に増分される数値。既存のカラムの最大値に、自動的に 1 を加えます。
- GLOBAL AUTOINCREMENT: 複数のデータベースにユニークなプライマリキーが設定されるようにします。
- NEWID 関数を使用して生成されるユニバーサルユニーク識別子 (UUID)。
- CURRENT DATE、TIME、または TIMESTAMP。
- CURRENT SERVER DATE、CURRENT SERVER TIME、CURRENT SERVER TIMESTAMP。
- データベースユーザの CURRENT USER。
- NULL 値。
- データベースオブジェクトを参照していない定数式。

このセクションの内容:

カラムデフォルトの作成 [732 ページ]
カラムデフォルトは、テーブルの作成時に CREATE TABLE 文を使って作成するか、後で ALTER TABLE 文を使って追加します。

カラムデフォルトの変更 [732 ページ]
カラムデフォルトは、作成と同様に、ALTER TABLE 文を使って修正または削除できます。

現在の日付と時刻デフォルト [733 ページ]
データ型が DATE、TIME、TIMESTAMP のカラムには、CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP をデフォルトとして使用できます。

ユーザ ID デフォルト [734 ページ]
DEFAULT USER をカラムのデフォルト値として指定しておくと、そのデータをデータベースに入力したユーザを確実に特定できます。

オートインクリメントデフォルト [735 ページ]
オートインクリメントデフォルトは、値それ自体には意味がない数値データフィールドで役立ちます。

GLOBAL AUTOINCREMENT デフォルト [735 ページ]
GLOBAL AUTOINCREMENT デフォルトは、SQL Remote レプリケーション環境または Mobile Link 同期環境で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。

NEWID デフォルト [737 ページ]
ユニバーサルユニーク識別子 (UUID) を使用して、テーブルのユニークなローを識別できます。UUID は、グローバルユニーク識別子 (GUID) とも呼ばれます。

NULL デフォルト [737 ページ]
NULL 値を許容するカラムに NULL デフォルトを指定しても、デフォルトを指定しない場合と同じになります。ローを挿入するクライアントが値を指定しなければ、そのローには NULL 値が設定されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

[文字列と数値デフォルト [737 ページ]](#)
カラムが文字列または数値のデータ型であれば、特定の文字列または数値をデフォルトに指定できます。

[定数式デフォルト [738 ページ]](#)
定数式もデフォルト値として使用できます。ただし、データベースオブジェクトを参照していない場合にかぎります。

1.9.5.1 カラムデフォルトの作成

カラムデフォルトは、テーブルの作成時に CREATE TABLE 文を使って作成するか、後で ALTER TABLE 文を使って追加します。

例

次に示す文では、SalesOrders テーブルの ID カラムにデフォルト設定を追加して、クライアントアプリケーションが値を指定しない限り、自動的に 1 ずつ値を増加させます。SQL Anywhere サンプルデータベースでは、このカラムは AUTOINCREMENT に設定済みです。

```
ALTER TABLE SalesOrders
ALTER ID DEFAULT AUTOINCREMENT;
```

1.9.5.2 カラムデフォルトの変更

カラムデフォルトは、作成と同様に、ALTER TABLE 文を使って修正または削除できます。

次に示す文は、OrderDate というカラムのデフォルト値を、CURRENT DATE に変更します。

```
ALTER TABLE SalesOrders
ALTER OrderDate DEFAULT CURRENT DATE;
```

削除する場合は、カラムデフォルトに NULL を設定します。次に示す文は、OrderDate カラムからデフォルトを削除します。

```
ALTER TABLE SalesOrders
ALTER OrderDate DEFAULT NULL;
```

このセクションの内容:

[カラムデフォルトの設定 [733 ページ]](#)
SQL Central で、カラムデフォルトの追加、修正、削除を行うには、カラムプロパティウィンドウの値タブを使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.5.2.1　カラムデフォルトの設定

SQL Central で、カラムデフォルトの追加、修正、削除を行うには、カラムプロパティウィンドウの値タブを使用します。

前提条件

そのカラムが属してい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。

- 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

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テーブルをダブルクリックします。
3. テーブルをダブルクリックします。
4. カラムタブをクリックします。
5. カラムを右クリックして、*[プロパティ]* をクリックします。
6. 値タブをクリックします。
7. 必要に応じてカラムデフォルトを変更します。

結果

カラムのプロパティが変更されます。

1.9.5.3　現在の日付と時刻デフォルト

データ型が DATE、TIME、TIMESTAMP のカラムには、CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP をデフォルトとして使用できます。

指定するデフォルト値は、カラムのデータ型に一致させてください。

CURRENT DATE をデフォルトにすると便利な例

次のような場合、CURRENT DATE をデフォルト値として設定すると便利です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 顧客データベースで、電話があった日付を記録します。
- 売り上げ入力データベースで、注文の日付を記録します。
- 図書館データベースで、会員が本を借りた日を記録します。

CURRENT TIMESTAMP

CURRENT TIMESTAMP は、CURRENT DATE と同じように使えるデフォルト値で、さらに細かい情報を記録できます。たとえば、コンタクト管理アプリケーションのユーザは、一日に何度も同一の顧客とやり取りすることがあります。デフォルトを CURRENT TIMESTAMP に設定すると、それぞれの接触を区別しやすくなります。

CURRENT TIMESTAMP は日付と時刻を 100 万分の 1 秒単位で記録するので、データベースに記録されている各イベントの順序が重要である場合にも便利です。

DEFAULT TIMESTAMP

DEFAULT TIMESTAMP は、テーブル内の各ローが最後に修正された日時を示します。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。

1.9.5.4 ユーザ ID デフォルト

DEFAULT USER をカラムのデフォルト値として指定しておくと、そのデータをデータベースに入力したユーザを確実に特定できます。

たとえば、売り上げ歩合制の営業員がデータベースを使用する場合、このような情報が必要になります。

ユーザ ID デフォルトをテーブルのプライマリキーに設定すると、不定期に接続するユーザがいる場合に情報更新の競合を防ぐことができます。このようなユーザは、データベースから必要な部分を携帯端末にコピーして、マルチユーザデータベースに接続していない状態でデータを修正し、後でサーバにアクセスしてトランザクションログを送ることができます。

LAST USER 特別値は、ローを最後に更新したユーザの名前を返します。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、別々のローに記録されます)。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.5.5 オートインクリメントデフォルト

オートインクリメントデフォルトは、値それ自体には意味がない数値データフィールドで役立ちます。

この機能は、新しく作成されたローの該当するカラムに、カラムの他の値よりも大きいユニークな値を割り当てます。注文伝票番号の記録、顧客からの問い合わせの電話の識別など、番号自体に意味がないエントリ番号のカラムに、オートインクリメントを適用できます。

AUTOINCREMENT カラムは、通常はプライマリキーカラム、つまりユニークな値を保持するように制約されたカラムになります。

AUTOINCREMENT カラムへ直前に追加された値は、グローバル変数 @@identity を使って取得できます。

AUTOINCREMENT と負の値

AUTOINCREMENT は正の整数を想定しています。

テーブルが作成されたときの AUTOINCREMENT の初期値は 0 です。意図的に負の値が設定されたカラムが挿入されると、初期値 0 はそのカラムの最大値としてそのまま残されます。特に初期値が設定されていない挿入に対しては、オートインクリメントにより 1 が設定され、それ以降も必ず正の値が設定されます。

AUTOINCREMENT と IDENTITY カラム

Transact-SQL アプリケーションでは、オートインクリメントのデフォルト値が設定されるカラムを IDENTITY カラムと呼びます。

関連情報

GLOBAL AUTOINCREMENT デフォルト [735 ページ]
ユニークな値を生成するためのシーケンスの使用 [830 ページ]
エンティティ整合性 [747 ページ]
特殊な IDENTITY カラム [537 ページ]

1.9.5.6 GLOBAL AUTOINCREMENT デフォルト

GLOBAL AUTOINCREMENT デフォルトは、SQL Remote レプリケーション環境または Mobile Link 同期環境で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。

複数のデータベースにユニークなプライマリキーを保証します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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。SQL Anywhere 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。

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。

このオプションを使用する場合、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 to (n + 1) p 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 (n + 1) p 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 同期を介して別のデータベースからレプリケートされた場合に、このような値が存在する可能性があります。

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。たとえば、テーブルのプライマリキーにカラムが含まれている場合に、この状況が発生します。

デフォルトの NULL 値は、分割で値が不足したときにも生成されます。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。カラムで NULL が許可されていない場合に NULL 値を挿入しようとすると、エラーが発生します。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。

GLOBAL AUTOINCREMENT カラムは、通常はプライマリキーカラム、つまりユニークな値を保持するように制約されたカラムになります。

これ以外の場合にも GLOBAL AUTOINCREMENT のデフォルトを適用できますが、データベースのパフォーマンスが低下することがあります。たとえば、各カラムの次の値が 64 ビットの符号付き整数として格納されている場合に、2^{31} - 1 より大きい値または大きい double または numeric の値が使用されると、オーバーフローが発生して負の値になることがあります。

AUTOINCREMENT カラムへ直前に追加された値は、グローバル変数 @@identity を使って取得できます。

関連情報

[エンティティ整合性 [747 ページ]](#)
[オートインクリメントデフォルト [735 ページ]](#)
[ユニークな値を生成するためのシーケンスの使用 [830 ページ]](#)

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.5.7 NEWID デフォルト

ユニバーサルユニーク識別子 (UUID) を使用して、テーブルのユニークなローを識別できます。UUID は、グローバルユニーク識別子 (GUID) とも呼ばれます。

この値は、1 台のコンピュータで生成された値が、他のコンピュータで生成された値と一致しないように生成されます。したがって、これらの値は、レプリケーション環境と同期環境でキーとして使用できます。

プライマリキーとして使用する場合、GLOBAL AUTOINCREMENT 値と比べると、UUID 値にはいくつかのトレードオフがあります。次に例を示します。

- UUID はリモートデータベースごとにユニークなデータベース ID を割り当てる必要がないため、GLOBAL AUTOINCREMENT より簡単に設定できます。また、システムのデータベース数や、それぞれのテーブルのローの数を考慮する必要もありません。抽出ユーティリティ (dbxtract) を使用してデータベース ID の割り当てを処理できます。GLOBAL AUTOINCREMENT では通常、BIGINT データ型を使用する場合はこの点を考慮する必要はありませんが、BIGINT より小さいデータ型を使用する場合は考慮する必要があります。
- UUID 値は GLOBAL AUTOINCREMENT に必要な値よりかなり大きいため、プライマリテーブルと外部テーブルの両方でより多くのテーブル領域が必要です。また、UUID を使用すると、これらのカラムのインデックスの効率も悪くなります。つまり、GLOBAL AUTOINCREMENT の方がパフォーマンスに優れています。
- UUID には暗黙的な順序付けがありません。たとえば、A と B が UUID 値で、A が B よりも大きい場合に、A と B が同じコンピュータ上で生成されたとしても、A が B の後で生成されたとはかぎりません。暗黙的な順序付けが必要な場合は、追加のカラムとインデックスが必要になります。

1.9.5.8 NULL デフォルト

NULL 値を許容するカラムに NULL デフォルトを指定しても、デフォルトを指定しない場合と同じになります。ローを挿入するクライアントが値を指定しなければ、そのローには NULL 値が設定されます。

NULL デフォルトは、カラムの入力を省略できる場合、または入力データが入手できないことがある場合に使います。

1.9.5.9 文字列と数値デフォルト

カラムが文字列または数値のデータ型であれば、特定の文字列または数値をデフォルトに指定できます。

指定するデフォルト値は、カラムのデータ型に変換可能なものにしてください。

文字列と数値デフォルトは、カラムに同じデータを入力することが多いときに便利です。たとえば、ある会社に本社 city_1 と小規模の事業所 city_2 の 2 つのオフィスがある場合、所在地カラムのデフォルトを city_1 にしておけば入力が簡単になります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.5.10 定数式デフォルト

定数式もデフォルト値として使用できます。ただし、データベースオブジェクトを参照していない場合にかぎります。

たとえば次の式では、本日より 15 日後というエントリを、デフォルト値としてカラムに設定できます。

```
... DEFAULT ( DATEADD( day, 15, GETDATE() ) );
```

1.9.6 テーブルとカラム制約

CREATE TABLE 文と ALTER TABLE 文を使用して、データの精度と整合性を制御できるテーブル属性を指定することができます。

制約を使って、カラムに含まれる値や、異なるカラムに含まれるそれぞれの値の関係に制限を課すことができます。制約は、テーブル全体、または個別のカラムに対して適用できます。

このセクションの内容:

カラムに対する検査制約 [738 ページ]
　カラムの値がある基準やルールを満たすようにするため、検査条件を使用します。

テーブルに対する検査制約 [740 ページ]
　通常、制約としてテーブルに適用された検査条件によって、ロー内の 2 つの値を定義された関係に準拠するようにします。

ドメインから継承されるカラム検査制約 [740 ページ]
　ドメインに検査制約を付加できます。そのドメインで定義されたカラムには、検査制約も継承されます。

制約の管理 [741 ページ]
　SQL Central を使用して、カラム制約の追加、変更、削除を、テーブルまたは カラムプロパティウィンドウの制約タブで行います。

一意性制約の追加 [742 ページ]
　SQL Central のカラムの一意性制約を作成または削除します。

検査制約の変更と削除の方法 [743 ページ]
　テーブル上の既存の検査制約を変更するには、いくつかの方法があります。

1.9.6.1 カラムに対する検査制約

カラムの値がある基準やルールを満たすようにするため、検査条件を使用します。

これらのルールや基準は、データが正しいかどうかを確認するためのものであったり、企業のポリシーや内部規定などが反映される厳密なものであったりします。個々のカラムに対する検査条件は、カラム内の値を一定の範囲に収める必要がある場合に便利です。

検査条件が使用されると、その後はローの修正前に値が条件に対して評価されます。検査制約が設定された値を更新すると、その値の制約が検査されるだけでなく、残りのローの制約も検査されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

カラムに対する検査制約に変数は使用できません。カラム検査制約内で @ で始まる文字列は、制約が有効にされているカラムの名前で置き換えられます。

カラムのデータ型がドメインの場合、カラムはドメインに定義されている検査制約を継承します。

注記

カラムの検査は、FALSE が返された場合にエラーになります。UNKNOWN が返されても、動作は TRUE が返される場合と同じで、その値が受け入れられます。

例

例 1

データのフォーマットを規定します。たとえば、テーブルに電話番号のカラムがあるとして、その電話番号カラムが同じフォーマットで入力されるようにします。北米地域の電話番号に対する制約の例を次に示します。

```
ALTER TABLE Customers
ALTER Phone
CHECK ( Phone LIKE '(___) ___-____' );
```

この検査条件が使用されると、たとえば Phone の値を 9835 に設定しようとしても、変更されません。

例 2

エントリに入力されるデータが、あらかじめ決められたいくつかの値のいずれかになるように設定できます。たとえば、City カラムに、その会社の事業所の所在地など、いくつかの許可された都市のいずれかしか入力できないようにするには、次のように制約を使用します。

```
ALTER TABLE Customers
ALTER City
CHECK ( City IN ( 'city_1', 'city_2', 'city_3' ) );
```

データベースの作成時に特に指定をしなかった場合、文字列の比較において大文字と小文字は区別されません。

例 3

日付や数値が一定の範囲内に収まるように設定できます。従業員の入社日 StartDate を会社の創立から現在までの日付にする文の例を示します。StartDate がこの 2 つの日付の間になるように、次の制約を使用します。

```
ALTER TABLE Employees
ALTER StartDate
CHECK ( StartDate BETWEEN '1983/06/27'
                   AND CURRENT DATE );
```

日付形式はいくつかの種類が用意されています。ここで使用した YYYY/MM/DD 形式は、現在のオプション設定に関係なく必ず認識される特長があります。

関連情報

ドメインから継承されるカラム検査制約 [740 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.6.2 テーブルに対する検査制約

通常、制約としてテーブルに適用された検査条件によって、ロー内の 2 つの値を定義された関係に準拠するようにします。

制約に名前を付けると、制約は個別にシステムテーブル内に格納され、個別に置換、削除できます。この方法の方が柔軟性が高いため、検査制約に名前を付けるか、できるかぎり個別にカラム制約を使用することをお奨めします。

たとえば Employees テーブルに制約を追加して、TerminationDate が常に StartDate と同じかそれ以降になるようにすることができます。

```
ALTER TABLE Employees
   ADD CONSTRAINT valid_term_date
   CHECK( TerminationDate >= StartDate );
```

テーブル検査制約内には変数を指定できますが、変数の名前は @ で始まるようにします。使用される値は、DML 文または LOAD 文が実行されるときの変数の値です。

1.9.6.3 ドメインから継承されるカラム検査制約

ドメインに検査制約を付加できます。そのドメインで定義されたカラムには、検査制約も継承されます。

そのカラムに明示的に検査制約を設定した場合は、ドメインよりも優先されます。たとえば、このドメイン定義における CHECK 句は、カラムに挿入される値は正の整数だけであることを要求します。

```
CREATE DATATYPE positive_integer INT
CHECK ( @col > 0 );
```

positive_integer ドメインを使用して定義されたカラムでは、検査制約が明示的に指定されていないかぎり、正の整数のみが受け入れられます。@ 記号のプレフィクスを持つ変数はすべて、検査制約が評価される時点でそのカラムの名前に置き換えられるので、@ 記号のプレフィクスさえあれば @col 以外の変数名を使用しても問題ありません。

ALTER TABLE 文と DELETE CHECK 句を組み合せると、ドメインから継承されたものを含め、テーブル定義からすべての検査制約を削除できます。

ドメインでカラムが定義されたあとに、ドメイン定義の制約が変更された場合は、そのカラムに変更は適用されません。カラムは作成時にドメインの制約を継承しますが、それ以降は、両者の間に関係はありません。

関連情報

カラムに対する検査制約 [738 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.6.4 制約の管理

SQL Central を使用して、カラム制約の追加、変更、削除を、テーブルまたは カラムプロパティウィンドウの制約タブで行います。

前提条件

そのカラムが属してい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。

- 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

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テーブルをダブルクリックします。
3. 変更するテーブルをダブルクリックします。
4. 右ウィンドウ枠で、*[制約]* タブをクリックし、既存の制約を変更するか、新しい制約を追加します。

結果

カラムの制約が表示されます。

次のステップ

必要に応じて制約を修正します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.6.5 一意性制約の追加

SQL Central のカラムの一意性制約を作成または削除します。

前提条件

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。

- テーブルに対する ALTER 権限と、ALTER ANY INDEX、COMMENT ANY OBJECT、CREATE ANY INDEX、または CREATE ANY OBJECT システム権限のいずれか
- ALTER ANY TABLE システム権限と、ALTER ANY INDEX、COMMENT ANY OBJECT、CREATE ANY INDEX、または CREATE ANY OBJECT システム権限のいずれか
- ALTER ANY OBJECT システム権限

コンテキスト

一意性制約には空間カラムは設定できません。

カラムの場合、一意性制約では、カラム値をユニークにする必要があることを指定します。テーブルの場合、一意性制約では、テーブル内のユニークなローを識別する 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。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テーブルをダブルクリックします。
3. 変更するテーブルをクリックします。
4. 右ウィンドウ枠で、*[制約]* タブをクリックします。
5. 制約テーブルを右クリックし、新規 > 一意性制約 をクリックします。
6. 一意性制約作成ウィザードの指示に従います。

結果

一意性制約が作成されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.6.6 検査制約の変更と削除の方法

テーブル上の既存の検査制約を変更するには、いくつかの方法があります。

- テーブルまたはカラムに新しい検査制約を追加できます。
- カラムの検査制約を NULL に設定すると削除できます。次に、Customers テーブルの Phone カラムから検査制約を削除する文の例を示します。

```
ALTER TABLE Customers
ALTER Phone CHECK NULL;
```

- 検査制約の追加と同じ方法で、カラムの検査制約を置換できます。次に、Customers テーブルの Phone カラムの検査制約を追加または置換する文の例を示します。

```
ALTER TABLE Customers
ALTER Phone
CHECK ( Phone LIKE '___-___-____' );
```

- テーブルに定義された検査制約を変更できます。
  - ALTER TABLE と ADD `table-constraint` 句を使って新しい検査制約を追加できます。
  - 制約名を定義済みの場合は、制約を個別に変更できます。
  - 制約名を定義していない場合は、ALTER TABLE DELETE CHECK を使って、既存のすべての検査制約 (カラム検査制約、ドメインから継承した検査制約など) を削除してから、新しい検査制約を追加できます。
    ALTER TABLE 文で DELETE CHECK 句を使用するには、次のように指定します。

```
ALTER TABLE table-name
DELETE CHECK;
```

SQL Central では、テーブル検査制約とカラム検査制約の両方を追加、変更、削除できます。

テーブルからカラムを削除しても、そのカラムと関連付けられていた検査制約はテーブル制約から削除されません。制約を削除しないと、テーブルに対してデータの挿入や、単に問い合わせを行っただけでもエラーメッセージが生成されます。

> **i 注記**
>
> テーブル検査制約は FALSE が返された場合にエラーとなります。UNKNOWN が返されても、動作は TRUE が返される場合と同じで、その値が受け入れられます。

関連情報

制約の管理 [741 ページ]

1.9.7 ドメインを使用したデータの整合性の向上方法

ドメインとは、値の許容範囲を制限したりデフォルト値を設定したりできるユーザ定義データ型のことです。

ドメインを使うと既存のデータ型を拡張できます。通常、値の許容範囲の制限には検査制約が使われます。さらに、ドメインではデフォルト値を設定できます。値は NULL であってもそうでなくてもかまいません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

独自のドメインを定義することには次のような利点があります。

- 不適切な値が入力された場合の一般的なエラーを防ぎます。ドメインに設定した制約で、値を一定の範囲またはフォーマットに保持させたいすべてのカラムと変数に、ある範囲内またはフォーマットの値だけを保持させることができます。たとえば、あるデータ型によって、データベースに入力されるクレジットカード番号に確実に正しい桁数が入るようにできます。
- アプリケーションやデータベースの構造をわかりやすくします。
- 利便性。たとえば、テーブルの識別子をすべて正の整数にし、デフォルト値としてオートインクリメントにするとします。テーブルを新しく作成するたびにこうした制約を設定するよりも、新しいドメインを定義して識別子がそのドメイン型の値以外をとらないように設定する方が、作業が少なくて済みます。

このセクションの内容:

ドメインの作成 [744 ページ]
  ユーザ定義データ型を作成します。

カラムへのドメインの適用 [745 ページ]
  SQL Central を使用して、カラムでドメイン (ユーザ定義データ型) を使用するように変更します。

ドメインの削除 [746 ページ]
  SQL Central を使用して、ユーザ定義データ型 (ドメイン) を削除します。

1.9.7.1 ドメインの作成

ユーザ定義データ型を作成します。

前提条件

CREATE DATATYPE または CREATE ANY OBJECT システム権限が必要です。

コンテキスト

SQL Anywhere には定義済みドメインがいくつかあります (通貨データドメインの MONEY など)。

手順

1. SQL Central で、*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. 左ウィンドウ枠で ドメインを右クリックし、 新規 › ドメイン をクリックします。
3. ドメイン作成ウィザードの指示に従います。

結果

新しいドメインが作成されます。

1.9.7.2 カラムへのドメインの適用

SQL Central を使用して、カラムでドメイン (ユーザ定義データ型) を使用するように変更します。

前提条件

そのカラムが属してい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。

- 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

手順

1. SQL Central で、*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. 左ウィンドウ枠で、テーブルをダブルクリックします。
3. テーブルをクリックします。
4. 右ウィンドウ枠で、*[カラム]* タブをクリックします。
5. カラムを右クリックし、*[プロパティ]* をクリックします。
6. データ型タブをクリックし、ドメインをクリックします。
7. ドメインリストからドメインを選択します。
8. *OK* をクリックします。

結果

カラムは選択されたドメインを使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.7.3 ドメインの削除

SQL Central を使用して、ユーザ定義データ型 (ドメイン) を削除します。

前提条件

DROP DATATYPE または DROP ANY OBJECT システム権限が必要です。

データベース内の変数やカラムによって使用されているドメインは削除できません。そのドメインを使用しているカラムや変数をすべて削除または変更してから、ドメインを削除してください。

手順

1. SQL Central で、*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. 左ウィンドウ枠で、ドメインをダブルクリックします。
3. 右ウィンドウ枠で、ドメインを右クリックし、*[削除]* をクリックします。
4. はいをクリックします。

結果

ドメインは削除されます。

1.9.8 エンティティ整合性と参照整合性

データベースの関係構造によって、データベースサーバがデータベース内の情報を識別し、各テーブル内のすべてのローで (データベーススキーマに定義されている) テーブル間の関係が維持されていることを保証できます。

このセクションの内容:

エンティティ整合性 [747 ページ]
ユーザがローを挿入または更新すると、データベースサーバによってそのテーブルのプライマリキーの有効性が確実に保持されます。つまり、テーブル内の各ローがプライマリキーによりユニークに識別されます。

クライアントアプリケーションがエンティティ整合性に違反する場合 [748 ページ]
エンティティ整合性は、プライマリキーの値がテーブル内でユニークで、かつそこに NULL が含まれていないことが必要です。

プライマリキーによるエンティティ整合性の確保 [748 ページ]
各テーブルにプライマリキーが設定されれば、エンティティ整合性を確保するためにクライアントアプリケーション開発者やデータベース管理者がそれ以上の操作をする必要はありません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

[参照整合性 [748 ページ]]
外部キー関係を有効にするには、外部キーのエントリが参照先のテーブルのローのプライマリキー値に対応していなければなりません。

[外部キーによる参照整合性の確保 [751 ページ]]
CREATE TABLE 文または ALTER TABLE 文を使用して、外部キーを作成します。

[参照整合性の喪失 [751 ページ]]
特定の条件が存在する場合、データベースの参照整合性が失われます。

[クライアントアプリケーションが参照整合性に違反する場合 [752 ページ]]
クライアントアプリケーションがあるテーブルからプライマリキーの値を更新または削除したときに、データベース内にそのプライマリキーを参照する外部キーがある場合、参照整合性に違反する可能性があります。

[参照整合性アクション [752 ページ]]
他から参照されているプライマリキーの更新や削除に対して参照整合性を維持する最も単純な方法は、更新や削除を禁止することです。しかしそれ以外にも、参照整合性を保つために各外部キーを操作することもできます。

[参照整合性の検査 [753 ページ]]
参照性制約に違反する可能性があるとして RESTRICT 操作に定義されている外部キーについては、文の実行時にデフォルトで検査が行われます。

1.9.8.1 エンティティ整合性

ユーザがローを挿入または更新すると、データベースサーバによってそのテーブルのプライマリキーの有効性が確実に保持されます。つまり、テーブル内の各ローがプライマリキーによりユニークに識別されます。

例

例 1

SQL Anywhere サンプルデータベースの Employees テーブルでは、employee ID をプライマリキーとして使用します。新しい従業員がこのテーブルに追加されると、データベースサーバは新しい employee ID 値がユニークであり、NULL でないことを検査します。

例 2

SQL Anywhere サンプルデータベースの SalesOrderItems テーブルは、プライマリキーとして 2 つのカラムを使用します。

このテーブルは注文された製品の情報を格納します。ID カラムには注文番号が入っていますが、1 つの注文番号に対して複数の製品が注文される場合があるため、このカラムだけではプライマリキーになりません。一方、LineID カラムは製品に対応する行を識別します。ID カラムと LineID カラムの 2 つがセットになって、ある製品をユニークに指定できるので、この 2 つがプライマリキーになります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.8.2 クライアントアプリケーションがエンティティ整合性に違反する場合

エンティティ整合性は、プライマリキーの値がテーブル内でユニークで、かつそこに NULL が含まれていないことが必要です。

クライアントアプリケーションが重複するプライマリキー値を追加または更新すると、エンティティ整合性が破られます。エンティティ整合性の違反が検出されると、新しい情報はデータベースに追加されず、クライアントアプリケーションにエラーが返されます。

整合性の違反をどのようにしてユーザに通知し、どう処理させるかは、アプリケーション側の問題です。適切な処置といっても、ユーザに対して重複しない値をプライマリキーに指定するよう促すことしかできません。

1.9.8.3 プライマリキーによるエンティティ整合性の確保

各テーブルにプライマリキーが設定されれば、エンティティ整合性を確保するためにクライアントアプリケーション開発者やデータベース管理者がそれ以上の操作をする必要はありません。

テーブルの所有者は、テーブルの作成時にプライマリキーを指定します。後日、テーブル構造を修正した場合は、プライマリキーも再定義します。

関連情報

プライマリキー [25 ページ]

1.9.8.4 参照整合性

外部キー関係を有効にするには、外部キーのエントリが参照先のテーブルのローのプライマリキー値に対応していなければなりません。

場合によっては、プライマリキー以外のユニークなカラムが参照先になります。

外部キーは、通常別のテーブルにあるプライマリキーまたは一意性制約を参照します。そのプライマリキーが存在しない場合、問題の外部キーはオーファンと呼ばれます。SQL Anywhere は、データベースに参照整合性に違反するローがないかを自動的に確認します。このプロセスを参照整合性の検証と呼びます。データベースサーバは、オーファンの数をカウントすることで、参照整合性を検証します。

複数カラムの外部キーを使用する場合、MATCH 句を使用して、孤立したローの構成内容と参照整合性違反の構成内容を対比できます。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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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。

例

例 1

SQL Anywhere サンプルデータベースには、Employees テーブルと Departments テーブルが含まれています。Employee テーブルのプライマリキーは employee ID、Department テーブルのプライマリキーは department ID です。この Employee テーブルで、department ID は、Department テーブルに対する外部キーです。Employee テーブルの各 department ID は、Department テーブルの department ID と厳密に一致しています。

外部キー関係は、多対 1 の関係です。Employees テーブルの中には、同じ department ID エントリを含む複数のエントリがありますが、department ID は Departments テーブルのプライマリキーであるため、その中には 1 つしかありません。外部キーが、重複したエントリもしくは NULL 値を含む Departments テーブルのカラムを参照できると、Departments テーブルのどのローが正しい参照先かを決められなくなります。これは、外部キーカラムを NOT NULL として定義することによって回避されます。

例 2

データベースに、事業所の所在地をリストする office テーブルが含まれているとします。Employees テーブルには、その従業員が勤務する事業所の所在地を示すために、office テーブルに対する外部キーが指定されています。ここでデータベースの設計者は、従業員が雇用されたときに事業所の所在地を指定しないでおくことができます。これは、まだ配置先が決定していない場合や、外で働く場合に対応するためです。このような場合、外部キーはオプションで、NULL 値を使用できます。

例 3

次の文を実行して、複合プライマリキーを作成します。

```
CREATE TABLE pt(
     pk1 INT NOT NULL,
     pk2 INT NOT NULL,
     str VARCHAR(10)
     PRIMARY KEY ( pk1, pk2 ));
```

次の文は、プライマリキーとカラムの順序が異なる外部キーと、外部キーカラムに対して異なるソートの程度を作成します。これは、外部キーインデックスを作成する場合に使用されます。

```
CREATE TABLE ft1(
     fpk INT PRIMARY KEY,
     ref1 INT,
     ref2 INT );
```

```
ALTER TABLE ft1 ADD FOREIGN KEY ( ref2 ASC, ref1 DESC)
     REFERENCES pt ( pk2, pk1 ) MATCH SIMPLE;
```

次の文を実行して、プライマリキーとカラムの順序が同じだが、外部キーインデックスに対してソートの程度が異なる外部キーを作成します。また、この例では、MATCH FULL 句を使用して、両方のカラムが NULL だった場合に孤立した

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ローになることを指定します。UNIQUE 句は、NULL ではないカラムの pt テーブルと ft2 テーブルの間に 1 対 1 の関係を適用します。

```
CREATE TABLE ft2(
     fpk INT PRIMARY KEY,
     ref1 INT,
     ref2 INT );
```

```
ALTER TABLE ft2 ADD FOREIGN KEY ( ref1, ref2 DESC )
     REFERENCES pt ( pk1, pk2 ) MATCH UNIQUE FULL;
```

このセクションの内容:

参照循環性 [750 ページ]
参照循環性とは、あるデータベースオブジェクトとそのデータベースオブジェクト自体、または、他のデータベースオブジェクトとの関係を指します。

1.9.8.4.1 参照循環性

参照循環性とは、あるデータベースオブジェクトとそのデータベースオブジェクト自体、または、他のデータベースオブジェクトとの関係を指します。

たとえば、テーブルに、それ自体を参照する外部キーを含めることができます。これを自己参照テーブルと呼びます。自己参照テーブルは、参照循環性の特殊な場合です。

例

SQL Anywhere サンプルデータベースには、従業員 (employee) 情報が入ったテーブルが 1 つと、部署 (department) 情報の入ったテーブルが 1 つあります。

```
CREATE TABLE "GROUPO"."Employees" (
    "EmployeeID"                    int NOT NULL
   ,"ManagerID"                     int NULL
   ,"Surname"                       "person_name_t" NOT NULL
   ,"GivenName"                     "person_name_t" NOT NULL
   ,"DepartmentID"                  int NOT NULL
   ,"Street"                        "street_t" NOT NULL
   ,"City"                          "city_t" NOT NULL
   ,"State"                         "state_t" NULL
   ,"Country"                       "country_t" NULL
   ,"PostalCode"                    "postal_code_t" NULL
   ,"Phone"                         "phone_number_t" NULL
   ,"Status"                        char(2) NULL
   ,"SocialSecurityNumber"          char(11) NOT NULL
   ,"Salary"                        numeric(20,3) NOT NULL
   ,"StartDate"                     date NOT NULL
   ,"TerminationDate"               date NULL
   ,"BirthDate"                     date NULL
   ,"BenefitHealthInsurance"        bit NULL
   ,"BenefitLifeInsurance"          bit NULL
   ,"BenefitDayCare"                bit NULL
   ,"Sex"                           char(2) NULL CONSTRAINT "Sexes" check(Sex
in( 'F','M','NA' ) )
   ,CONSTRAINT "EmployeesKey" PRIMARY KEY ("EmployeeID")
)
ALTER TABLE "GROUPO"."Employees"
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    ADD CONSTRAINT "SSN" UNIQUE ( "SocialSecurityNumber" )
CREATE TABLE "GROUPO"."Departments" (
    "DepartmentID"                    int NOT NULL
   ,"DepartmentName"                  char(40) NOT NULL
   ,"DepartmentHeadID"                int NULL
   ,CONSTRAINT "DepartmentsKey" PRIMARY KEY ("DepartmentID")
   ,CONSTRAINT "DepartmentRange" check(DepartmentID > 0 and DepartmentID <= 999)
)
```

Employees テーブルには、プライマリキー "EmployeeID" と候補キー "SocialSecurityNumber" があります。Departments テーブルには、プライマリキー"DepartmentID" があります。Employees テーブルは、外部キーの定義によって Departments テーブルに関連付けられます。

```
ALTER TABLE "GROUPO"."Employees"
    ADD NOT NULL FOREIGN KEY "FK_DepartmentID_DepartmentID" ("DepartmentID")
    REFERENCES "GROUPO"."Departments" ("DepartmentID")
```

特定の従業員の所属部署名を探せるように、その従業員の部署名を Employees テーブルに格納しておく必要はありません。その代わり Employees テーブルには、Departments テーブルの DepartmentID 値の 1 つと一致する部署番号の入った "DepartmentID" カラムがあります。

Employees テーブルは、Employees と Departments 間の多対 1 の関係を宣言する上記の参照整合性制約によって、Departments テーブルを参照します。さらに、Employees テーブルの外部キーカラムである DepartmentID が NOT NULL と宣言されているため、これは必須の関係です。しかし、これは Employees テーブルと Departments テーブル間の唯一の関係ではありません。Departments テーブル自体に、各部署の長を示すための Employees テーブルへの外部キーがあります。

```
ALTER TABLE "GROUPO"."Departments"
    ADD FOREIGN KEY "FK_DepartmentHeadID_EmployeeID" ("DepartmentHeadID")
    REFERENCES "GROUPO"."Employees" ("EmployeeID")
    ON DELETE SET NULL
```

これは、Departments テーブルと Employees テーブル間のオプションの多対 1 の関係です。参照整合性制約だけでは複数の部署の長が同じになることを回避できないため、多対 1 になっています。その結果、Employees テーブルと Departments テーブルは参照循環性を形成し、それぞれに他方への外部キーがあります。

1.9.8.5 外部キーによる参照整合性の確保

CREATE TABLE 文または ALTER TABLE 文を使用して、外部キーを作成します。

外部キーを作成すると、外部キーに指定したカラムに入力できる値が、関連付けたテーブルのプライマリキー値として存在する値に限られます。

1.9.8.6 参照整合性の喪失

特定の条件が存在する場合、データベースの参照整合性が失われます。

- プライマリキーの値が更新または削除された場合。そのプライマリキーを参照している外部キーがすべて無効になります。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 外部テーブルに新しく追加されたローに、プライマリキーと対応しない外部キーの値が入力された場合。データベースが無効になります。

SQL Anywhere には、参照整合性が失われるこれら 2 つの事態の両方に対する保護機能が備わっています。

1.9.8.7 クライアントアプリケーションが参照整合性に違反する場合

クライアントアプリケーションがあるテーブルからプライマリキーの値を更新または削除したときに、データベース内にそのプライマリキーを参照する外部キーがある場合、参照整合性に違反する可能性があります。

例

サーバがプライマリキーの更新または削除を許可して、それを参照する外部キーに何も変更を加えないと、外部キーの参照は無効になります。KEY JOIN 句を使用した SELECT 文など、外部キーの参照を使う処理は、参照先のテーブルに対応する値がないためエラーになります。

データベースサーバは、エンティティ整合性に違反していると、単にデータ入力を拒否してエラーメッセージを表示するだけですが、参照整合性の違反への対応は複雑になる場合があります。参照整合性を維持するために、参照整合性アクションとして知られている手段はいくつかあります。

1.9.8.8 参照整合性アクション

他から参照されているプライマリキーの更新や削除に対して参照整合性を維持する最も単純な方法は、更新や削除を禁止することです。しかしそれ以外にも、参照整合性を保つために各外部キーを操作することもできます。

データベース管理者やテーブル所有者は、CREATE TABLE 文と ALTER TABLE 文を使って、変更されたプライマリキーを参照している外部キーに対し、整合性の違反が発生したときに実行するアクションを指定できます。

注記

参照整合性アクションは、ユニークな値に対する論理的な更新ではなく物理的な更新によってトリガされます。たとえば、大文字と小文字を区別しないデータベースでも、プライマリキーの値を `SAMPLE-VALUE` から `sample-value` に更新すると、2 つの値は論理的には同じであっても参照整合性アクションがトリガされます。

次の参照整合性アクションは、プライマリキーの更新と削除に対してそれぞれ別に指定できます。

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。

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。

CASCADE

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このアクションを ON UPDATE と併用した場合、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。このアクションを ON DELETE と併用した場合、削除されたプライマリキーを参照している外部キーがあるすべてのローが削除されます。

システムトリガによって参照整合性アクションが実装されます。トリガはプライマリテーブル上で定義され、セカンダリテーブルの所有者の権限を使って実行されます。つまり、特に権限が与えられていなくても、所有者の違うテーブルに対するカスケード処理ができることになります。

1.9.8.9　参照整合性の検査

参照性制約に違反する可能性があるとして RESTRICT 操作に定義されている外部キーについては、文の実行時にデフォルトで検査が行われます。

CHECK ON COMMIT 句を指定した場合、検査が行われるのはトランザクションのコミット時にかぎられます。

検査のタイミングを制御するデータベースオプション

参照整合性に違反する操作を制限するよう定義されている外部キーは、wait_for_commit データベースオプションを使ってその動作を制御できます。このオプションは、CHECK ON COMMIT 句によって無効になります。

デフォルトでは wait_for_commit が Off に設定されており、データベースに整合性違反を生じさせる可能性のある操作は実行できません。たとえば、従業員がまだ存在している部署に対する DELETE は実行できません。次の文を実行すると、エラーになります。

```
DELETE FROM Departments
WHERE DepartmentID = 200;
```

wait_for_commit を On にすると、実際にコミットが実行されるまで参照整合性は検査されません。データベースの整合性が失われていると、データベースはコミットの実行を許可せずにエラーを返します。このモードの場合、データベースユーザはまだ従業員がいる部署を削除できますが、以下の操作が終了しないかぎり変更をコミットできません。

- その部署に所属する従業員を削除するか、もしくは他の部署に移します。
- DepartmentID ローを Departments テーブルに戻します。
- トランザクションをロールバックして、DELETE 操作を取り消します。

このセクションの内容:

INSERT 文の整合性検査 [754 ページ]
データベースサーバは、INSERT 文の実行時に整合性検査を実行します。

DELETE 文または UPDATE 文の整合性検査 [754 ページ]
更新オペレーションまたは削除オペレーションを行う場合に、外部キーエラーが発生する可能性があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.9.8.9.1 INSERT 文の整合性検査

データベースサーバは、INSERT 文の実行時に整合性検査を実行します。

たとえば、部署を新設しようとする場合に、すでに使用されている DepartmentID 値を指定すると仮定します。

```
INSERT
INTO Departments ( DepartmentID, DepartmentName, DepartmentHeadID )
VALUES ( 200, 'Eastern Sales', 902 );
```

テーブルのプライマリキーがユニークでなくなるため、INSERT は拒否されます。DepartmentID カラムはプライマリキーなので、重複する値は許可されません。

関係に違反する値の挿入

次の文は SalesOrders テーブルに新しいローを挿入しますが、Employees テーブル内には存在しない SalesRepresentative ID を誤って指定します。

```
INSERT
INTO SalesOrders ( ID, CustomerID, OrderDate, SalesRepresentative)
VALUES ( 2700, 186, '2000-10-19', 284 );
```

Employees テーブルと SalesOrders テーブルとの関係は、SalesOrders テーブルの SalesRepresentative カラムと Employees テーブルの EmployeeID カラムに基づいた 1 対多の関係です。プライマリテーブル (Employees) にレコードが入力されている場合にのみ、外部テーブル (SalesOrders) 内の対応するレコードを挿入できます。

外部キー

Employees テーブルのプライマリキーは従業員 ID 番号です。SalesRepresentative テーブル内の営業担当者 ID 番号は、Employees テーブルの外部キーです。つまり、SalesOrders テーブル内にある各営業担当者 ID 番号と、Employees テーブル内にある同じ従業員の従業員 ID 番号を一致させる必要があります。

営業担当者 284 に対して注文を追加しようとすると、エラーが発生します。

Employees テーブルには、その ID 番号を持つ従業員は存在しません。これによって、有効な営業担当者 ID のない注文が挿入されるのを防ぐことができます。

1.9.8.9.2 DELETE 文または UPDATE 文の整合性検査

更新オペレーションまたは削除オペレーションを行う場合に、外部キーエラーが発生する可能性があります。

たとえば、Departments テーブルから R&D 部を削除するとします。DepartmentID フィールドは Departments テーブルのプライマリキーなので、1 対多の関係の "1" の側を構成します (対応する外部キーは Employees テーブルの DepartmentID フィールドで、"多" の側を構成します)。1 対多の関係の "1" のレコードは、対応する "多" のレコードがすべて削除されるまで削除できません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

DELETE 文の参照整合性エラー

Departments テーブルの R&D 部 (DepartmentID は 100) を削除するとします。R&D 部を参照するその他のレコードがデータベース内にあることを示すエラーがレポートされて、削除オペレーションが実行されません。R&D 部を削除するには、次のように、その部署に所属しているすべての従業員を削除する必要があります。

```
DELETE
FROM Employees
WHERE DepartmentID = 100;
```

これで、R&D 部に所属するすべての従業員が削除されたため、R&D 部を削除することができます。

```
DELETE
FROM Departments
WHERE DepartmentID = 100;
```

次の ROLLBACK 文を入力して、データベースのこれらの変更をキャンセルしてください。

```
ROLLBACK;
```

UPDATE 文の参照整合性エラー

たとえば、Employees テーブルの DepartmentID 項目を変更するとします。DepartmentID フィールドは Employees テーブルの外部キーなので、1 対多の関係の "多" の側を構成します (対応するプライマリキーは Departments テーブルの DepartmentID フィールドで、"1" の側を構成します)。1 対多の関係の "多" の側のレコードは、"1" の側のレコードに対応しないかぎり、つまり参照するプライマリキーがなければ変更できません。

たとえば次の UPDATE 文を実行すると、整合性エラーが発生します。

```
UPDATE Employees
SET DepartmentID = 600
WHERE DepartmentID = 100;
```

DepartmentID が 600 の部署は Departments テーブルに存在しないため、エラーが発生します。

Employees テーブルの DepartmentID フィールドの値を変更するには、Departments テーブルの既存の値に対応する必要があります。次に例を示します。

```
UPDATE Employees
SET DepartmentID = 300
WHERE DepartmentID = 100;
```

この文は、DepartmentID 300 は既存の Finance 部に対応するので実行できます。

次の ROLLBACK 文を入力して、データベースのこれらの変更をキャンセルしてください。

```
ROLLBACK;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コミット時の整合性の検査

前述の例では、それぞれの文の実行時にデータベースの整合性が検査されています。データベースの整合性を失わせる可能性があるオペレーションは実行されません。

wait_for_commit オプションを使用すると、コミット時まで整合性が検査されないようにデータベースを設定することができます。これは、変更を加えている間にデータの一貫性が一時的に失われる可能性があるような変更の必要がある場合に便利です。たとえば、Employees テーブルと Departments テーブルの R&D 部を削除するとします。これらのテーブルには相互参照があり、かつ削除は一度に 1 テーブルずつ実行する必要があるため、削除中はテーブル間で一貫性が失われます。この場合、データベースでは削除が完了するまでコミットを実行できません。wait_for_commit オプションを On に設定して、コミットが実行されるまでデータの一貫性が失われることを許容してください。

また、プライマリキーに加えた変更に合わせて、外部キーが自動的に修正されるように定義することもできます。上の例では、Employees テーブルから Departments テーブルへの外部キーが ON DELETE CASCADE を使用して定義された場合、部署 ID を削除すると、Employees テーブルの対応するエントリが自動的に削除されます。

ここまでの例では、整合性のないデータベースが確定的なものとしてコミットされることはありません。変更することによってデータベースの整合性が失われる可能性がある場合、SQL Anywhere では代替の動作もサポートされています。

関連情報

データ整合性 [727 ページ]

1.9.9 システムテーブルの整合性ルール

データベースの整合性検査と規則に関する情報は、すべてカタログに格納されています。

システムビュー	説明
SYS.SYSCONSTRAINT	SYS.SYSCONSTRAINT システムビュー内の各ローには、データベース内の制約が記述されています。現在サポートされている制約には、テーブルとカラムの検査、プライマリキー、外部キー、一意性制約が含まれます。 テーブルとカラムの検査制約の場合、実際の検査条件は SYS.ISYSCHECK システムテーブルに含まれています。
SYS.SYSCHECK	SYS.SYSCHECK システムビューの各ローは、SYS.SYSCONSTRAINT システムビューにリストされている検査制約を定義します。
SYS.SYSFKEY	SYS.SYSFKEY システムビューの各ローは、キーに定義した一致タイプなど、外部キーに関して記述します。
SYS.SYSIDX	SYS.SYSIDX システムビューの各ローは、データベース内のインデックスを定義します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

システムビュー	説明
SYS.SYSTRIGGER	SYS.SYSTRIGGER システムビューの各ローは、参照トリガアクション (ON DELETE CASCADE など) が設定された外部キー制約を自動的に作成するトリガなどの、データベース内のトリガ 1 つを示します。 referential_action カラムには、アクションがカスケード (C)、削除 (D)、NULL 設定 (N)、制限 (R) のいずれであるかを示すアルファベット 1 文字が格納されています。 event カラムには、各アクションを実行するイベントを示すアルファベット 1 文字が格納されます (A = 挿入と削除、B = 挿入と更新、C = 更新、D = 削除、E = 削除と更新、I = 挿入、U = 更新、M = 挿入、削除、更新)。 trigger_time カラムには、アクションの実行がイベントのトリガの後 (A) または前 (B) のどちらに行われるかが表示されます。

1.10 トランザクションと独立性レベル

トランザクションと独立性レベルは、一貫性を通じてデータ整合性の確保に役立ちます。

一貫性の概念は例によって的確に理解できます。

一貫性の例

預金口座を扱うデータベースを使っていて、ある顧客の口座から別の口座に送金したいとします。送金前と送金後では、データベースの状態は一貫しています。しかし、1 つの口座から資金を引き落とした後、次の口座に振り込むまでのデータベースの状態は一貫していません。送金処理において、顧客の口座残高が送金前と同じ段階では、データベースの状態は一貫性があります。ある程度送金されると、データベースの状態は一貫性がなくなります。引き落としと振り込みの両方を処理するか、またはどちらも処理しないようにする必要があります。

トランザクションは作業の論理単位

トランザクションは、作業の論理単位です。各トランザクションは、1 つのタスクを実行し、データベースをある一貫した状態から別の状態に変換する、論理的に関連した一連の文です。一貫した状態の性質は、使用しているデータベースに依存します。

各トランザクション内の文は不可分の単位として処理されます。すべての文が実行されるか、またはどの文も実行されません。各トランザクションの最後に、変更を「コミット」して確定します。何らかの理由でトランザクション内の文のいくつかが正しく処理されない場合は、途中の変更が取り消されるか、またはロールバックされます。この特性を、トランザクションが「アトミック」であるといいます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

複数の文をトランザクションにグループ分けすることは、媒体障害時またはシステム障害時にデータの一貫性を保護するため、およびデータベースの同時操作を管理するために重要です。トランザクションは安全にインターリーブされ、各トランザクションが完了すると、データベース内の情報が一貫しているポイントにマークが付けられます。データベースを 1 つの一貫した状態から別の状態へ変更するタスクを実行するために各トランザクションを設計します。

通常の操作においてシステム障害やデータベースクラッシュが発生した場合、データベースサーバはそのデータベースが次に起動されたときにデータを自動的にリカバリします。自動リカバリ処理では、完了済みのすべてのトランザクションを復元し、障害発生時にコミットされていなかったトランザクションをロールバックします。トランザクションのアトミックな性質により、データベースは確実に一貫した状態にリカバリされます。

このセクションの内容:

トランザクション [758 ページ]
トランザクションをコミットして、データベースの変更を永続的なものにします。

同時実行性 [761 ページ]
同時実行性とは複数のトランザクションを同時に処理するためにデータベースサーバで必要な機能です。

トランザクション内のセーブポイント [762 ページ]
関連する文のグループを区切るために、トランザクション内でセーブポイントを定義できます。

独立性レベルと一貫性 [763 ページ]
独立性レベルを設定すると、あるトランザクションの操作が、同時に処理されている別のトランザクションの操作からどの程度参照できるかを制御できます。

トランザクションのブロックとデッドロック [779 ページ]
トランザクションの実行中、データベースサーバはローにロックをかけ、処理中のローが他のトランザクションからの干渉を受けないようにします。

ロックの仕組み [782 ページ]
ロックは、複数のトランザクションを同時に実行しているときにデータの整合性を保護する同時制御メカニズムです。

独立性レベル選択のガイドライン [802 ページ]
独立性レベルの選択は、アプリケーションが実行するタスクの種類によって異なります。

独立性レベルに関するチュートリアル [807 ページ]
それぞれの独立性レベルは異なる動作を実行し、データベースとそこで実行する操作に応じて、使用するレベルを決定する必要があります。

ユニークな値を生成するためのシーケンスの使用 [830 ページ]
シーケンスを使用すると、複数のテーブル間でユニークな値、つまり一連の自然数と異なる値を生成できます。

1.10.1 トランザクション

トランザクションをコミットして、データベースの変更を永続的なものにします。

データを変更すると、変更がトランザクションログに記録されます。変更は、COMMIT 文を実行するまでは永続的なものにはなりません。

トランザクションは次のいずれかのイベントで開始します。

- データベースへの接続後、最初の文
- トランザクションの終了後、最初の文

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

トランザクションは次のいずれかのイベントで完了します。

- データベースへの変更を確定する COMMIT 文
- トランザクションで行われたすべての変更を取り消す ROLLBACK 文
- 関連動作としてオートコミットが実行される文。ALTER、CREATE、COMMENT、DROP などのほとんどのデータ定義文では、関連動作としてオートコミットが実行されます。
- データベースへの接続を切断することにより、暗黙的なロールバックの原因となります。
- ODBC と JDBC には各文の後で COMMIT 文を実行するオートコミットの設定があります。デフォルトでは、ODBC と JDBC のオートコミットの設定は ON にする必要があり、各文は単一のトランザクションとして処理されます。トランザクション設計機能を活用する場合は、オートコミットの設定を OFF にします。
- chained データベースオプションを Off にしておくと、各文の後にオートコミットを実行したのと同様の処理が行われます。デフォルトでは、jConnect または Open Client アプリケーションを使用する接続では chained は Off に設定されています。

SQL Anywhere コックピット を使用してデータベースに接続することで、未処理のトランザクションがある接続を確定します。接続ページを検査して、コミットされていない操作がある接続を確認します。

トランザクションの開始タイミングの決定

TransactionStartTime 接続プロパティは、COMMIT または ROLLBACK の後にデータベースサーバがデータベースを最初に修正した時間を返します。このプロパティを使用して、すべてのアクティブな接続で最も早いトランザクションの開始時刻を調べます。

次の例では、TransactionStartTime 接続プロパティを使用して、データベースに対するすべての接続で最も早いトランザクションの開始時刻を特定します。このプロパティは、現在のデータベースのすべての接続をループし、データベースに対する最も早いトランザクションのタイムスタンプを文字列として返します。この情報は、トランザクションがローやテーブルを取得するときに有益です。その他のトランザクションは、ブロックオプションに応じてテーブルとローロックをブロックできます。トランザクションの実行に時間がかかると、他のユーザがブロックされたり、パフォーマンスに影響を与える原因となる可能性があります。次に例を示します。

```
BEGIN
  DECLARE connid int;
  DECLARE earliest char(50);
  DECLARE connstart char(50);
  SET connid=next_connection(null);
  SET earliest = NULL;
  lp: LOOP
  IF connid IS NULL THEN LEAVE lp END IF;
    SET connstart = CONNECTION_PROPERTY('TransactionStartTime',connid);
    IF connstart <> '' THEN
      IF earliest IS NULL
      OR CAST(connstart AS TIMESTAMP) < CAST(earliest AS TIMESTAMP) THEN
        SET earliest = connstart;
      END IF;
    END IF;
    SET connid=next_connection(connid);
  END LOOP;
  SELECT earliest
END
```

このセクションの内容:

トランザクションの終了方法および終了のタイミングの制御 (SQL の場合) [760 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

Interactive SQL には、いつ、どのようにトランザクションを終了するのかを指定する 2 つのオプションがあります。

関連情報

Transact-SQL との互換性 [523 ページ]

1.10.1.1 トランザクションの終了方法および終了のタイミングの制御 (SQL の場合)

Interactive SQL には、いつ、どのようにトランザクションを終了するのかを指定する 2 つのオプションがあります。

コンテキスト

デフォルトでは、ODBC はオートコミットモードで動作します。Interactive SQL で auto_commit オプションを OFF に設定しても、ODBC データソースの ODBC 設定によって Interactive SQL の設定が上書きされます。ODBC の設定は、SQL_ATTR_AUTOCOMMIT 接続属性を使用して変更します。ODBC オートコミットは chained オプションとは無関係です。

手順

トランザクションをいつ、どのように終了するのかを管理するには、次のオプションから 1 つ選択します。

オプション	アクション
auto_commit オプションの使用	文が正常に実行された場合はすべて結果が自動的にコミットされ、失敗した場合は ROLLBACK が自動的に実行されます。次の文を実行します。 `SET OPTION auto_commit=ON;`
comiit_on_exit オプションの使用	Interactive SQL の終了時に、コミットされていない変更をどうするかを決定します。このオプションが ON (デフォルト) に設定されている場合、Interactive SQL は COMMIT を実行し、それ以外の場合は、コミットされていない変更を ROLLBACK 文で取り消します。次の文を実行します。 `SET OPTION commit_on_exit={ ON | OFF }`

結果

トランザクション終了のタイミングと方法について、Interactive SQL でどのように決定するか設定しました。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.2 同時実行性

同時実行性とは複数のトランザクションを同時に処理するためにデータベースサーバで必要な機能です。

データベースサーバ内に特殊なメカニズムがない場合、同時に発生したトランザクションがお互いに干渉して、情報の一貫性と正確性が失われる可能性があります。

同時実行性に関する知識が必要なユーザ

同時実行性は、すべてのデータベース管理者と開発者に関係のある問題です。シングルユーザデータベースで作業する場合でも、複数のアプリケーションからの要求や、単一のアプリケーションの複数の接続からの要求を処理する場合には、同時性を考慮してください。これらの複数のアプリケーションや接続は、ネットワーク上でマルチユーザが経験するのとまったく同様に、お互いに干渉し合います。

同時実行性に影響するトランザクションサイズ

SQL 文をトランザクションにまとめる方法は、データの整合性とシステムのパフォーマンスに大きな影響を与えます。トランザクションが短かすぎて論理的にひとまとまりにならない場合、データベースの一貫性が失われる可能性があります。トランザクションが長すぎて複数の互いに関係のない操作が含まれていると、本来なら安全にデータベースにコミットできたはずの操作が、不要な ROLLBACK によって取り消される可能性が高くなります。

トランザクションが長い場合、他のトランザクションの同時処理が防止されるため、同時実行性が低下します。

アプリケーションのタイプや環境要因に応じて、トランザクションの適切な長さを決定する要素は数多くあります。

このセクションの内容:

プライマリキーの生成と同時実行性 [761 ページ]
データベースは、プライマリキーと呼ばれるユニークな番号を自動的に生成できます。

1.10.2.1 プライマリキーの生成と同時実行性

データベースは、プライマリキーと呼ばれるユニークな番号を自動的に生成できます。

たとえば、商品の送り状を格納するテーブルを作成する場合、販売スタッフではなくデータベースが自動的にユニークな送り状番号を割り当てることができます。

例

たとえば、商品の送り状番号は、1 つ前の送り状番号に 1 を加えて作成できます。しかし、複数の人間がデータベースに送り状番号を入力するときは、この方法は使えません。2 人の従業員が同じ番号を選択する可能性があるからです。

この問題を解決するには、次に示すように方法がいくつかあります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- 送り状番号を入力するユーザごとに数字の範囲を設定します。
  このスキームは、カラム user name と invoice number を持つテーブルを作成することで実装します。ローの 1 つは、送り状番号を入力するユーザを記録するのに使います。ユーザが送り状を追加するごとに、テーブル内の番号は 1 ずつ増えて新しい送り状に使われます。データベースのすべてのテーブルを処理するには、テーブルに 3 つのカラム (テーブル名、ユーザ名、最後のキー値) が必要です。各ユーザに十分な数字が確保されているかどうかを定期的に確認する必要があります。
- 2 つのカラム table name と last key を持つテーブルを作成します。
  このテーブルには、最後に使用した送り状番号が格納されるローが 1 つあります。送り状番号は、ユーザが送り状の追加、新しい接続の確立、送り状番号の増分、または変更の即時コミットを行うたびに、自動的に増分されます。ローは別のトランザクションによって直ちに更新されるため、他のユーザは新しい送り状番号にアクセスできます。
- NEWID のデフォルト値のあるカラムを UNIQUEIDENTIFIER バイナリデータ型と組み合わせて使用して、完全にユニークな識別子を生成します。
  UUID/GUID 値を使用すると、テーブルのローをユニークに識別できます。あるコンピュータで生成された値は別のコンピュータで生成された値と一致しないため、レプリケーションと同期の環境では、UUID/GUID 値をキーとして使用できます。
- AUTOINCREMENT のデフォルト値を持つカラムを使用します。次に例を示します。

  ```
  CREATE TABLE Orders (
     OrderID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
     OrderDate DATE,
     primary key( OrderID )
  );
  ```

  テーブルに挿入するときに、AUTOINCREMENT カラムに対して値を指定しないと、ユニークな値が生成されます。値を指定すると、その値が使われます。値がカラムの現在の最大値より大きい場合は、その後の挿入開始ポイントとしてこの値が使われます。AUTOINCREMENT カラムに最後に追加されたローの値は、グローバル変数 @@identity で取得できます。

関連情報

NEWID デフォルト [737 ページ]

1.10.3 トランザクション内のセーブポイント

関連する文のグループを区切るために、トランザクション内でセーブポイントを定義できます。

SAVEPOINT 文は、トランザクション内の中間ポイントを定義します。そのポイント以降のすべての変更は ROLLBACK TO SAVEPOINT 文を使って取り消しできます。RELEASE SAVEPOINT 文の実行後、またはトランザクションの終了後は、セーブポイントは使えなくなります。セーブポイントは、COMMIT には影響しません。COMMIT が実行されると、トランザクション中に加えられたすべての変更がデータベースの中で永続的なものになります。

RELEASE SAVEPOINT 文または ROLLBACK TO SAVEPOINT 文では、ロックは解放されません。ロックはトランザクションの最後でのみ解放されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

セーブポイントの命名とネスト

名前を付けてネストしたセーブポイントを使って、トランザクション内にアクティブなセーブポイントを多数設定できます。SAVEPOINT と RELEASE SAVEPOINT の間の更新も、その前のセーブポイントにロールバックするか、トランザクション全体をロールバックすればキャンセルできます。トランザクション内の変更は、トランザクションがコミットされるまで確定していません。トランザクションが終了すると、セーブポイントはすべて解放されます。

セーブポイントはバルクオペレーションモードでは使用できません。セーブポイントを使用しても、オーバヘッドはほとんど増えません。

1.10.4 独立性レベルと一貫性

独立性レベルを設定すると、あるトランザクションの操作が、同時に処理されている別のトランザクションの操作からどの程度参照できるかを制御できます。

これには isolation_level データベースオプションを使用します。クエリ内の個々のテーブルの独立性レベルは、対応するテーブルヒントによって制御されます。

次の独立性レベルがあります。

独立性レベル	特性
0 - (コミットされていない読み出し)	• 書き込みロックの有無にかかわらず、ローで読み込みが許可されます。 • 読み込みロックは適用されません。 • 同時トランザクションがローを変更しないこと、またはローに対しての変更がロールバックされないことは保証されません。 • テーブルヒント NOLOCK と READUNCOMMITTED に対応します。 • ダーティリード、繰り返し不可能読み出し、幻ローを許可します。
1 - コミットされた読み出し	• 書き込みロックのないローでは読み込みのみ許可されます。 • 現在のローでの読み込みに対してのみ読み込みロックが取得されて保持されるが、カーソルがローから移動すると解放されます。 • トランザクション中にデータが変更されないという保証はありません。 • テーブルヒント READCOMMITTED に対応します。 • ダーティリードを防ぎます。 • 繰り返し不可能読み出しと幻ローを許可します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

独立性レベル	特性
2 - 繰り返し可能読み出し	● 書き込みロックのないローでは読み込みのみ許可されます。 ● 結果セットの各ローとして取得された読み込みロックが読み込まれ、トランザクションが終了するまで保持されます。 ● テーブルヒント REPEATABLEREAD に対応します。 ● ダーティリードと繰り返し不可能読み出しを防ぎます。 ● 幻ローを許可します。
3 - 直列化可能	● 書き込みロックのない結果内のローに対しては読み込みのみ許可されます。 ● カーソルが開いているときに取得された読み込みロックは、トランザクションの終了時まで保持されます。 ● テーブルヒント HOLDLOCK と SERIALIZABLE に対応します。 ● ダーティリード、繰り返し不可能読み出し、幻ローを防ぎます。
snapshot [1]	● 読み込みロックは適用されません。 ● すべてのローで読み込みが許可されます。 ● コミットされたデータのデータベーススナップショットは、トランザクションが最初のローの読み込みまたは更新を行った時点で作成されます。
statement-snapshot[1]	● 読み込みロックは適用されません。 ● すべてのローで読み込みが許可されます。 ● コミットされたデータのデータベーススナップショットは、文が最初のローの読み込みを行った時点で作成されます。
readonly-statement-snapshot[1]	● 読み込みロックは適用されません。 ● すべてのローで読み込みが許可されません。 ● コミットされたデータのデータベーススナップショットは、読み込み専用の文が最初のローの読み込みを行った時点で作成されます。 ● 更新可能な文の updatable_statement_isolation オプションで指定された独立性レベル (0、1、2、または 3) を使用します。

[1] データベースで allow_snapshot_isolation が On に設定され、そのデータベースでスナップショットアイソレーションが有効である必要があります。

デフォルトの独立性レベルは 0 です。ただし、Open Client、jConnect、TDS の各接続におけるデフォルトの独立性レベルは 1 です。

ロックベースの独立性レベルは、一部またはすべての干渉を防ぎます。レベル 3 は最も高いレベルの独立性を提供します。2 以下のレベルでは、一貫性のレベルは低くなりますが、パフォーマンスは一般にレベル 3 より高くなります。レベル 0 (コミットされない読み込み) がデフォルト設定です。

スナップショットアイソレーションのレベルは、読み込みと書き込み間の干渉を防ぎます。書き込みは相互に干渉する可能性があります。一貫性のない動作が多少生じる可能性がありますが、競合関連のパフォーマンスは独立性レベルを 0 に設定した場合と同じです。ローのバージョンを保存して使用する必要があるため、競合に関連しないパフォーマンスは悪くなります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

一般的に、各独立性レベルは、必要なロックの種類や、他のトランザクションが保持するロックをどのように扱うかによって特徴づけられます。独立性レベルが 0 の場合、データベースサーバは書き込みロックだけを必要とします。データベースサーバは、これらのロックを使用して、2 つのトランザクションが競合する修正を行わないようにします。たとえば、レベル 0 のトランザクションは、ローの更新や削除をする前に書き込みロックをかけ、すでに書き込みロックがかかっている新しいローを挿入します。

レベル 0 のトランザクションは、読み込み中のローはチェックしません。たとえば、レベル 0 のトランザクションがローを読み込むときは、他のトランザクションがそのローにどのようなロックをかけているかをチェックしません。チェックが不要のため、レベル 0 のトランザクション処理は速くなります。この速度は一貫性を犠牲にして得られたものです。別のトランザクションが書き込みロックをかけているローを読むと、ダーティデータを返す危険性があります。レベル 1 では、トランザクションはローを読む前に書き込みロックがかかっていないかをチェックします。操作は 1 つ増えますが、このトランザクションでは読み込むデータはすべてコミット済みであることが保証されます。

i 注記

どの独立性レベルにおいても、各トランザクションが完全に実行されるかまったく実行されないこと、および更新内容が失われないことが保証されます。

独立性レベルはトランザクション間にのみ当てはまります。同じトランザクション内の複数のカーソルが相互に干渉することはありません。

このセクションの内容:

スナップショットアイソレーション [766 ページ]
スナップショットアイソレーションは、さまざまなバージョンのデータを管理することにより同時実行性と一貫性を向上させる目的で設計されています。

典型的な矛盾のケース [772 ページ]
トランザクションの同時実行中に発生する可能性のある典型的な矛盾には 3 つのタイプがあります。

独立性レベルの設定方法 [774 ページ]
データベースへの各接続は独自の独立性レベルを持ちます。

ODBC 実行可能アプリケーションでの独立性レベル [776 ページ]
ODBC アプリケーションは、SQLSetConnectAttr を呼び出すときに、Attribute を SQL_ATTR_TXN_ISOLATION に、ValuePtr を対応する独立性レベルに設定します。

独立性レベルの参照 [778 ページ]
CONNECTION_PROPERTY 関数を使用して、現在の接続の独立性レベルを表示します。

関連情報

スナップショットアイソレーションを有効化する方法 [769 ページ]

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

1.10.4.1 スナップショットアイソレーション

スナップショットアイソレーションは、さまざまなバージョンのデータを管理することにより同時実行性と一貫性を向上させる目的で設計されています。

複数のユーザが同じデータに対して同時に読み込みと書き込みを行うと、ブロックやデッドロックが発生することがあります。トランザクションでスナップショットアイソレーションを使用すると、データベースサーバは読み込み要求の応答としてコミットされたバージョンのデータを返します。これは読み込みロックを取得せずに行われるため、データを書き込んでいるユーザとの干渉は発生しません。

スナップショットとは、データベースでコミットされた一連のデータです。スナップショットアイソレーションを使用すると、トランザクション内のすべてのクエリで同じデータのセットが使用されます。データベースのテーブルでロックは取得されません。これにより、他のトランザクションはブロックせずにアクセスして修正できます。処理中のスナップショットトランザクションでは、他のトランザクションにより変更されたすべてのデータのコピーをデータベースに保存しておく必要があります。スナップショットトランザクションを小さくしておくことで、パフォーマンスの影響を最小限にします。

スナップショットを作成するときに、スナップショットアイソレーションの 3 つのレベルを制御できます。

snapshot

トランザクションによって最初のローの読み込み、挿入、更新、または削除が行われた時点から、コミットされたデータのスナップショットを使用します。

statement-snapshot

文で最初のローが読み込まれた時点から、コミットされたデータのスナップショットを使用します。トランザクション内の各文で参照されるデータのスナップショットはそれぞれ異なる時点のものになります。

readonly-statement-snapshot

読み込み専用の文についてのみ、最初のローが読み込まれた時点から、コミットされたデータのスナップショットを使用します。トランザクション内の各読み込み専用文で参照されるデータのスナップショットはそれぞれ異なる時点のものになります。挿入、更新、削除の文については、updatable_statement_isolation オプションに指定された独立性レベル (0 (デフォルト)、1、2、3 のいずれか) を使用します。

BEGIN SNAPSHOT 文を使用して、トランザクションのスナップショットの開始時に指定するオプションもあります。

スナップショットアイソレーションは、主に次のような場合に便利です。

読み込みは多いが更新は少ないアプリケーション

スナップショットトランザクションは、データベースを修正する文の場合のみ、書き込みロックを取得します。トランザクションが主に読み込み操作を実行する場合、スナップショットトランザクションは、他のユーザと干渉する可能性のある読み込みロックを取得しません。

他のユーザがデータにアクセスする必要があるにもかかわらず、時間がかかるトランザクションを実行するアプリケーション

スナップショットトランザクションは読み込みロックを取得しないため、他のユーザはスナップショットトランザクションの実行中にデータの読み込みや更新を実行できます。

データベースからの一貫したデータのセットを読み取る必要があるアプリケーション

スナップショットはある時点からのコミット済みデータのセットを表示するため、トランザクションの実行中に他のユーザが変更を加えた場合でも、スナップショットアイソレーションを使用すれば、そのトランザクション内では変更されない一貫したデータを確認できます。

スナップショットアイソレーションは、すべてのユーザが共有するベーステーブルとグローバルテンポラリテーブルのみに影響します。その他のテーブルの種類では読み込み操作を行っても、古いバージョンのデータは表示されず、スナップショットは開

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

始されません。別のテーブルの種類に対する更新によってスナップショットが開始されるのは、isolation_level オプションが snapshot に設定されていて、更新によりトランザクションが開始される場合だけです。

文またはトランザクションスナップショットを使用する、WITH HOLD 句で開かれたカーソルがある場合、次の文は実行できません。

- ALTER INDEX 文
- ALTER TABLE 文
- CREATE INDEX 文
- DROP INDEX 文
- REFRESH MATERIALIZED VIEW 文
- REORGANIZE TABLE 文
- CREATE TEXT INDEX 文
- REFRESH TEXT INDEX 文

WITH HOLD 句でカーソルを開くと、スナップショットの開始時にコミットされたすべてのローのスナップショットが表示されます。カーソルが開かれているトランザクションの開始時点以降の接続で完了された変更もすべて表示されます。

高速トランケーションが実行されない場合にのみ、TRUNCATE TABLE を使用できます。これは、このような場合に個別の DELETE がトランザクションログに記録されるためです。

また、これらの文のいずれかをスナップショットでないトランザクションから実行した場合、すでに進行中のスナップショットトランザクションで、それ以降にテーブルを使用しようとすると、スキーマが変更されたことを示すエラーが返されます。

トランザクションのスナップショットが開始された後でビューが再表示された場合、マテリアライズドビューマッチングではそのビューは使用されません。

スナップショットアイソレーションのレベルは、すべてのプログラミングインタフェースでサポートされています。スナップショットアイソレーションのレベルは SET OPTION 文を使用して設定できます。

ローバージョン

スナップショットアイソレーションがデータベースで有効になると、ローが更新されるたびに、データベースサーバは元のローのコピーをテンポラリファイルに格納されたバージョンに追加します。元のローバージョンのエントリは、元のロー値にアクセスする必要がある可能性のある、すべてのアクティブなスナップショットトランザクションが完了するまで格納されます。スナップショットアイソレーションを使用するトランザクションは、コミット済みの値だけを確認できます。そのため、スナップショットトランザクションの開始前にローの更新がコミットされなかったかロールバックされた場合、スナップショットトランザクションは元のロー値にアクセスする必要があります。これにより、スナップショットアイソレーションを使用するトランザクションは、基本となるテーブルにロックを設定せずにデータを閲覧できます。

VersionStorePages データベースプロパティは、バージョンストア用に現在使用されているテンポラリファイル内のページ数を返します。この値を取得するには、次のクエリを実行します。

```
SELECT DB_PROPERTY ( 'VersionStorePages' );
```

古いローバージョンのエントリは、不要になると削除されます。BLOB の古いバージョンは、不要になるまで、テンポラリファイルではなく元のテーブルに格納されています。古いローバージョンのインデックスエントリは、不要になるまで元のインデックスに格納されています。

テンポラリファイル内の空き領域のサイズは、sa_disk_free_space システムプロシージャを使用すると取得できます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ロー値を更新するトリガが起動されると、それらのローの元の値もテンポラリファイルに格納されます。

短いトランザクションと短いスナップショットを使用するようにアプリケーションを設定すると、必要なテンポラリファイルの領域は減少します。

テンポラリファイルの増大に関心がある場合は、テンポラリファイルが特定のサイズに達したときに実行するアクションを指定する GrowTemp システムイベントを設定できます。

このセクションの内容:

スナップショットトランザクションの理解 [768 ページ]
スナップショットトランザクションは、更新時に書き込みロックを取得しますが、スナップショットを使用するトランザクションや文では読み込みロックを取得しません。その結果、読み込み処理は書き込み処理をブロックせず、書き込み処理は読み込み処理をブロックしません。ただし、書き込み処理は、同じローを更新しようとする他の書き込み処理をブロックすることがあります。

スナップショットアイソレーションを有効化する方法 [769 ページ]
データベースのスナップショットアイソレーションは、allow_snapshot_isolation オプションを使用して有効または無効にします。

スナップショットアイソレーションの例 [770 ページ]
スナップショットアイソレーションを使用して、ブロックなしで一貫性を維持できます。

更新の競合とスナップショットアイソレーション [772 ページ]
スナップショットアイソレーションを使用する場合、トランザクションがローの古いバージョンを認識し、それを更新または削除しようとすると、更新の競合が発生することがあります。

1.10.4.1.1 スナップショットトランザクションの理解

スナップショットトランザクションは、更新時に書き込みロックを取得しますが、スナップショットを使用するトランザクションや文では読み込みロックを取得しません。その結果、読み込み処理は書き込み処理をブロックせず、書き込み処理は読み込み処理をブロックしません。ただし、書き込み処理は、同じローを更新しようとする他の書き込み処理をブロックすることがあります。

スナップショットアイソレーションを使用すると、トランザクションは BEGIN TRANSACTION 文で開始されません。トランザクションで使用されるスナップショットアイソレーションのレベルに応じて、トランザクション内での最初の読み込み、挿入、更新、または削除時にトランザクションが開始します。次の例は、スナップショットアイソレーションでトランザクションが開始するタイミングを示します。

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
         SET TEMPORARY OPTION isolation_level = 'snapshot';
   SELECT * FROM Products; --transaction begins and the statement only
                           --sees changes that are already committed
   INSERT INTO Products
         SELECT ID + 30, Name, Description,
         'Extra large', Color, 50, UnitPrice, NULL
         FROM Products
         WHERE Name = 'Tee Shirt';
COMMIT; --transaction ends
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.4.1.2 スナップショットアイソレーションを有効化する方法

データベースのスナップショットアイソレーションは、allow_snapshot_isolation オプションを使用して有効または無効にします。

オプションを On にすると、ローバージョンがテンポラリファイル内で管理され、接続で任意のスナップショットアイソレーションのレベルを使用できます。オプションを Off にすると、スナップショットアイソレーションを使用しようとすると、エラーが発生します。

データベースがスナップショットアイソレーションを使用できるようにすると、パフォーマンスに影響を与える可能性があります。これは、スナップショットアイソレーションを使用するトランザクションの数に関係なく、修正されたすべてのローのコピーを保持する必要があるからです。

次の文は、データベースのスナップショットアイソレーションを有効にします。

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
```

allow_snapshot_isolation オプションの設定は、ユーザがデータベースに接続している場合でも変更できます。このオプションの設定を Off から On に変更した場合、新しいトランザクションがスナップショットアイソレーションを使用するには、現在のすべてのトランザクションが完了する必要があります。このオプションの設定を On から Off に変更した場合、データベースサーバがローバージョン情報の管理を停止するには、スナップショットアイソレーションを使用するすべての未処理のトランザクションが完了する必要があります。

データベースについて現在のスナップショットアイソレーションの設定を確認するには、SnapshotIsolationState データベースプロパティの値を問い合わせます。

```
SELECT DB_PROPERTY ( 'SnapshotIsolationState' );
```

SnapshotIsolationState プロパティの値は、次のいずれかです。

On

データベースでスナップショットアイソレーションが有効になっています。

Off

データベースでスナップショットアイソレーションが無効になっています。

in_transition_to_on

現在のトランザクションが完了するとスナップショットアイソレーションが有効になります。

in_transition_to_off

現在のトランザクションが完了するとスナップショットアイソレーションが無効になります。

スナップショットアイソレーションがデータベースで有効になると、スナップショットが使用されていない場合でも、トランザクションがコミットまたはロールバックしないかぎり、ローバージョンはトランザクションで管理される必要があります。そのため、スナップショットアイソレーションを使用しない場合は、allow_snapshot_isolation オプションを Off にすることをお奨めします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.4.1.3 スナップショットアイソレーションの例

スナップショットアイソレーションを使用して、ブロックなしで一貫性を維持できます。

例

このことを示すために、この例では、サンプルデータベースに対する 2 つの接続を使用しています。

1. 次のコマンドを実行して、サンプルデータベースへの Interactive SQL 接続 (Connection1) を作成します。

```
dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql;ConnectionName=Connection1"
```

2. 次のコマンドを実行して、サンプルデータベースへの Interactive SQL 接続 (Connection2) を作成します。

```
dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql;ConnectionName=Connection2"
```

3. Connection1 で、次の文を実行して独立性レベルを 1 (コミットされた読み出し) に設定します。

```
SET OPTION isolation_level = 1;
```

4. Connection1 で、次の文を実行します。

```
SELECT * FROM Products;
```

ID	Name	Description	Size	Color	Quantity	...
300	Tee Shirt	Tank Top	Small	White	28	...
301	Tee Shirt	V-neck	Medium	Orange	54	...
302	Tee Shirt	Crew Neck	One size fits all	Black	75	...
400	Baseball Cap	Cotton Cap	One size fits all	Black	112	...
...	...	...	...	...	...	...

5. Connection2 で、次の文を実行します。

```
UPDATE Products
SET Name = 'New Tee Shirt'
WHERE ID = 302;
```

6. Connection1 で、もう一度 SELECT 文を実行します。

```
SELECT * FROM Products;
```

Connection2 の UPDATE 文がコミットまたはロールバックされていないため、SELECT 文がブロックされ (停止ボタンのみが選択可能)、先に進むことはできません。SELECT 文は、Connection2 のトランザクションが完了するまで処理を待機する必要があります。これにより、SELECT 文はコミットされていないデータを結果に読み込みません。

7. Connection2 で、次の文を実行します。

```
ROLLBACK;
```

Connection2 のトランザクションが完了し、Connection1 の SELECT 文が処理されます。
独立性レベル statement snapshot を使用することで、ブロックなしで独立性レベル 1 と同じ同時実行性を実現します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

8. Connection1 で、次の文を実行してスナップショットアイソレーションを許可します。

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
```

9. Connection1 で、次の文を実行し、独立性レベルを statement snapshot に変更します。

```
SET TEMPORARY OPTION isolation_level = 'statement-snapshot';
```

10. Connection1 で、次の文を実行します。

```
SELECT * FROM Products;
```

11. Connection2 で、次の文を実行します。

```
UPDATE Products
SET Name = 'New Tee Shirt'
WHERE ID = 302;
```

12. Connection1 で、もう一度 SELECT 文を実行します。

```
SELECT * FROM Products;
```

SELECT 文は、ブロックされずに実行されますが、Connection2 によって実行された UPDATE 文からのデータは含まれません。

13. Connection2 で、次の文を実行してトランザクションを終了します。

```
COMMIT;
```

14. Connection1 で、トランザクション (Products テーブルに対するクエリ) を終了し、もう一度 SELECT 文を実行して更新されたデータを確認します。

```
COMMIT;
SELECT * FROM Products;
```

ID	Name	Description	Size	Color	Quantity	...
300	Tee Shirt	Tank Top	Small	White	28	...
301	Tee Shirt	V-neck	Medium	Orange	54	...
302	New Tee Shirt	Crew Neck	One size fits all	Black	75	...
400	Baseball Cap	Cotton Cap	One size fits all	Black	112	...
...	...	...	...	...	...	...

15. 次の文を実行し、サンプルデータベースに対する変更を取り消します。

```
UPDATE Products
SET Name = 'Tee Shirt'
WHERE id = 302;
COMMIT;
```

関連情報

チュートリアル: ダーティリードの知識 [809 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

チュートリアル: 繰り返し不可能読み出しの知識 [814 ページ]
チュートリアル: 幻ローの知識 [820 ページ]
チュートリアル: 幻ロックの知識 [826 ページ]

1.10.4.1.4 更新の競合とスナップショットアイソレーション

スナップショットアイソレーションを使用する場合、トランザクションがローの古いバージョンを認識し、それを更新または削除しようとすると、更新の競合が発生することがあります。

このような状況では、競合が検出されるとサーバでエラーが発生します。コミットされた変更の場合、これは更新または削除が試みられた時になります。コミットされていない変更の場合、更新または削除はブロックされ、変更がコミットされるときにサーバがエラーを返します。

readonly-statement-snapshot を使用すると、更新可能な文は、スナップショットアイソレーションではなく実行し、常に最新バージョンのデータベースを認識するため、更新の競合は発生しません。そのため、readonly-statement-snapshot 独立性レベルには、スナップショットアイソレーションの多くの利点があり、元々別の独立性レベルで実行するように設計されたアプリケーションを大きく変更する必要はありません。readonly-statement-snapshot 独立性レベルを使用するときは、次のようになります。

- 読み込み専用文では、読み込みロックは取得されません。
- 読み込み専用文は、データベースのコミットされた状態を常に認識します。

1.10.4.2 典型的な矛盾のケース

トランザクションの同時実行中に発生する可能性のある典型的な矛盾には 3 つのタイプがあります。

これらの 3 つのタイプは ISO SQL 標準に記載されており、低い独立性レベルで発生する可能性のある動作という観点で定義されています。他のタイプの矛盾も発生し得るため、この 3 つがすべてというわけではありません。

ダーティリード

トランザクション A がローを修正し、変更のコミットもロールバックもしないとします。その修正がコミットまたはロールバックされる前に、トランザクション B がそのローを読みます。その後、COMMIT が実行される前に、トランザクション A がさらにそのローを変更するか、またはその修正をロールバックしたとします。いずれの場合も、トランザクション B はコミットされなかった状態でローを読んでしまったことになります。

繰り返し不可能読み出し

トランザクション A がローを読みます。次にトランザクション B がそのローを修正または削除して、COMMIT を実行します。トランザクション A がもう一度同じローを読もうとしたときには、ローは修正されているか、削除されてしまっています。

幻ロー

トランザクション A が、一定の条件を満たす一連のローを読みます。次に、トランザクション B は、前にトランザクション A の条件を満たさなかったローで INSERT または UPDATE を実行します。トランザクション B はこれらの変更をコミットします。新しくコミットされたローはトランザクション A の条件を満たします。トランザクション A がもう一度データを読み込むと、更新されたローのセットを取得します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

独立性レベルとダーティリード、繰り返し不可能読み出し、幻ロー

データベースサーバでは、使用する独立性レベルに応じて、ダーティリード、繰り返し不可能読み出し、幻ローが許可されます。次の表で X は、その独立性レベルで動作が許可されていることを表します。

独立性レベル	ダーティリード	繰り返し不可能読み出し	幻ロー
0 - コミットされない読み出し	X	X	X
readonly-statement-snapshot	X[1]	X[2]	X[3]
1 - コミットされた読み出し		X	X
statement-snapshot		X[2]	X[3]
2 - 繰り返し可能読み出し			X
3 - 直列化可能			
snapshot			

[1] ダーティリードは、updatable_statement_isolation オプションで指定された独立性レベルがその発生を妨げない場合に、トランザクション内の更新可能な文で発生することがあります。

[2] 繰り返し不可能読み出しは、updatable_statement_isolation オプションで指定された独立性レベルがその発生を妨げない場合に、トランザクション内の文で発生することがあります。繰り返し不可能読み出しは、各文によって新しいスナップショットが開始する場合に発生することがあります。そのため、ある文が認識しない変更を別の文が認識する可能性があります。

[3] 幻ローは、updatable_statement_isolation オプションで指定された独立性レベルがその発生を妨げない場合に、トランザクション内の文で発生することがあります。幻ローは、各文によって新しいスナップショットが開始する場合に発生することがあります。そのため、ある文が認識しない変更を別の文が認識する可能性があります。

この表から、以下の 2 つのことがわかります。

- 各独立性レベルは、3 つの典型的な矛盾のケースのうち 1 つを防止します。
- 各レベルは、それ以下のレベルで防止される矛盾のケースも防止します。
- statement snapshot 独立性レベルでは、繰り返し不可能読み出しと幻ローは、トランザクション内で発生する可能性がありますが、トランザクションの単一文内では発生しません。

ODBC では、独立性レベルに異なる名前が付いています。これらの名前は、それぞれのレベルで防止される矛盾の名前に基づいて付けられています。

このセクションの内容:

カーソル不安定性 [774 ページ]
トランザクションの同時実行中に発生する可能性のある重大な矛盾がカーソル不安定性です。

関連情報

ODBC 実行可能アプリケーションでの独立性レベル [776 ページ]
チュートリアル: ダーティリードの知識 [809 ページ]
チュートリアル: 繰り返し不可能読み出しの知識 [814 ページ]
チュートリアル: 幻ローの知識 [820 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.4.2.1 カーソル不安定性

トランザクションの同時実行中に発生する可能性のある重大な矛盾がカーソル不安定性です。

この矛盾が起きると、あるトランザクションのカーソルが参照しているローを、他のトランザクションが修正できてしまいます。カーソルを使用するアプリケーションでは、カーソル安定性があれば、データベース内のデータに対する矛盾を確実に回避できます。

例

トランザクション A はカーソルを使用してローを読み込みます。トランザクション B が、そのローを修正し、コミットします。修正されたことに気づかず、トランザクション A がそのローを修正します。

カーソル不安定性への対処

カーソル安定性は、独立性レベル 1、2、3 で提供されます。カーソル安定性により、カーソルの現在のローに含まれる情報を、他のトランザクションが修正できなくなります。カーソルのローの情報は、特定のテーブルに含まれる情報のコピーや、複数のテーブルの異なるローにあるデータを組み合わせたものです。SELECT 文内でジョインまたはサブ選択を使用する場合、複数のテーブルが関係する可能性があります。

カーソルが使用されるのは、他のアプリケーションを介して SQL Anywhere を使っているときのみです。

基本となるデータに加えた変更がカーソルを使用するアプリケーションから参照できるかどうかは、カーソルを使用するアプリケーションと関連していますが、別個の問題です。変更がアプリケーションから参照できるかどうかは、カーソルの sensitivity を指定して制御できます。

関連情報

ストアドプロシージャ、トリガ、バッチ、ユーザ定義関数 [87 ページ]

1.10.4.3 独立性レベルの設定方法

データベースへの各接続は独自の独立性レベルを持ちます。

さらに、データベースはユーザやユーザ拡張ロールごとにデフォルトの独立性レベルを保存できます。isolation_level データベースオプションで PUBLIC を設定すると、デフォルトの独立性レベルを設定できます。

テーブルヒントを使用して独立性レベルを設定することもできますが、これは個別の文に独立性レベルを設定するための高度な機能です。

接続の独立性レベルや、ユーザ ID に設定されたデフォルトのレベルは、SET OPTION 文を使用して変更できます。他のユーザまたはグループの独立性レベルを変更することもできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

デフォルトの独立性レベル

データベースに接続すると、データベースサーバは次のように最初の独立性レベルを決定します。

1. デフォルトの独立性レベルは、ユーザやロールごとに設定できます。レベルがユーザ ID のデータベースに保存されている場合、データベースサーバはそのレベルを使用します。
2. レベルが保存されていない場合、データベースサーバはレベルが見つかるまで、ユーザが属しているグループをチェックします。最初に他の設定が見つからない場合、データベースサーバは PUBLIC に割り当てられているレベルを使用します。

注記

スナップショットアイソレーションを使用する場合は、最初にデータベースでスナップショットアイソレーションを有効にする必要があります。

例

現在のユーザの独立性レベルの設定 - SET OPTION 文を実行します。たとえば、次の文は、現在のユーザの独立性レベルを 3 に設定します。

```
SET OPTION isolation_level = 3;
```

ユーザまたは **PUBLIC** ロールの独立性レベルの設定

1. データベースに接続します。
2. isolation_level の前に被付与者名とピリオドを付加し、SET OPTION 文を実行します。たとえば、次の文は、PUBLIC ロールのデフォルトの独立性レベルを 3 に設定します。

   ```
   SET OPTION PUBLIC.isolation_level = 3;
   ```

現在の接続の独立性レベルの設定 - TEMPORARY キーワードを使用して SET OPTION 文を実行します。たとえば、次の文は、現在の接続に対して独立性レベルを 3 に設定します。

```
SET TEMPORARY OPTION isolation_level = 3;
```

関連情報

スナップショットアイソレーションを有効化する方法 [769 ページ]
トランザクション内の独立性レベルの変更 [777 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.4.4 ODBC 実行可能アプリケーションでの独立性レベル

ODBC アプリケーションは、SQLSetConnectAttr を呼び出すときに、Attribute を SQL_ATTR_TXN_ISOLATION に、ValuePtr を対応する独立性レベルに設定します。

ValuePtr パラメータ

ValuePtr	独立性レベル
SQL_TXN_READ_UNCOMMITTED	0
SQL_TXN_READ_COMMITTED	1
SQL_TXN_REPEATABLE_READ	2
SQL_TXN_SERIALIZABLE	3
SA_SQL_TXN_SNAPSHOT	snapshot
SA_SQL_TXN_STATEMENT_SNAPSHOT	statement-snapshot
SA_SQL_TXN_READONLY_STATEMENT_SNAPSHOT	readonly-statement-snapshot

ODBC を介した独立性レベルの変更

ODBC を介して接続の独立性レベルを変更するには、`ODBC32.dll` ライブラリの SQLSetConnectAttr 関数を使用します。

SQLSetConnectAttr 関数は、4 つのパラメータ (ODBC コネクションハンドルの値、独立性レベルの設定要求、設定する独立性レベルに対応する値、ゼロ) を取ります。独立性レベルに対応する値を、下のテーブルに表示します。

文字列	値
SQL_TXN_ISOLATION	108
SQL_TXN_READ_UNCOMMITTED	1
SQL_TXN_READ_COMMITTED	2
SQL_TXN_REPEATABLE_READ	4
SQL_TXN_SERIALIZABLE	8
SA_SQL_TXN_SNAPSHOT	32
SA_SQL_TXN_STATEMENT_SNAPSHOT	64
SA_SQL_TXN_READONLY_STATEMENT_SNAPSHOT	128

ODBC アプリケーションから SET OPTION 文を使用して、独立性レベルを変更しないでください。ODBC ドライバは文を解析しないため、ODBC で文を実行しても ODBC ドライバによって認識されません。このため、ロックが予期しない動作をする場合があります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

例

次の関数呼び出しで、独立性レベルを statement-snapshot に設定します。

```
SQLSetConnectAttr (dbc, SA_SQL_ATTR_TXN_ISOLATION, (SQLPOINTER*)
SA_SQL_TXN_STATEMENT_SNAPSHOT, 0);
```

ODBC は、独立性機能を使用して各種のデータベースロックオプションをサポートします。たとえば、PowerBuilder では、データベースに接続するときに、トランザクションオブジェクトの Lock 属性を使用して独立性レベルを設定できます。Lock 属性は文字列で、次のように設定されます。

```
SQLCA.lock = "RU"
```

Lock オプションは、CONNECT が発生したときだけ有効になります。CONNECT の後に Lock 属性を変更しても、接続には影響しません。

このセクションの内容:

トランザクション内の独立性レベルの変更 [777 ページ]
1 つのトランザクションの中の異なる部分に、別々の独立性レベルを設定したい場合、

1.10.4.4.1 トランザクション内の独立性レベルの変更

1 つのトランザクションの中の異なる部分に、別々の独立性レベルを設定したい場合、

データベースサーバでは、トランザクションの途中でデータベースの独立性レベルを変更できます。

トランザクション中に isolation_level オプションを変更すると、新しい設定は次の項目だけに反映されます。

- 変更後に表示されたカーソル
- 変更後に実行された文

トランザクションの途中で独立性レベルを変更し、トランザクションが実施するロック数を制御する必要がある場合があります。トランザクションで大きなテーブルを読み込む必要があるが、詳細な作業をするのは一部のローだけという場合もあります。矛盾が生じてもトランザクションには重大な影響を及ぼさない場合は、その大きなテーブルをスキャンする間は独立性レベルを低レベルに設定し、他の処理が遅れないようにしてください。

たとえば、1 つのテーブルまたはテーブルグループだけがシリアルアクセスを要求している場合などは、トランザクション中に独立性レベルを変更できます。

幻ローの理解に関するチュートリアルでは、トランザクション中に独立性レベルを変更する例を参照できます。

注記

FROM 句で WITH `table-hint` 句を指定して独立性レベルを設定することもできますが (レベル 0 ～ 3 のみ)、これは高度な機能であるため必要な場合にのみ使用してください。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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の独立性レベルの変更

スナップショットアイソレーションを使用するときは、トランザクション内で独立性レベルを変更できます。これを行うには、isolation_level オプションの設定を変更するか、クエリ内の独立性レベルに影響するテーブルヒントを使用します。いつでも statement-snapshot、readonly-statement-snapshot、独立性レベル 0 ～ 3 を使用できます。ただし、snapshot 以外の独立性レベルでトランザクションを開始した場合は、そのトランザクションで snapshot 独立性レベルを使用できません。トランザクションは更新によって開始され、次の COMMIT または ROLLBACK まで続行します。最初の更新がスナップショット以外の独立性レベルで開始された場合、トランザクションがコミットまたはロールバックする前にスナップショット独立性レベルを使用しようとする文を実行すると、エラー -1065 (SQLE_NON_SNAPSHOT_TRANSACTION) が返されます。次に例を示します。

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
BEGIN TRANSACTION
    SET OPTION isolation_level = 3;
    INSERT INTO Departments
        ( DepartmentID, DepartmentName, DepartmentHeadID )
        VALUES( 700, 'Foreign Sales', 129 );
    SET TEMPORARY OPTION isolation_level = 'snapshot';
    SELECT * FROM Departments;
```

関連情報

チュートリアル: 幻ローの知識 [820 ページ]

1.10.4.5 独立性レベルの参照

CONNECTION_PROPERTY 関数を使用して、現在の接続の独立性レベルを表示します。

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。

手順

次の文を実行します。

```
SELECT CONNECTION_PROPERTY( 'isolation_level' );
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

現在の接続の独立性レベルが返されます。

1.10.5 トランザクションのブロックとデッドロック

トランザクションの実行中、データベースサーバはローにロックをかけ、処理中のローが他のトランザクションからの干渉を受けないようにします。

ロックは、許可する干渉の量とタイプを制御します。

データベースサーバでは、「トランザクションブロック」の使用により、干渉をまったくなくすか制限して、トランザクションを同時に実行できます。トランザクションはロックを取得し、同時に実行されている他のトランザクションが特定のローを修正したり、アクセスしたりすることを防止できます。このトランザクションブロックスキームは、いくつかのタイプの干渉を常に防止します。たとえば、テーブル内の特定ローを更新するトランザクションは、そのローのロックを取得し、他のトランザクションが同じローを同時に更新または削除できないようにします。

トランザクションのブロック

あるトランザクションが操作を実行しようとしたにもかかわらず、他のトランザクションが保持するロックによって妨げられた場合は、競合が発生し、操作を実行しようとしたトランザクションの進行は妨げられます。

一連のトランザクションが、どれも進行できない状態になることもあります。

このセクションの内容:

ブロックオプション [779 ページ]
2 つのトランザクションが、ある 1 つのローに対してそれぞれ読み込みロックをかけている場合、一方がローを変更しようとしたときにどうなるかは、データベースのブロックオプションの設定によって異なります。

デッドロック [780 ページ]
トランザクションのブロックによって、デッドロックが起こる可能性があります。デッドロックとは、トランザクションの集合で、そのどれもが処理を進行できない状態をいいます。

1.10.5.1 ブロックオプション

2 つのトランザクションが、ある 1 つのローに対してそれぞれ読み込みロックをかけている場合、一方がローを変更しようとしたときにどうなるかは、データベースのブロックオプションの設定によって異なります。

ローを修正するトランザクションは、他方のトランザクションをブロックしなければなりませんが、他方のトランザクションにブロックされている間はそれができません。

- ブロックオプションが On (デフォルト) の場合、書き込みをしようとするトランザクションは、もう一方のトランザクションが読み込みロックを解放するまで待機します。解放されると、書き込みが実行されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ブロックオプションが Off の場合、書き込みをしようとする文はエラーを受け取ります。

ブロックオプションが Off の場合、文は待機せず終了し、行った部分的な変更はロールバックされます。この場合は、あとでもう一度トランザクションの実行を試みます。

ブロックは、独立性レベルが高くなると起こりやすくなります。ロックもチェックの回数も独立性とともに増えるからです。独立性レベルが高いと、通常は同時実行性が低下します。低下の度合いは、同時に実行するトランザクションによって異なります。

1.10.5.2 デッドロック

トランザクションのブロックによって、デッドロックが起こる可能性があります。デッドロックとは、トランザクションの集合で、そのどれもが処理を進行できない状態をいいます。

デッドロックの理由

デッドロックが発生する理由は次の 2 つです。

環状ブロックの競合

トランザクション A がトランザクション B にブロックされ、トランザクション B がトランザクション A にブロックされている状態。この状態から脱け出すには、どちらかのトランザクションをキャンセルします。同様の状況は 3 つ以上のトランザクションが環状にブロックされた場合にも発生します。

トランザクションのデッドロックを排除するために、データベースサーバはデッドロックに関わっている接続を選択し、その接続でアクティブなトランザクションの変更をロールバックして、エラーを返します。データベースサーバは内部のヒューリスティックを使用して、ロールバック対象の接続を選択します。その際、blocking_timeout オプションによって決められた残りブロック待機時間が最も短い接続が優先されます。すべての接続が永久に待機するように設定されている場合は、サーバによってデッドロックが検出された接続が、犠牲の接続として選択されます。

すべてのワーカがブロックされています。

トランザクションがブロックされても、ワーカは放棄されたわけではありません。たとえば、データベースサーバが 3 つのワーカに設定されており、トランザクション A、B、C が現在要求を実行していないトランザクション D によってブロックされると、これ以上使用できるワーカがなくなるため、デッドロック状態が発生します。この状況は、スレッドデッドロックと呼ばれます。

データベースサーバに n ワーカを設定したと想定します。n-1 の数のワーカがブロックされているときに最後のワーカをブロックしようとすると、スレッドデッドロックが発生します。データベースサーバのカーネルは、最後のワーカをブロックすることを許可できません。ブロックすることによって、すべてのワーカがブロックされ、データベースサーバがハングするためです。代わりに、データベースサーバは最後のワーカをブロックしようとしたタスクを終了し、その接続でアクティブなトランザクションの変更をロールバックして、エラー (SQLCODE -307、SQLSTATE 40W06) を返します。

数十または数百の接続のあるデータベースサーバでは、データベースのサイズまたはブロックが原因で、多数の要求の実行時間が長くなる場合、スレッドデッドロックが発生することがあります。この場合、データベースサーバのマルチプログラミングレベルを高くすることが、適切な解決法となることがあります。また、アプリケーションの設計によって、過度な競合または意図しない競合が発生し、スレッドデッドロックの原因となる場合もあります。この場合、アプリケーションでより大きなデータセットを使用するようにすると、問題が悪化することがあります。データベースサーバのマルチプログラミングレベルを高くしても、問題は解決されないことがあります。

サーバが使用するデータベーススレッドの数は、個々のデータベースの設定によって異なります。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

例: デッドロックでブロックされるユーザの決定 [781 ページ]
sa_conn_info システムプロシージャを使用するイベントを作成し、どの接続がデッドロックでブロックされているかを判別します。

1.10.5.2.1 例: デッドロックでブロックされるユーザの決定

sa_conn_info システムプロシージャを使用するイベントを作成し、どの接続がデッドロックでブロックされているかを判別します。

このプロシージャは、接続ごとに 1 つのローで構成される結果セットを返します。結果セットのカラムの 1 つには、接続がブロックされているかどうか、およびブロックされている場合は、どの接続でブロックされているかが一覧表示されます。結果セットは、接続がブロックされているかどうかと、ブロックの原因となっている接続を示します。

デッドロックが発生した場合は、デッドロックイベントを使用してアクションを実行することもできます。イベントハンドラでは、sa_report_deadlocks プロシージャを使用して、デッドロックが発生するに至った状況に関する情報を取得できます。データベースサーバからデッドロックの詳細を取り出すには、log_deadlocks オプションを使用して RememberLastStatement 機能を有効にします。

アプリケーションで頻繁にデッドロックが発生する場合は、プロファイラ を使用するとデッドロックの原因の究明に役立ちます。

例

次の例では、デッドロックが発生した場合にその情報を取得するために使用できるテーブルとシステムイベントの設定方法を示します。

1. 次のように、sa_report_deadlocks システムプロシージャが返すデータを格納するテーブルを作成します。

```
CREATE TABLE DeadlockDetails(
  deadlockId INT PRIMARY KEY DEFAULT AUTOINCREMENT,
  snapshotId BIGINT,
  snapshotAt TIMESTAMP,
  waiter INTEGER,
  who VARCHAR(128),
  what LONG VARCHAR,
  object_id UNSIGNED BIGINT,
  record_id BIGINT,
  owner INTEGER,
  is_victim BIT,
  rollback_operation_count UNSIGNED INTEGER );
```

2. デッドロックが発生したときに起動するイベントを作成します。
このイベントにより、sa_report_deadlocks システムプロシージャの結果がテーブルにコピーされ、管理者にデッドロックが通知されます。

```
CREATE EVENT DeadlockNotification
TYPE Deadlock
HANDLER
BEGIN
 INSERT INTO DeadlockDetails WITH AUTO NAME
 SELECT snapshotId, snapshotAt, waiter, who, what, object_id, record_id,
        owner, is_victim, rollback_operation_count
    FROM sa_report_deadlocks ();
 COMMIT;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
 CALL xp_startmail ( mail_user ='George Smith',
                     mail_password ='mypwd' );
 CALL xp_sendmail( recipient='DBAdmin',
                   subject='Deadlock details added to the DeadlockDetails
table.' );
 CALL xp_stopmail ( );
END;
```

3. log_deadlocks オプションを On に設定します。

```
SET OPTION PUBLIC.log_deadlocks = 'On';
```

4. 最後に実行された文のロギングを有効にします。

```
CALL sa_server_option( 'RememberLastStatement', 'YES' );
```

1.10.6 ロックの仕組み

ロックは、複数のトランザクションを同時に実行しているときにデータの整合性を保護する同時制御メカニズムです。

データベースサーバでは、2 つの接続によって同じデータが同時に変更されないようにするために、また変更処理の最中に他の接続によってデータが読み込まれないようにするために、自動的にロックが適用されます。ロックによって、更新中の情報が保護され、クエリ結果の一貫性が高められます。

データベースサーバは自動的にこれらのロックを設定するので、明示的な指示は必要ありません。あるトランザクションによって獲得されたすべてのロックは、たとえば COMMIT 文または ROLLBACK 文によってそのトランザクションが完了するまで、データベースサーバで保持されます。このルールには 1 つのみ例外があります。

ローにアクセスしているトランザクションは、ロックを保持しているといえます。ロックの種類により、他のトランザクションのそのローへのアクセスは限定されるか、まったくできなくなります。

このセクションの内容:

ロックの種類 [783 ページ]
データベースの一貫性を実現し、トランザクション間で適切な独立性レベルをサポートするために、データベースサーバでは複数の種類のロックを使用します。

ロックの競合 [790 ページ]
ロックの競合が発生するのは、あるトランザクションが、別のトランザクションがロックをかけているローに排他ロックをかけようとした場合や、別のトランザクションが排他ロックをかけているローに共有ロックをかけようとした場合です。

クエリ時のロック [791 ページ]
ユーザが SELECT 文を入力したときにデータベースサーバが使用するロックは、トランザクションの独立性レベルによって異なります。

挿入時のロック [793 ページ]
挿入操作によって新しいローが作成されます。データベースサーバは挿入時にさまざまなタイプのロックを利用して、データの整合性を保証します。

更新時のロック [795 ページ]
データベースサーバは、ロックを使用するときに特定のレコードに含まれる情報を修正します。

削除時のロック [796 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

DELETE オペレーションは、INSERT オペレーションとほとんど同じ手順を実行しますが、その順序は反対になります。

ロック期間 [797 ページ]

ロックは、通常、トランザクションが完了するまでトランザクションによって保持されます。

ロックの表示 (SQL Central の場合) [797 ページ]

SQL Central の**ロック**タブを使用して、データベースに現在保持されているロックを表示します。

ロックの表示 (Interactive SQL) [798 ページ]

ロックされているローの情報を取得して、データベースのロックの問題を診断します。

ミューテックスおよびセマフォ [799 ページ]

アプリケーションロジックのミューテックスおよびセマフォを使用して、ロック動作を実現し、リソースの可用性を制御および伝達します。

1.10.6.1 ロックの種類

データベースの一貫性を実現し、トランザクション間で適切な独立性レベルをサポートするために、データベースサーバでは複数の種類のロックを使用します。

スキーマロック

スキーマロックは、データベーススキーマへの変更を直列化し、またほかの接続によって初期化されたスキーマの変更により、テーブルを使用するトランザクションが影響を受けないようにします。たとえば、テーブルの構造を変更するために新しいカラムを挿入するトランザクションはテーブルをロックして、他のトランザクションがスキーマの変更による影響を受けないようにできます。このような場合は、他のトランザクションのアクセスを制限してエラーを回避することが不可欠です。

ローロック

ローロックによって、ローレベルでの複数のユーザのアクセスと特定のテーブルの変更を可能にするため、同時に実行しているトランザクション間に一貫性が実現されます。たとえば、トランザクションは特定のローをロックして、他のトランザクションがそのローを変更するのを防ぐことができます。ローロックには、読み込み (共有) ロック、書き込み (排他) ロック、意図的ロックというクラスがあります。

テーブルロック

テーブルロックによってすべてのローをロックすることで、トランザクションがテーブルを更新している間に、別のトランザクションが同じテーブルを更新するのを防ぎます。テーブルロックには、読み込み (共有) ロック、書き込み (排他) ロック、意図的ロックというタイプがあります。

位置ロック

位置ロックによって、テーブルの逐次スキャンまたはインデックススキャンにおける一貫性を実現します。通常、トランザクションはローを逐次スキャンするか、インデックスによって指定された順序に従ってスキャンします。いずれの場合も、スキャン位置にロックをかけることができます。たとえば、インデックスにロックをかけると、別のトランザクションがインデックスに特定の値や値の範囲を持つローを挿入しないようにできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ロック期間

ロックは、通常、トランザクションが完了するまでトランザクションによって保持されます。この動作によって、他のトランザクションが変更を加えたことによって、元のトランザクションがロールバックできなくなることが回避されます。独立性レベル 3 では、トランザクションの直列可能性を保障するために、トランザクションが終了するまですべてのロックが保持されます。

カーソルの安定性を実装するためにローロックを使用する場合、ロックはトランザクションの終了まで保持されません。これらのローロックは、対象のローがカーソルの現在のローである間、保持されます。ほとんどの場合、ロックの継続時間はトランザクションの存続時間よりも短くなります。カーソルが WITH HOLD で開かれている場合、接続が存続する間ロックが保持されることがあります。

ロックは、次の期間で保持されます。

位置

特定のローにおける読み込みロックのような短期間のロックで、独立性レベル 1 でカーソルの安定性を実装するために使用される

トランザクション

たとえば、トランザクションの終了まで保持されるローロック、テーブルロック、位置ロック

接続

WITH HOLD カーソルの使用時に作成されるスキーマロックのように、トランザクションが終了しても保持されるスキーマロック

このセクションの内容:

スキーマロック [785 ページ]
スキーマロックは、データベーススキーマへの変更を直列化し、またほかの接続によって初期化されたスキーマの変更により、テーブルを使用するトランザクションが影響を受けないようにします。

ローロック [785 ページ]
ローロックは、更新内容の消失のようなトランザクションの矛盾を防ぎます。

テーブルロック [787 ページ]
テーブルロックは、トランザクションがテーブルを更新している間に、別のトランザクションが同じテーブルを更新するのを防ぎます。

位置ロック [788 ページ]
位置ロックは、幻または幻ローが存在するための異常状態を避けるために設計された、キー範囲のロック形式です。

関連情報

スキーマロック [785 ページ]
ローロック [785 ページ]
テーブルロック [787 ページ]
位置ロック [788 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.6.1.1 スキーマロック

スキーマロックは、データベーススキーマへの変更を直列化し、またほかの接続によって初期化されたスキーマの変更により、テーブルを使用するトランザクションが影響を受けないようにします。

たとえば共有スキーマロックを使用すると、別の接続上のオープンカーソルでテーブルを読み取り中に、ALTER TABLE 文によってそのテーブルからカラムが削除されるのを防ぐことができます。

スキーマロックには、共有と排他の 2 つのクラスがあります。

共有ロック

共有スキーマロックは、トランザクションがデータベース内のテーブルを直接的または間接的に参照するときに取得されます。共有スキーマロックは他の共有スキーマロックと競合しません。同時に同じテーブルで共有スキーマロックを取得できるトランザクションの数に制限はありません。共有スキーマロックは、トランザクションが COMMIT または ROLLBACK で完了するまで保持されます。

共有スキーマロックを保持する接続は、変更が他の接続と競合しない場合に、テーブルデータを変更できます。テーブルスキーマは共有 (読み込み) モードでロックされる。

排他ロック

排他スキーマロックは、テーブルのスキーマが修正されるときに取得されます。スキーマは、通常は DDL 文を使用して修正されます。修正前にテーブルで排他スキーマロックを取得する DDL 文には、ALTER TABLE 文などがあります。あるテーブルに対して一度に 1 つの接続だけが、排他スキーマロックを取得できます。他のすべての接続は、テーブルのスキーマをロック (共有または排他) しようとしても、ブロックされるかエラーで失敗します。独立性レベル 0 (最も制限が少ない独立性レベル) で実行している接続は、スキーマが排他モードでロックされたテーブルからローを読み取ろうとするとブロックされます。

テーブルスキーマの排他ロックを保持する接続のみがテーブルデータを変更できます。テーブルスキーマは、単一接続の排他的使用のためにロックされます。

1.10.6.1.2 ローロック

ローロックは、更新内容の消失のようなトランザクションの矛盾を防ぎます。

ローロックでは、暗黙的または明示的な COMMIT 文を発行して変更をコミットするか、ROLLBACK 文で変更をアボートすることによりトランザクションが完了するまで、そのトランザクションによって修正されるローは別のトランザクションによって修正されません。

ローロックには、読み込み (共有) ロック、書き込み (排他) ロック、意図的ロックという 3 つのクラスがあります。データベースサーバは、トランザクションごとにこれらのロックを自動的に取得します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

読み込みロック

トランザクションがローを読み込むと、トランザクションのアイソレーションレベルは読み込みロックが取得されているかどうかを判断します。一度読み込みロックのかかったローに対しては、他のどのトランザクションも書き込みロックを取得できません。読み込みロックが取得されると、ローの読み込み中は、別のトランザクションはそのローを修正または削除しません。任意のローに同時に取得できる読み込みロックの数に制限はありません。そのため、読み込みロックは、共有ロックまたは非排他ロックと呼ばれることもあります。

読み込みロックは、保持される期間が異なることがあります。独立性レベル 2 と 3 では、トランザクションが取得したどの読み込みロックも、トランザクションが COMMIT または ROLLBACK によって完了するまで保持されます。これらの読み込みロックは、長期間の読み込みロックと呼ばれます。

独立性レベル 1 で実行されるトランザクションの場合、データベースサーバはカーソルが位置するローで短期間の読み込みロックを取得します。アプリケーションがカーソルをスクロールすると、カーソルが直前に位置していたローで短期間の読み込みロックは解放され、新しい短期間の読み込みロックがその次のローで取得されます。この技術はカーソルの安定性と呼ばれます。アプリケーションは現在のローで読み込みロックを保持するため、アプリケーションがそのローからカーソルを移動するまで、別のトランザクションがそのローに対して変更を加えることができません。カーソルが複数のテーブルを伴うクエリに対する場合は、複数のロックを取得できます。短期間の読み込みロックは、カーソル内の位置が要求 (通常は、これらの要求はアプリケーションによって発行される FETCH 文) 間で維持される必要がある場合だけ取得されます。たとえば SELECT COUNT(*) クエリの処理時は短期間の読み込みロックは取得されません。この文で開かれているカーソルが、ベーステーブルの特定ローに位置することがないためです。この場合、データベースサーバは、コミットされた読み出しのセマンティック、つまりこの文で処理されるローは他のトランザクションによってコミットされたことを保証すれば良いことになります。

独立性レベル 0 (コミットされない読み出し) で実行されるトランザクションの場合は、長期間または短期間の読み込みロックを取得しないため、他のトランザクションと競合しません (排他スキーマロックの場合を除く)。ただし、独立性レベル 0 のトランザクションは、同時に実行している他のトランザクションによって加えられたコミットされていない変更を処理することがあります。コミットされていない変更を処理しないようにするには、スナップショットアイソレーションを使用します。

書き込みロック

トランザクションがローを追加、更新、削除するときには、書き込みロックが設定されます。この動作は、独立性レベル 0 やスナップショットアイソレーションのレベルを含む、どの独立性レベルのトランザクションでも当てはまります。書き込みロックが設定されると、他のトランザクションはそのローに対して読み込みロック、意図的ロック、書き込みロックのいずれも取得できません。あるローに対して排他的なロックを保持できるトランザクションは一度に 1 つだけであるため、書き込みロックは排他ロックとも呼ばれます。他のトランザクションが同じローに何らかの種類のロックをかけている間は、書き込みロックを取得することはできません。同様に、トランザクションが書き込みロックを取得すると、他のトランザクションによるそのローへのロック要求は拒否されます。

意図的ロック

意図的ロックは更新を意図したロックとも呼ばれ、特定のローを修正する意図を表します。意図的ロックは、トランザクションが次の場合に取得されます。

- FETCH FOR UPDATE 文を発行する。
- SELECT...FOR UPDATE BY LOCK 文を発行する。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

- ODBC アプリケーションで SQL_CONCUR_LOCK を同時実行性の基準として使用する (SQLSetStmtAttr ODBC API 呼び出しの SQL_ATTR_CONCURRENCY パラメータを使用して設定する)。
- SELECT...FROM T WITH (UPDLOCK) 文を発行する。

意図的ロックは、読み込みロックと競合しないため、意図的ロックを取得しても、他のトランザクションが同じローを読み込むことを妨げません。ただし、意図的ロックは、他のトランザクションが同じローで意図的ロックまたは書き込みロックを取得することを妨げるため、更新に先立って他のトランザクションによってローが変更されることはありません。

スナップショットアイソレーションを使用するトランザクションによって意図的ロックが取得されるのは、そのローがデータベースで未修正のローであり、かつすべての同時実行トランザクションに共通である場合に限られます。ただし、ローがスナップショットのコピーである場合は、元のローが別のトランザクションによってすでに修正されているため、意図的ロックは取得されません。スナップショットトランザクションによってそのローを更新しようとしても失敗となり、スナップショットの更新競合エラーが返されます。

関連情報

スナップショットアイソレーション [766 ページ]
独立性レベル選択のガイドライン [802 ページ]

1.10.6.1.3 テーブルロック

テーブルロックは、トランザクションがテーブルを更新している間に、別のトランザクションが同じテーブルを更新するのを防ぎます。

テーブルロックには、共有、書き込みを意図、排他という 3 つの種類があります。テーブルロックは、トランザクションが COMMIT または ROLLBACK で終了したときに解放されます。

テーブルロックは、テーブルのスキーマにロックをかけるスキーマロックと異なり、テーブル内のすべてのローに対してロックをかけます。

次の表に、競合するテーブルロックの組合せを示します。

	共有	意図的	排他
共有		競合	競合
意図的	競合		競合
排他	競合	競合	競合

共有テーブルロック

共有テーブルロックでは、複数のトランザクションでベーステーブルのデータを読み込めます。ベーステーブルに共有テーブルロックを持つトランザクションは、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、テーブルを変更できます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

共有テーブルロックは、たとえば LOCK TABLE...IN SHARED MODE 文を実行することで取得できます。REFRESH MATERIALIZED VIEW 文と REFRESH TEXT INDEX 文も WITH SHARE MODE 句をサポートするため、再表示操作の実行中に、基本となるテーブルに共有テーブルロックを設定するために使用できます。

書き込みを意図したテーブルロック

書き込みを意図したテーブルロックは意図的テーブルロックとも呼ばれます。意図的テーブルロックは、トランザクションによってローの書き込みロックが最初に取得されるときに、暗黙的に取得されます。つまり、意図的テーブルロックは、ローの更新、挿入または削除時に取得されます。共有テーブルロックと同様に、意図的テーブルロックは、トランザクションが COMMIT または ROLLBACK で完了するまで保持されます。意図的テーブルロックは、共有テーブルロックや排他テーブルロックと競合しますが、他の意図的テーブルロックとは競合しません。

排他テーブルロック

排他テーブルロックは、新しいデータの挿入など、他のトランザクションがテーブルのスキーマまたはデータを変更するのを防ぎます。排他スキーマロックとは異なり、独立性レベル 0 で実行しているトランザクションは、テーブルロックが排他的に設定されているテーブルのローを読み込むことはできます。一度に 1 つのトランザクションだけが、テーブルに対して排他ロックをかけられます。排他テーブルロックは、その他のすべてのテーブルロックとローロックとの間で競合します。

LOAD TABLE 文を使用すると、排他テーブルロックを暗黙的に取得します。

排他テーブルロックは、LOCK TABLE...IN EXCLUSIVE MODE 文を使用することで、明示的に取得します。REFRESH MATERIALIZED VIEW 文と REFRESH TEXT INDEX 文も WITH EXCLUSIVE MODE 句を提供するため、再表示操作の実行中に、基本となるテーブルに排他テーブルロックを設定するために使用できます。

関連情報

スキーマロック [785 ページ]
独立性レベル選択のガイドライン [802 ページ]

1.10.6.1.4 位置ロック

位置ロックは、幻または幻ローが存在するための異常状態を避けるために設計された、キー範囲のロック形式です。

位置ロックが関係するのは、独立性レベル 3 で動作しているトランザクションをデータベースサーバが処理しているときのみです。

独立性レベル 3 で実行しているトランザクションは直列化可能です。独立性レベル 3 のトランザクションの動作は、他のトランザクションによって同時に発生する更新アクティビティの影響を受けないはずです。特に、独立性レベル 3 のトランザクションは、計算結果に影響を与える可能性のあるローが取り込まれる INSERT または UPDATE (つまり幻) による影響を受けるこ

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

とはありません。データベースサーバは位置ロックを使用して、そのような更新が発生することを防ぎます。位置ロックは、独立性レベル 2 (繰り返し可能読み出し) と独立性レベル 3 とを区別する追加のロック処理です。

幻ローが作成されないように、データベースサーバはテーブルの物理スキャン内で位置ロックを取得します。逐次スキャンの場合は、現在のローのロー識別子に基づいてスキャン位置が決定されます。インデックススキャンの場合は、現在のローのインデックスキー値に基づいてスキャン位置が決定されます (インデックスキー値は、ユニークな場合もそうでない場合もある)。トランザクションはスキャン位置をロックすることによって、他のトランザクションが、ローの順序付けに使用される特定範囲の値に関する挿入を行うことを防ぐことができます。この動作は、INSERT 文やインデックス付き属性の値を変更する UPDATE 文に適用されます。スキャン位置がロックされた場合、UPDATE 文は、直後に INSERT 要求が続く、インデックスエントリに対する DELETE 要求であると見なされます。

サポートされる位置ロックには、幻ロックと挿入ロックの 2 種類があります。どちらの種類も共有ロックであり、複数のトランザクションが同じローで同じ種類のロックを取得できます。ただし、幻ロックと対幻ロックは競合します。

幻ロック

幻ロックは対挿入ロックとも呼ばれ、その後で他のトランザクションによって幻ローが作成されないように、スキャン位置に設定されます。幻ロックが取得されると、テーブル内で、対挿入ロックがかかっているローの直前に、他のトランザクションがローを挿入することを防止します。幻ロックは長期間のロックで、トランザクションの終了まで保持されます。

幻ロックは、独立性レベル 3 で実行しているトランザクションのみが取得できます。独立性レベル 3 は、幻に関して一貫性を保証する唯一の独立性レベルです。

インデックススキャンでは、幻ロックはインデックスを介して読み込まれる各ローで取得されます。また、条件を満たすインデックス範囲の最後にインデックスが挿入されることを防ぐため、インデックススキャンの最後に幻ロックが 1 つ追加取得されます。インデックススキャンで幻ロックを使用すると、テーブルに新しいローが挿入されたり、インデックス付きの値 (幻ロックの対象となるポイントにインデックスエントリが作成される原因となる) が更新されたりすることが原因で、幻が作成されないようになります。

逐次スキャンでは、挿入処理によって結果セットが変更されないように、幻ロックはテーブルの行ごとに取得されます。独立性レベル 3 のスキャンにより、データベースの同時実行性が悪影響を受けることがあります。1 つ以上の幻ロックは挿入ロックと競合し、1 つ以上の読み込みロックは書き込みロックと競合しますが、幻ロック/挿入ロックと読み込みロック/書き込みロックの間に相互作用はありません。たとえば、書き込みロックは読み込みロックのかかったローにかけることはできませんが、幻ロックだけがかかったローにはかけることができます。この柔軟性のため、データベースサーバでは多くのオプションを利用できます。しかしそのために、幻ロックをかける場合は、読み込みロックの設定に特別な注意が要求されます。これを怠ると、他のトランザクションがローを削除してしまう可能性があります。

挿入ロック

挿入ロックは対幻ロックとも呼ばれ、ローを挿入する権利を確保するためにスキャン位置に設定される短期間のロックです。ロックは、その挿入の期間だけ保持されます。ローがデータベースページ内に正しく挿入されると、一貫性を確保するためにそのローは書き込みロックがかけられ、挿入ロックは解放されます。トランザクションがあるローに対して挿入ロックを設定すると、他のトランザクションはそのローに幻ロックを設定できません。データベースサーバは、新しい要求によって発生する可能性のある、アクティブな接続による独立性レベル 3 のスキャン操作を予期する必要があるため、挿入ロックが必要です。幻ロックと挿入ロックは、同じトランザクションによって保持される場合は相互に競合しません。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

独立性レベル選択のガイドライン [802 ページ]

1.10.6.2 ロックの競合

ロックの競合が発生するのは、あるトランザクションが、別のトランザクションがロックをかけているローに排他ロックをかけようとした場合や、別のトランザクションが排他ロックをかけているローに共有ロックをかけようとした場合です。

このような場合、あるトランザクションは「別のトランザクション」の完了を待たなければなりません。待機しなければならないトランザクションは、もう一方のトランザクションにブロックされます。

データベースサーバは、スキーマロック、ローロック、テーブルロック、位置ロックを必要に応じて使用し、必要な一貫性レベルを確保します。特定のロックの使用を明示的に要求する必要はありません。ただし、要件に最も合う独立性レベルを選択することで維持される一貫性レベルを管理する必要があります。ロックの種類を知っておくと、独立性レベルの選択、および各レベルのパフォーマンスへの影響を理解する上で便利です。1 つのトランザクションがロックを取得することで自分自身をブロックすることはできないことに注意してください。ロックの競合は、2 つ以上のトランザクション間でのみ発生します。

データベースサーバが、トランザクションの即時処理を禁止するロックの競合を認識すると、トランザクションの実行を一時停止するか、またはトランザクションを終了し、変更をロールバックし、エラーを返すことができます。ブロックオプションを設定して、その手段を制御します。ブロックをオンに設定すると、2 番目のトランザクションが待機します。

どのロックが競合するか

4 つのロックはそれぞれ特定の目的がありますが、すべての種類が干渉し合うため、トランザクション間でロックの競合が発生する原因となります。データベースの一貫性を確保するために、一度に 1 つのトランザクションだけが 1 つのローを変更できます。2 つのトランザクションが同時に 1 つの値を変更できてしまうと、1 つの値が 2 つの異なる値に変更されることになります。このため、ローの書き込みロックは排他ロックであることが重要です。これに対して、複数のトランザクションが 1 つのローを読んでも問題は生じません。ローを変更するわけではないので、競合することはありません。このため、ローの読み込みロックは多くの接続間で共有されていても構いません。

次の表に、競合するロックの組み合わせを示します。スキーマロックはローに適用されないため、含めてありません。

ローロック	readpk	read	intent	writenopk	write
readpk					競合
read				競合	競合
intent			競合	競合	競合
writenopk		競合	競合	競合	競合
write	競合	競合	競合	競合	競合

テーブルロック	共有	意図的	排他
共有		競合	競合

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

テーブルロック	共有	意図的	排他
意図的	競合		競合
排他	競合	競合	競合

位置ロック	幻	挿入
幻		競合
挿入	競合	

関連情報

独立性レベル選択のガイドライン [802 ページ]

1.10.6.3 クエリ時のロック

ユーザが SELECT 文を入力したときにデータベースサーバが使用するロックは、トランザクションの独立性レベルによって異なります。

すべての SELECT 文は、独立性レベルに関係なく、参照先テーブルで共有スキーマロックを取得します。

独立性レベル 0 の SELECT 文

独立性レベル 0 で SELECT 文を実行するときは、ロック操作は不要です。各トランザクションは他のトランザクションによる変更から保護されません。プログラマまたはデータベースユーザは、この制限を念頭においてこのようなクエリの結果を解釈する責任があります。

独立性レベル 1 の SELECT 文

独立性レベル 1 でトランザクションを実行するとき、データベースサーバは独立性レベル 0 で実行するときより数多くのロックは使用しません。各レベルでデータベースサーバのオペレーションが異なる点は、2 つしかありません。

オペレーションの最初の違いはロックの設定とは無関係で、むしろロックへの配慮に関するものです。独立性レベル 0 では、別のトランザクションが書き込みロックを取得しても、トランザクションはすべてのローを読み込むことができます。一方独立性レベル 1 のトランザクションは、各ローを読み込む前に書き込みロックがかかっているかをチェックします。書き込みロックがかかっているローは読み込むことができません。この場合、ダーティデータを読み込むことになるからです。READPAST ヒントを使用すると、サーバは書き込みロックがかかっているローを無視できます。ただし、トランザクションのブロックがなくなると、READPAST ヒントのセマンティックは独立性レベル 1 のセマンティックと一致しなくなります。

オペレーションの 2 番目の違いは、カーソル安定性に影響します。カーソル安定性は、カーソルの現在のローに短期間の読み込みロックを設定して達成されます。この読み込みロックはカーソルを移動すると解放されます。カーソルの内容がジョイン

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

の結果を示している場合は、複数のローが影響を受けます。この場合、データベースサーバはカーソルの現在のローに情報を提供したすべてのローに短期間の読み込みロックをかけ、カーソルの別のローが現在のローになるとこれらのロックを解放します。

独立性レベル 2 の SELECT 文

独立性レベル 2 では、データベースサーバはそのオペレーションを修正し、繰り返し可能読み出しのセマンティックを保証します。SELECT 文がテーブルのすべてのローから値を返す場合、データベースサーバはローを読み込むときに各ローに読み込みロックをかけます。一方、SELECT 文に WHERE 句、または結果のローを制限する他の条件が含まれている場合、データベースサーバは各ローを読み込み、ローの値が条件を満たしているかどうかをテストし、条件を満たすローに読み込みロックをかけます。取得される読み込みロックは、長期間の読み込みロックであり、暗黙的または明示的な COMMIT 文または ROLLBACK 文によってトランザクションが完了するまで保持されます。独立性レベル 1 と同じように、独立性レベル 2 ではカーソル安定性が保証され、ダーティリードは許可されません。

独立性レベル 3 の SELECT 文

独立性レベル 3 の処理では、データベースサーバはすべてのトランザクションスケジュールが直列化可能であることを確認する必要があります。特に、独立性レベル 2 での要件に加えて、同じ文を再実行するとすべての環境で同じ結果を返すことが保証されるように、幻ローを防ぐ必要があります。

この要件を満たすために、データベースサーバは読み込みロックと幻ロックを使用します。独立性レベル 3 で SELECT 文を実行すると、データベースサーバは結果セットの計算で処理される各ローで読み込みロックを取得します。こうすることで、トランザクションが完了するまで、他のトランザクションがそれらのローを修正できないようにします。

この要件は、データベースサーバが独立性レベル 2 で実行するオペレーションと似ていますが、これらのローが SELECT 文の WHERE 句、ON 句、または HAVING 句の述部を満たすかどうかに関係なく、読み込まれた各ローにロックをかけなければならない点が異なります。たとえば、販売部のすべての従業員名を選択する場合、サーバはトランザクションが独立性レベル 2 または 3 のどちらで実行されているかに関係なく、販売部の従業員に関する情報が含まれているすべてのローにロックをかける必要があります。ただし、独立性レベル 3 では、販売部に所属しない従業員のローにも読み込みロックをかける必要があります。そうしない場合、最初のトランザクションが実行されている間に、別のトランザクションが別の従業員を販売部に移動する可能性があります。

読み込まれた各ローに読み込みロックをかける必要がある場合、次の 2 つの影響があります。

- データベースサーバは、独立性レベル 2 よりも多くのロックをかける必要があります。取得される幻ロックの数は、スキャンで取得される読み込みロックの数よりも 1 多くなります。倍増したロックのオーバヘッドのために、要求の実行時間が長くなります。
- 各ローの読み込みで読み込みロックを取得すると、同じテーブルに対するデータベース更新オペレーションの同時実行性に悪影響があります。

データベースサーバが取得する幻ロックの数には大きな幅があり、クエリオプティマイザによって選択された実行方式によって異なります。SQL Anywhere クエリオプティマイザは、システム全体の同時実行性に悪影響を与える可能性があるため、独立性レベル 3 での逐次スキャンを回避しようとします。しかし、このようなオプティマイザの機能は、文の述部と、参照先テーブルで利用できる適切なインデックスに依存します。

たとえば、Employee ID 123 の従業員に関する情報を選択したいとします。EmployeeID は従業員テーブルのプライマリキーであるため、ローを効率的に検索するために、クエリオプティマイザがプライマリキーインデックスを使用するインデックス方式

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

を選択しようとするのはほぼ確実です。さらに、プライマリキーの値はユニークであるため、別のトランザクションが他の EmployeeID を 123 に変更する危険性もありません。サーバは、従業員 123 に関する情報を含むローに読み込みロックをかけるだけで、別の従業員にその ID 番号が割り当てられることを防止できます。

一方、販売部の全従業員を選択する場合は、読み込みロック以外のロックも取得する必要があります。適切なインデックスがないため、データベースサーバは従業員テーブルの各ローを読み込み、各従業員が販売部に所属するかどうかをテストする必要があります。この場合は、テーブルの各ローに読み込みロックと幻ロックの両方を設定する必要があります。

SELECT 文とスナップショットアイソレーション

snapshot、statement-snapshot、または readonly-statement-snapshot で実行される SELECT 文は、読み込みロックを取得しません。これは、各スナップショットトランザクション (または文) は、以前のある時点における、コミットされた状態のデータベースのスナップショットを認識するためです。この特定の時点は、3 種類あるスナップショットアイソレーションのレベルのうちどれが文で使用されるかによって決まります。つまり、読み込みトランザクションが更新トランザクションをブロックしたり、更新トランザクションが読み込みトランザクションをブロックしたりすることはありません。そのため、スナップショットアイソレーションを使用すると、一貫性という明白な長所だけでなく、同時実行性という重要な長所を得ることができます。ただし、トレードオフとして、スナップショットアイソレーションは非常にコストがかかることがあります。これは、スナップショットアイソレーションの一貫性保証では、同時に実行される他のトランザクションのために、変更されたローのコピーを保存、追跡、(最終的に) 削除する必要があるためです。

関連情報

独立性レベル選択のガイドライン [802 ページ]

1.10.6.4 挿入時のロック

挿入操作によって新しいローが作成されます。データベースサーバは挿入時にさまざまなタイプのロックを利用して、データの整合性を保証します。

どの独立性レベルであっても実行している INSERT 文では、次の一連の操作手順が発生します。

1. テーブルで共有スキーマロックを保持していない場合は、取得します。
2. テーブルで書き込みを意図したテーブルロックを保持していない場合は、取得します。
3. 新しいローを格納するために、ページでロックされていない位置を検索します。ロック競合を最小限に抑えるために、データベースサーバは、削除された (しかし、まだコミットされていない) ローによって利用可能になった領域をただちに再利用しません。新しいローを確保するために、新しいページがテーブルに割り当てられることがあります。また、データベースファイルのサイズが増大することがあります。
4. 新しいローに値を入れます。
5. ローを追加するテーブルに挿入ロックをかけます。挿入ロックは排他ロックであるため、一度挿入ロックをかけると、独立性レベル 3 の他のトランザクションは、幻ロックをかけて挿入をブロックすることができません。
6. 新しいローに書き込みロックをかけます。書き込みロックが取得されると、挿入ロックが解放されます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

7. テーブルにローを挿入します。ここで、独立性レベル 3 の他のトランザクションは初めて新しいローの存在に気が付きます。ただし、すでに書き込みロックがかかっているため、これらのトランザクションはそのローの修正や削除はできません。
8. 影響を受けるすべてのインデックスを更新し、必要に応じてユニークであることを確認します。プライマリキーの値はユニークである必要があります。他のカラムもユニークな値だけを含むように定義される場合があります。このようなカラムが存在する場合は、ユニーク性が検証されます。
9. テーブルが外部テーブルである場合は、プライマリテーブルの共有スキーマロックを保持していなければ取得し、挿入される外部キーカラムの値が NULL でない場合は、プライマリテーブルの一致するプライマリローで読み込みロックを取得します。データベースサーバは、挿入トランザクションで COMMIT が実行されたときにプライマリローが存在していることを保証する必要があります。この確認は、プライマリローの読み込みロックを取得して行います。読み込みロックをかけても他のトランザクションは自由にそのローを読むことができますが、削除や更新はできません。
   対応するプライマリローが存在しない場合は、参照整合性制約違反が発生します。

最後の手順の後、テーブルで定義された AFTER INSERT トリガが起動します。トリガ内の処理におけるロック動作は、アプリケーションの場合と同じです。トランザクションがコミット (すべての参照整合性制約が満たされる) またはロールバックされると、すべての長期間ロックが解放されます。

ユニーク性

特定のカラムまたはカラムの組み合わせに設定される値のすべてをユニークにすることができます。ユーザがあえて作成しなくても、データベースサーバがそのユニークなカラムに対するインデックスを作成し、それによって値のユニーク性を保証しています。

特に、プライマリキーの値はすべてユニークである必要があります。データベースサーバは、すべてのテーブルのプライマリキーに対するインデックスを自動的に作成します。プライマリキーに対するインデックスを作成するようデータベースサーバに要求しないでください。これをすると、重複するインデックスが作成されてしまいます。

オーファンと参照整合性

外部キーは、通常別のテーブルにあるプライマリキーまたは一意性制約を参照します。そのプライマリキーが存在しない場合、問題の外部キーはオーファンと呼ばれます。データベースサーバは、参照整合性に違反するローがデータベースに含まれていないことを自動的に確認します。このプロセスを参照整合性の検証と呼びます。データベースサーバは、オーファンの数をカウントすることで、参照整合性を検証します。

wait_for_commit

データベースサーバに対し、トランザクションの終了まで参照整合性の検証を遅延するように指示できます。このモードでは、外部キーを含む 1 つのローを挿入し、次にプライマリキーを持たないプライマリローを挿入できます。この両方のオペレーションは同じトランザクションで実行する必要があります。

データベースサーバがコミット時間まで参照整合性の検証を遅延するように要求するには、wait_for_commit オプションを On に設定します。デフォルトでは、このオプションは Off に設定されます。ON にするには、次の文を実行します。

```
SET OPTION wait_for_commit = On;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

新しい外部キー値が挿入されるときにサーバが一致するプライマリローを見つけられず、wait_for_commit が On の場合、サーバはオーファンとして挿入を許可します。孤立した外部ローの場合は、挿入時に次の手順が発生します。

- サーバは、プライマリテーブルで共有スキーマロックを保持していない場合は取得します。また、プライマリテーブルで書き込みを意図したロックを取得します。
- サーバは、プライマリテーブルに代理ローを挿入します。実際のローはプライマリテーブルに挿入されません。ただし、サーバはロックをかけるためにそのローのユニークなロー識別子を作成し、この代理ローで書き込みロックを取得します。次に、サーバはプライマリテーブルのプライマリキーインデックスに適切な値を挿入します。

トランザクションをコミットする前に、データベースサーバはトランザクションが作成したオーファン数をチェックして参照整合性が維持されていることを確認します。各トランザクションの終了時に、この数は 0 になっていなければなりません。

1.10.6.5 更新時のロック

データベースサーバは、ロックを使用するときに特定のレコードに含まれる情報を修正します。

挿入時と同様に、独立性レベルに関係なく、すべてのトランザクションで次の操作手順が発生します。

1. テーブルで共有スキーマロックを保持していない場合は、取得します。
2. 更新される各テーブルで書き込みを意図したテーブルロックを保持していない場合は、取得します。
    1. 更新されるテーブルごとに、テーブルにトリガがある場合は、必要に応じて OLD 値と NEW 値のテンポラリテーブルを作成します。
    2. 更新の候補となるローを識別します。ローがスキャンされている間は、ローはロックされます。
       独立性レベル 2 と 3 では、デフォルトのロック動作とは異なる次のような違いが発生します。読み込みロックではなく書き込みを意図したローレベルのロックが取得されます。また、書き込みを意図したロックは、更新の候補としては最終的には拒否されたローで取得される場合があります。
    3. 手順 2.a で識別された候補となる各ローは、残りのシーケンスに従います。
3. 影響を受けるローに書き込みロックをかけます。
4. UPDATE 文により、影響を受けるそれぞれのカラムの値を更新します。
5. インデックス付きの値を変更した場合は、新しいインデックスエントリを追加します。ローの元のインデックスエントリは残りますが、削除済みのマークが付けられます。短期間の挿入ロックが保持されている間に、新しい値の新しいインデックスエントリが挿入されます。サーバは、必要に応じてインデックスのユニーク性を検証します。
6. 一意性違反が発生した場合は、ローの古い値と新しい値を格納するためにテンポラリ「保持」テーブルが作成されます。古い値と新しい値は保持テーブルにコピーされ、ベーステーブルのローは削除されます。DELETE トリガは起動されません。ローごとの処理が終わるまで、手順 7 〜 9 は行いません。
7. ローの外部キー値が変更された場合、プライマリテーブルで共有スキーマロックを取得し、新しい外部キー値を挿入する手順に従います。
   同様に、必要に応じて wait_for_commit の手順に従います。
8. テーブルが参照整合性関係においてプライマリテーブルであり、かつ関係の UPDATE アクションが RESTRICT でない場合、外部テーブル (間) で共有スキーマロックを、各外部テーブルで書き込みを意図したテーブルロックをそれぞれ取得することで、外部テーブル内で影響のあるローを特定し、次に影響のあるすべてのローで書き込みロックを取得して、必要に応じてそれぞれのローを修正します。このプロセスは、参照整合性制約のネストした階層でカスケードされることがあります。
9. 必要に応じて AFTER ROW トリガを起動します。

保持テンポラリテーブルが必要だった場合には、最後の手順の後で、保持テンポラリテーブル内の各ローはベーステーブルに挿入されます (ただし、INSERT トリガは起動されません)。ローの挿入が成功した場合は、上記の手順 7 〜 9 が実行され、古いロー値と新しいロー値は OLD および NEW テンポラリテーブルにコピーされて、AFTER STATEMENT UPDATE トリ

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

ガによって、変更されたすべてのローが正しく処理されます。保持されたローがすべて処理された後、AFTER STATEMENT UPDATE トリガが順番に起動されます。COMMIT の実行時、サーバはこのトランザクションで生成されたオーファンの数が 0 であることを確認することで参照整合性を検証し、次にすべてのロックを解放します。

カラムの値を変更するために、多くの操作が必要になることがあります。データベースサーバが実行する作業量は、修正するカラムがプライマリキーまたは外部キーの一部でない場合は大幅に少なくなります。カラムをユニークと宣言したために、変更する値が明示的または暗黙的にインデックスに含まれていない場合も作業量は少なくて済みます。

UPDATE オペレーション中に参照整合性を確認するオペレーションは、INSERT 中に確認する場合よりも複雑になります。実際、プライマリキーの値を変更すると、オーファンが作成されることがあります。置換値を挿入すると、データベースサーバはもう一度オーファンをチェックしなければなりません。

関連情報

挿入時のロック [793 ページ]
独立性レベルと一貫性 [763 ページ]
独立性レベル選択のガイドライン [802 ページ]

1.10.6.6 削除時のロック

DELETE オペレーションは、INSERT オペレーションとほとんど同じ手順を実行しますが、その順序は反対になります。

挿入時や更新時と同様に、独立性レベルに関係なくすべてのトランザクションで次の操作手順が発生します。

1. テーブルで共有スキーマロックを保持していない場合は、取得します。
2. テーブルで書き込みを意図したテーブルロックを保持していない場合は、取得します。
    1. 更新の候補となるローを識別します。ローがスキャンされている間は、ローはロックされます。
       独立性レベル 2 と 3 では、デフォルトのロック動作とは異なる次のような違いが発生します。読み込みロックではなく書き込みを意図したローレベルのロックが取得されます。また、書き込みを意図したロックは、更新の候補としては最終的には拒否されたローで取得される場合があります。
    2. 手順 2.a で識別された候補となる各ローは、残りのシーケンスに従います。
3. 削除するローに書き込みロックをかけます。
4. 他のトランザクションから見えなくなるように、ローをテーブルから削除します。ローは、トランザクションがコミットされるまで破壊できません。ローを破壊すると、トランザクションをロールバックするオプションが削除されてしまうからです。削除されたローのインデックスエントリはトランザクションの完了まで残りますが、削除済みのマークが付けられます。これにより、他のトランザクションは同じローを再挿入できません。
5. テーブルが参照整合性関係においてプライマリテーブルであり、かつ関係の DELETE アクションが RESTRICT でない場合、外部テーブル (間) で共有スキーマロックを、各外部テーブルで書き込みを意図したテーブルロックをそれぞれ取得することで、外部テーブル内で影響のあるローを特定し、次に影響のあるすべてのローで書き込みロックを取得して、必要に応じてそれぞれのローを修正します。このプロセスは、参照整合性制約のネストした階層でカスケードされることがあります。

参照整合性に違反しない場合は、トランザクションをコミットできます。参照整合性を調べるために、データベースサーバは削除によって作成されたオーファンを追跡します。COMMIT の実行時、サーバはオペレーションをトランザクションログファイルに記録し、すべてのロックを解放します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

独立性レベルと一貫性 [763 ページ]
独立性レベル選択のガイドライン [802 ページ]

1.10.6.7 ロック期間

ロックは、通常、トランザクションが完了するまでトランザクションによって保持されます。

この動作によって、他のトランザクションが変更を加えたことによって、元のトランザクションがロールバックできなくなることが回避されます。独立性レベル 3 では、トランザクションの直列可能性を保障するために、トランザクションが終了するまですべてのロックが保持されます。

トランザクションの終了まで保持されないロックは、カーソル安定性ロックのみです。このローロックは、対象のローがカーソルの現在のローである間、保持されます。多くの場合、カーソル安定性ロックの継続時間はトランザクションの存続時間よりも短くなりますが、WITH HOLD カーソルの場合、接続が存続する間、カーソル安定性ロックが保持されることがあります。

1.10.6.8 ロックの表示 (SQL Central の場合)

SQL Central のロックタブを使用して、データベースに現在保持されているロックを表示します。

コンテキスト

ロックされているローの内容は、データベースのロックの問題の診断に使用できます。

手順

1. 左のウィンドウ枠でデータベースをクリックします。
2. 右ウィンドウ枠で、ロックタブをクリックします。

結果

このタブに、ロックに関する情報が表示されます。

SQL Central では、クエリが開始された時点でロックの存在を表示します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

スキーマロック [785 ページ]

1.10.6.9 ロックの表示 (Interactive SQL)

ロックされているローの情報を取得して、データベースのロックの問題を診断します。

前提条件

sa_locks、sa_conn_info、および connection_properties に対する EXECUTE 権限が必要です。MONITOR システム権限および、SERVER OPERATOR または DROP CONNECTION システム権限のいずれかが必要です。

コンテキスト

ロック、ロックの期間、ロックの種類に関する情報を含め、接続で保持しているロックを表示します。

手順

1. Interactive SQL でデータベースに接続します。
2. ロックビューアウィンドウを使用します。 ツール ロックビューア をクリックします。

結果

Interactive SQL では、接続で保持しているロック、ロックされているオブジェクト、およびその結果としてブロックされている接続を表示します。

この情報は、ステータスバーで確認することもできます。ステータスバーのインジケータに、選択したタブのステータス情報が表示されます。

関連情報

トランザクションのブロックとデッドロック [779 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.6.10 ミューテックスおよびセマフォ

アプリケーションロジックのミューテックスおよびセマフォを使用して、ロック動作を実現し、リソースの可用性を制御および伝達します。

ミューテックスおよびセマフォとは、外部ライブラリやプロシージャなどの共有リソースの可用性や使用を制御するロックメカニズムです。ミューテックスおよびセマフォをインクルードすれば、アプリケーションが必要とする種類のロック動作を実現できます。アプリケーションの要件に従って、ミューテックスまたはセマフォを使用するかどうかを選択します。

ミューテックスによって、アプリケーションに同時実行制御メカニズムが備わります。たとえば、一度に行う接続を 1 つに制限して、ストアドプロシージャ、ユーザ定義関数、トリガ、またはイベントのクリティカルセクションを実行するためにミューテックスを使用することができます。ミューテックスによって、データベースオブジェクトに直接対応していないアプリケーションリソースをロックすることもできます。セマフォによって、データベースのプロデューサ/コンシューマアプリケーションや、制限されたアプリケーションリソースへのアクセスに対するサポートが提供されます。

ミューテックスとセマフォは、データベースのローロックやテーブルロックの場合と同じデッドロック検出を利用できます。

ミューテックスのロックと解除、セマフォの通知と待機を行うには UPDATE ANY MUTEX SEMAPHORE、作成と置換を行うには CREATE ANY MUTEX SEMAPHORE、削除と置換を行うには DROP ANY MUTEX SEMAPHORE が必要です。ミューテックスまたはセマフォの更新が可能なユーザに関してより細かいレベル調整をするには、ユーザに対してではなく、ミューテックスやセマフォが使用されているオブジェクトに対して権限を付与することができます。たとえば、ミューテックスを含むシステムプロシージャに対して、EXECUTE 権限を付与することが可能です。

ミューテックスに関する詳細

ミューテックスはロックおよび解除のメカニズムであり、外部ライブラリやストアドプロシージャなどの、共有リソースのクリティカルセクションの可用性を制限します。LOCK MUTEX 文および RELEASE MUTEX 文を実行することで、それぞれ、ミューテックスのロックおよび解除が可能です。

ミューテックスのスコープは、トランザクションまたは接続のいずれかです。トランザクションスコープミューテックスでは、ミューテックスをロックしているトランザクションが終了するまでロックが保持されます。接続スコープミューテックスでは、RELEASE MUTEX 文が接続によって実行されるまで、または接続が終了するまでロックが保持されます。

ミューテックスのモードは、排他または共有のいずれかです。排他モードでは、リソースを使用することができるのは、ロックを保持しているトランザクションまたは接続のみです。共有モードでは、複数のトランザクションまたは接続がミューテックスをロックできます。

ミューテックスは再帰的にロックすることができます (つまり、コード内で同じミューテックスに対する LOCK MUTEX 文をネストできます)。ただし、接続スコープミューテックスでは、ミューテックスを解除するために同数の RELEASE MUTEX 文が必要です。

共有モードで接続がミューテックスをロックし、その後排他モードで (再帰的に) 再びロックする場合、ロックが 2 回解除されるまで、またはトランザクションの終了まで、排他モードでロックが保持されます。

ミューテックスを使用して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方法を示す簡単なシナリオを、以下にご紹介します。このシナリオでは、一度に 1 つの接続によってクリティカルセクションを実行することができます (ただし、複数のトランザクションにまたがる場合があります)。

1. 次の文により新しいミューテックスが作成され、クリティカルセクションを保護します。

   ```
   CREATE MUTEX protect_my_cr_section SCOPE CONNECTION;
   ```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. 次の文により、他の接続がアクセスできないように、排他モードでクリティカルセクションをロックします。

```
LOCK MUTEX protect_my_cr_section IN EXCLUSIVE MODE;
```

3. 次の文により、クリティカルセクションを解除します。

```
RELEASE MUTEX protect_my_cr_section;
```

4. クリティカルセクションの保護が不要になった場合、次の文でミューテックスを削除します。

```
DROP MUTEX protect_my_cr_section;
```

セマフォに関する詳細

セマフォとは通知メカニズムであり、カウンタを使用してリソースの可用性を伝達します。NOTIFY SEMAPHORE 文および WAITFOR SEMAPHORE 文を実行することで、それぞれ、セマフォカウンタの増分および減分が可能です。リソース可用性モデル、またはプロデューサ - コンシューマモデルでセマフォを使用します。いずれのモデルでも、セマフォの値は 0 を下回ることができません。このようにして、カウンタを使用してリソースの可用性 (この例ではライセンス) を制限します。

リソース可用性モデルとは、カウンタを使用してリソースの可用性を制限する場合です。たとえば、アプリケーションの使用を一度に 10 ユーザに制限するライセンスがあるとします。START WITH 句を使用して、セマフォカウンタを作成時刻で 10 に設定します。ユーザがログインし、WAITFOR SEMAPHORE 文が実行されると、カウントが 1 ずつ減分されます。カウントが 0 になると、ユーザは指定のタイムアウト時間まで待機します。タイムアウト前にカウントが 0 から増えると、ログインします。カウントが増えない場合は、ユーザログイン試行がタイムアウトします。ユーザがログアウトし、NOTIFY SEMAPHORE 文が実行されると、カウントが 1 ずつ増分されます。ユーザがログインするたびにカウントが減分され、ログアウトするたびにカウントが増分されます。

プロデューサ - コンシューマモデルとは、カウンタを使用してリソースの可用性を伝達する場合です。たとえば、別のプロシージャが生成するものを消費するプロシージャがあるとします。このコンシューマが WAITFOR SEMAPHORE 文を実行し、処理対象を待機します。プロデューサが出力を行うと、NOTIFY SEMAPHORE 文を実行して、後続の処理が可能であることを通知します。この文では、セマフォに関連付けられたカウンタを増分します。コンシューマの処理を待機しているときは、カウンタは減分されます。プロデューサ - コンシューマモデルでは、カウンタは 0 を下回ることはできませんが、プロデューサが必要なだけカウンタを増分することができます。

セマフォを使用してアプリケーションのライセンス数を制御する方法を示す簡単なシナリオを、以下にご紹介します。このシナリオでは、合計 3 つのライセンスが使用可能で、アプリケーションに正常にログインするごとにライセンスを 1 つ消費するとします。

1. 次の文により、初期カウントとして指定されたライセンス数で、新しいセマフォを作成します。

```
CREATE SEMAPHORE license_counter START WITH 3;
```

2. 次の文により、license_counter セマフォを使用してライセンスを取得します。

```
WAITFOR SEMAPHORE license_counter;
```

3. 次の文により、ライセンスを解除します。

```
NOTIFY SEMAPHORE license_counter INCREMENT BY 1;
```

4. ライセンスによるアプリケーションの制限が不要になった場合、次の文でセマフォを削除します。

```
DROP SEMAPHORE license_counter;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

プロデューサ - コンシューマモデルでセマフォを使用する一般的な方法は、以下のようになります。

```
CREATE SEMAPHORE producer_counter;
CREATE SEMAPHORE consumer_counter START WITH 100;
CREATE PROCEDURE DBA.MyProducer( )
   BEGIN
      WHILE 1 = 1 LOOP
          WAITFOR SEMAPHORE consumer_counter;
          -- produce some data and put it somewhere (e.g. a table)
          NOTIFY SEMAPHORE producer_counter;
      END LOOP
   END
go
CREATE PROCEDURE DBA.MyConsumer( )
   BEGIN
      WHILE 1 = 1 LOOP
          WAITFOR SEMAPHORE producer_counter;
          -- read the next bit of data and do something with it
          NOTIFY SEMAPHORE consumer_counter;
      END LOOP
   END
go
```

この例では、MyProducer と MyConsumer は異なる接続で実行されています。MyProducer はデータをフェッチし、MyConsumer より先に最大 100 回反復することができます。MyConsumer が MyProducer より速い場合、producer_counter は最終的に 0 となります。0 になった時点で、MyProducer がさらにデータを取得するまで、MyConsumer がブロックされます。MyProducer が MyConsumer より速い場合、consumer_counter は最終的に 0 となります。0 になった時点で、MyConsumer がさらにデータを取得するまで、MyProducer がブロックされます。

このセクションの内容:

ミューテックスおよびセマフォの作成 (SQL Central の場合) [801 ページ]
アプリケーション内のミューテックスまたはセマフォを使用してロック動作を実現し、またリソースの可用性を制御および伝達します。

1.10.6.10.1 ミューテックスおよびセマフォの作成 (SQL Central の場合)

アプリケーション内のミューテックスまたはセマフォを使用してロック動作を実現し、またリソースの可用性を制御および伝達します。

前提条件

CREATE ANY MUTEX SEMAPHORE システム権限が必要です。

コンテキスト

ミューテックスおよびセマフォをインクルードして、アプリケーションに必要なロック動作の種類を実現します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. 左ウィンドウ枠で、ミューテックスとセマフォを右クリックし、新規をクリック、次にミューテックスまたはセマフォのいずれかをクリックします。
2. ウィザードの指示に従います。

結果

ミューテックスまたはセマフォが作成されます。

次のステップ

ミューテックスの場合、LOCK MUTEX 文および RELEASE MUTEX 文を実行して、外部ライブラリやストアドプロシージャなどの、コードまたは共有リソースのクリティカルセクションの可用性を制限します。

セマフォの場合、WAITFOR SEMAPHORE 文または NOTIFY SEMAPHORE 文を実行して、ライセンスなどのリソースの可用性を制限します。

関連情報

ミューテックスおよびセマフォ [799 ページ]

1.10.7 独立性レベル選択のガイドライン

独立性レベルの選択は、アプリケーションが実行するタスクの種類によって異なります。

適切な独立性レベルを選択するには、一貫性と正当性のニーズと、同時に実行するトランザクションが妨げられずに処理を行うためのニーズのバランスを取る必要があります。トランザクションが 1 つのテーブルで 1 つまたは 2 つの特定の値しか使用しない場合は、数多くの大きなテーブルを検索するために数多くのローまたはテーブル全体をロックする必要があり、処理完了にきわめて長い時間がかかるプロセスに比べると、他のプロセスとの干渉が発生する可能性はかなり少なくなります。

たとえば、トランザクションで銀行口座間の資金移動を行う場合、確実に正確な情報が返されるようにする必要があるでしょう。ただし、休止中の口座の割合の概算を計算する場合は、そのトランザクションが他のトランザクションを待つかどうかを配慮しません。また、データベースの他のユーザを妨げないようにするために、多少の精度を犠牲にします。

さらに、金銭の振り込みは、2 つの口座の残高を含む 2 つのローだけに影響するのに対して、概算を計算するためにはすべての口座を読み込む必要があります。このため、金銭の振り込みによって他のトランザクションを遅らせる可能性は少なくなります。

レベル 0、1、2、3 の 4 つの独立性レベルがあります。レベル 3 は完全な独立性を提供し、トランザクションはスケジュールが直列化可能となる方法でインターリーブされます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

データベースでスナップショットアイソレーションを有効にした場合は、さらに snapshot、statement-snapshot、readonly-statement-snapshot という 3 つの独立性レベルが利用可能になります。

このセクションの内容:

スナップショットアイソレーションのレベルの選択 [803 ページ]
スナップショットアイソレーションには、同時実行性と一貫性の両方に利点があります。

直列化可能なスケジュール [804 ページ]
各種トランザクションのコンポーネントオペレーションをインターリーブする順序を、スケジュールと呼びます。

各種独立性レベルでの典型的なトランザクション [805 ページ]
独立性レベルは、データベースサーバによって実行されるタスクの種類に応じて設定する必要があります。

独立性レベル 2 と 3 での同時実行性の改善 [806 ページ]
独立性レベル 2 と 3 では数多くのロックが使用されます。これらの独立性レベルを常用するデータベースでは、精密な設計が重要です。

ロックの影響を削減するヒント [806 ページ]
同時に実行される他のトランザクションに影響を及ぼす可能性のある多数のロックの設定を回避するには、トランザクションを独立性レベル 3 で実行しないようにします。

関連情報

スナップショットアイソレーション [766 ページ]

1.10.7.1 スナップショットアイソレーションのレベルの選択

スナップショットアイソレーションには、同時実行性と一貫性の両方に利点があります。

スナップショットアイソレーションを使用すると、古いバージョンのローは実行中のトランザクションで必要とされる可能性があるかぎり保存されるため、コストがかかってしまいます。そのため、スナップショットを長期間実行すると、大量の古いローバージョンを格納する必要がある可能性があります。通常、statement-snapshot で使用されるスナップショットは、snapshot で使用されるスナップショットほど長くは存続しません。そのため、statement-snapshot の方が、snapshoto よりも一貫性は低くなりますが (トランザクション内の文ごとに、異なる時点のデータベースを認識する)、使用する領域の点で強みがあります。

ほとんどの場合は、snapshot 独立性レベルをお奨めします。これにより、トランザクション全体のデータベースに関する単一ビューが表示されるためです。

statement-snapshot 独立性レベルを使用すると、データの一貫性は低くなりますが、トランザクションを長時間実行したためにバージョン情報を格納するテンポラリファイルのサイズが大きくなりすぎる場合には有益です。

readonly-statement-snapshot 独立性レベルを使用すると、statement-snapshot よりも一貫性は低くなりますが、更新の競合が発生する可能性はなくなります。このため、この属性は元々異なる独立性レベルで実行することを想定していたアプリケーションを移植するのに最も適しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

スナップショットアイソレーション [766 ページ]

1.10.7.2 直列化可能なスケジュール

各種トランザクションのコンポーネントオペレーションをインターリーブする順序を、スケジュールと呼びます。

トランザクションを同時に処理するには、データベースサーバは 1 つのトランザクションでいくつかのコンポーネント文を実行し、次に他のトランザクションのコンポーネント文を実行してから、最初のトランザクションのオペレーション処理を続行する必要があります。

この方法でトランザクションを同時に適用すると、前の項で説明した一貫性が失われる 3 つのケースを含む、さまざまな結果が生じる可能性があります。トランザクションが順次実行された場合、つまり 1 つのトランザクションが完全に終了した後、次のトランザクションが開始された場合、データベースの最終状態が達成されることがあります。トランザクションをある順序で順次実行した結果、データベースが実際のスケジュールと同じ状態になった場合、そのスケジュールは直列化可能であるといいます。

直列化可能性は、一般に正当性の基準として受け入れられています。直列化可能なスケジュールは、データベースがトランザクションの同時実行により影響を受けないため、正しいスケジュールとして受け入れられます。

トランザクションの直列化可能性は、独立性レベルの影響を受けます。独立性レベル 3 では、すべてのスケジュールは直列化可能です。デフォルト設定値は 0 です。

直列化可能とは、同時実行性が影響を与えないということ

トランザクションが順次実行されるときでも、データベースの最終状態はこれらのトランザクションが実行される順序によって異なります。たとえば、1 つのトランザクションが特定のセルに値 5 を設定し、別のトランザクションが同じセルに 6 を設定すると、そのセルの最終値は最後に実行されたトランザクションによって決まります。

スケジュールが直列化可能であることがわかっても、トランザクションを最も効率よく実行する順序が決まるわけではありません。同時実行性の影響がないというだけです。トランザクションセットをある順序で順次実行することによって達成される結果は、すべて正しいと見なされます。

直列化不可能なスケジュールは矛盾を生じさせる

矛盾は、スケジュールが直列化可能でないときに生じる典型的な問題です。いずれのケースでも、すべてのトランザクションが順次実行された場合には起こり得ないような結果を生じさせる方法で文がインターリーブされたため、一貫性が失われました。たとえば、あるトランザクションがあるローにデータを挿入または更新しているときに、別のトランザクションがそのローを選択できる場合、ダーティリードが発生します。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

典型的な矛盾のケース [772 ページ]

1.10.7.3 各種独立性レベルでの典型的なトランザクション

独立性レベルは、データベースサーバによって実行されるタスクの種類に応じて設定する必要があります。

下記の情報を参考にして、特定の操作に最も合うレベルを判断してください。

典型的なレベル 0 トランザクション

データのブラウズや入力を伴うトランザクションは、終了するのに何分もかかり、相当数のローを読み込みます。独立性レベルが 2 または 3 の場合、同時実行性が犠牲になる可能性があります。この種のトランザクションには、通常、レベル 0 または 1 が使われます。

たとえば、データベースから大量のデータを読み込んで統計的にまとめる作業を行う意思決定支援アプリケーションは、後で修正されるローを多少読み込んでも大きな影響は受けません。このようなアプリケーションに上位レベルの独立性を要求すると、大量のデータに読み込みロックをかけてしまい、他のアプリケーションが書き込みアクセスできなくなります。

典型的なレベル 1 トランザクション

独立性レベル 1 は、カーソルとともに使用すると便利です。この 2 つを組み合わせると、ロック要件をそれほど増大せずにカーソルの安定性を保てます。データベースサーバは、カーソルの現在のローにかけられた読み込みロックを早期に解放することで、この利点を達成します。読み込みロックは、レベル 2 または 3 では繰り返し可能読み出しを保証するために、トランザクションが終了するまで維持する必要があります。

たとえば、カーソルを使用して在庫レベルを更新するトランザクションには独立性レベル 1 が適しています。なぜなら、品目の入荷や販売があるたびに在庫レベルを調整した内容が失われることがなく、このように頻繁に調整しても、他のトランザクションへの影響が最小限で済むからです。

典型的なレベル 2 トランザクション

独立性レベル 2 では、条件を満たすローは他のトランザクションが変更することはできません。このレベルは、ローを複数回読み込み、最初の結果セットに含まれるローが変更されないことを保証する必要があるときに使用します。

比較的多数の読み込みロックが必要となるため、この独立性レベルの使用には注意を要します。レベル 3 のトランザクションと同様に、データベースとインデックスを慎重に設計することで、ロック数も減り、データベースのパフォーマンスを向上させることができます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

典型的なレベル 3 トランザクション

独立性レベル 3 はセキュリティを最も重んじるトランザクションに適しています。幻ローを防ぐと、新しいローがオペレーションの途中で追加されて結果が壊される心配をせずに、マルチステップオペレーションを実行できます。

独立性レベル 3 がいかに高度の整合性を提供するとしても、同時に多数のトランザクションの実行をサポートする必要がある大きなシステムでは使用を控える必要があります。データベースサーバはこのレベルでは他のレベルよりも多くのロックを設定するため、1 つのトランザクションが他の多くのトランザクションの処理を妨げる可能性があります。

1.10.7.4 独立性レベル 2 と 3 での同時実行性の改善

独立性レベル 2 と 3 では数多くのロックが使用されます。これらの独立性レベルを常用するデータベースでは、精密な設計が重要です。

直列化可能なトランザクションを利用する必要がある場合、データベース、特にインデックスに関しては、プロジェクトのビジネスルールを念頭において設計することが重要です。大きなトランザクションを小さく分割すると、ローがロックされる時間も短縮し、パフォーマンスが向上します。

直列化可能なトランザクションは、他のトランザクションをブロックする可能性が最も大きくなりますが、必ずしも効率が低下するわけではありません。これらのトランザクションを処理する場合、データベースサーバは、ロック数が増加してもパフォーマンスは向上するという特定の最適化を実行できます。たとえば、ローが検索条件に一致するかどうかに関係なく、すべてのローには読み込みロックがかけられるため、データベースサーバはローの読み込みとロックの設定のオペレーションを自由に組み合わせて実行することができます。

1.10.7.5 ロックの影響を削減するヒント

同時に実行される他のトランザクションに影響を及ぼす可能性のある多数のロックの設定を回避するには、トランザクションを独立性レベル 3 で実行しないようにします。

オペレーションの性質上、独立性レベル 3 で実行する必要がある場合は、読み込むローやインデックスエントリの数をできるだけ少なくするようにクエリを設計することで、同時実行性への影響を軽減できます。これによって、レベル 3 のトランザクションの実行速度が向上するほか、さらに重要なこととして、設定するロックの数を減らすことができます。

独立性レベル 3 で実行するオペレーションが 1 つでもある場合は、インデックスを追加することでトランザクションの速度が向上する場合があります。インデックスには次の 2 つの利点があります。

- インデックスの使用により、ローを効率良く見つけることができます。
- 検索にインデックスを使用するとロック数が少なくて済みます。

関連情報

ロックの仕組み [782 ページ]

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.8 独立性レベルに関するチュートリアル

それぞれの独立性レベルは異なる動作を実行し、データベースとそこで実行する操作に応じて、使用するレベルを決定する必要があります。

以下のチュートリアルは、それぞれのタスクにどの独立性レベルを設定するかを決定するのに役立ちます。

このセクションの内容:

チュートリアル:独立性レベルのチュートリアルのシナリオ設定 [807 ページ]
Sales Manager と Accountant として動作する 2 つの Interactive SQL ウィンドウを開いて、独立性レベルのチュートリアルのデータベースを設定します。

チュートリアル: ダーティリードの知識 [809 ページ]
このチュートリアルでは、複数のトランザクションを同時に実行するときに発生する可能性のある矛盾である、ダーティリードについて説明します。

チュートリアル: 繰り返し不可能読み出しの知識 [814 ページ]
このチュートリアルでは、複数のトランザクションを同時に実行するときに発生する可能性のある矛盾である、繰り返し不可能読み出しを説明します。

チュートリアル: 幻ローの知識 [820 ページ]
このチュートリアルでは、複数のトランザクションを同時に実行するときに発生する可能性のある矛盾である、幻ローを再現してみます。

チュートリアル: 幻ロックの知識 [826 ページ]
幻ロックを使用して幻ローを防ぎます。

関連情報

独立性レベルと一貫性 [763 ページ]
スナップショットトランザクションの理解 [768 ページ]
典型的な矛盾のケース [772 ページ]
クエリ時のロック [791 ページ]

1.10.8.1 チュートリアル:独立性レベルのチュートリアルのシナリオ設定

Sales Manager と Accountant として動作する 2 つの Interactive SQL ウィンドウを開いて、独立性レベルのチュートリアルのデータベースを設定します。

コンテキスト

独立性レベルのすべてのチュートリアルは、Sales Manager と Accountant が同じ情報に同時にアクセスして変更するという、架空のシナリオを使用します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. Interactive SQL を起動します。 スタート > プログラム > *SQL Anywhere17* > 管理ツール > *Interactive SQL* をクリックします。
2. 接続ウィンドウで、Sales Manager として SQL Anywhere サンプルデータベースに接続します。
    a. パスワードフィールドにパスワード `sql` を入力します。
    b. アクションドロップダウンリストで、*ODBC* データソースを使用した接続をクリックします。
    c. *ODBC* データソース名をクリックし、その下のフィールドに *SQL Anywhere 17 Demo* と入力します。
    d. 詳細タブをクリックし、接続名フィールドに `Sales Manager` と入力します。
    e. 接続をクリックします。
3. Interactive SQL をもう 1 つ起動します。
4. 接続ウィンドウで、Accountant として SQL Anywhere サンプルデータベースに接続します。
    a. パスワードフィールドにパスワード `sql` を入力します。
    b. アクションドロップダウンリストで、*ODBC* データソースを使用した接続をクリックします。
    c. *ODBC* データソース名をクリックし、その下のフィールドに *SQL Anywhere 17 Demo* と入力します。
    d. 詳細オプションタブをクリックし、*ConnectionName* フィールドに `Accountant` と入力します。
    e. 接続をクリックします。

結果

Sales Manager と the Accountant としてのサンプルデータベースに接続されます。

次のステップ

独立性レベルのチュートリアルの 1 つを実行します。

関連情報

チュートリアル: ダーティリードの知識 [809 ページ]
チュートリアル: 繰り返し不可能読み出しの知識 [814 ページ]
チュートリアル: 幻ローの知識 [820 ページ]
チュートリアル: 幻ロックの知識 [826 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.8.2 チュートリアル: ダーティリードの知識

このチュートリアルでは、複数のトランザクションを同時に実行するときに発生する可能性のある矛盾である、ダーティリードについて説明します。

前提条件

SELECT ANY TABLE、UPDATE ANY TABLE、SET ANY SYSTEM OPTION のシステム権限が必要です。

"独立性レベルのチュートリアルのシナリオ設定" で説明されているように、このチュートリアルでは、Sales Manager および Accountant としてサンプルデータベースに接続していると仮定します。

コンテキスト

このシナリオでは、小規模商社の 2 人の従業員が会社のデータベースに同時にアクセスします。1 人は会社の Sales Manager で、もう 1 人は Accountant です。

Sales Manager は、会社で販売している T シャツの価格を 0.95 ドル上げようとしていますが、SQL 言語の構文に少し問題があります。それと同時に、Sales Manager が知らないうちに、Accountant が現在の在庫の小売り価格を計算し、次の管理ミーティングに必要なレポートに記載しようとしています。

> **注記**
>
> このチュートリアルを正常に機能させるには、Interactive SQL のデータベースロックの自動解放オプションをオフにする必要があります。このオプションの設定を確認するには、*[ツール]* > *[オプション]* をクリックし、左ウィンドウ枠の *[SQL Anywhere]* をクリックします。

このセクションの内容:

レッスン 1: ダーティリードの作成 [810 ページ]
Sales Manager が価格更新のプロセスを実行しているときに Accountant が計算を行う、ダーティリードを作成します。

レッスン 2: スナップショットアイソレーションを使用してダーティリードを避ける [812 ページ]
独立性レベルをスナップショットアイソレーションに設定します。

関連情報

独立性レベルと一貫性 [763 ページ]
スナップショットトランザクションの理解 [768 ページ]
典型的な矛盾のケース [772 ページ]
クエリ時のロック [791 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

チュートリアル:独立性レベルのチュートリアルのシナリオ設定 [807 ページ]

1.10.8.2.1 レッスン 1: ダーティリードの作成

Sales Manager が価格更新のプロセスを実行しているときに Accountant が計算を行う、ダーティリードを作成します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Accountant は、Sales Manager が入力後に訂正を加えている過程の間違った情報を使用して計算しました。

手順

1. Sales Manager として、次の文を実行してすべての T シャツの価格を 0.95 ドル上げます。

```
UPDATE GROUPO.Products
   SET UnitPrice = UnitPrice + 95
   WHERE Name = 'Tee Shirt';
SELECT ID, Name, UnitPrice
   FROM GROUPO.Products;
```

次の結果セットが返されます。

ID	Name	UnitPrice
300	Tee Shirt	104.00
301	Tee Shirt	109.00
302	Tee Shirt	109.00
400	Baseball Cap	9.00
...	...	...

ここで、Sales Manager 95 ではなく 0.95 を入力しなくてはならなかったことに気が付きます。しかし、間違いを訂正する前に、Accountant が別のオフィスからそのデータベースにアクセスしてきました。

2. Accountant は、在庫額が多いことを懸念しています。Accountant として次の文を実行し、全在庫品の小売り価格の合計を計算します。

```
SELECT SUM( Quantity * UnitPrice )
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
 AS Inventory
   FROM GROUPO.Products;
```

次の結果が返されます。

Inventory
21453.00

残念ながら、この計算は正確ではありません。Sales Manager が誤って T シャツの価格を 95 ドル上げてしまったため、合計に誤りがあります。このような誤りは、ダーティリードと呼ばれる典型的な矛盾の例です。Accountant であるあなたは、Sales Manager が入力したデータにアクセスしますが、このデータはまだコミットされていません。

3. Sales Manager として、最初の変更をロールバックし、正しい UPDATE 文を入力して間違いを訂正します。新しく入力した値が正しいかをチェックします。

```
ROLLBACK;
UPDATE GROUPO.Products
SET UnitPrice = UnitPrice + 0.95
WHERE NAME = 'Tee Shirt';
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```

次の結果セットが返されます。

ID	Name	UnitPrice
300	Tee Shirt	9.95
301	Tee Shirt	14.95
302	Tee Shirt	14.95
400	Baseball Cap	9.00
...	...	...

4. Accountant は、計算した値に誤りがあったことに気づきません。Accountant のウィンドウでもう一度 SELECT 文を実行すると、正しい値が表示されます。

```
SELECT SUM( Quantity * UnitPrice )
 AS Inventory
   FROM GROUPO.Products;
```

Inventory
6687.15

5. Sales Manager のウィンドウでのトランザクションを終了します。Sales Manager は COMMIT 文を入力して変更を確定できますが、ここでは、ROLLBACK 文を実行して、SQL Anywhere サンプルデータベースのローカルコピーが変更されないようにします。

```
ROLLBACK;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

Accountant は、データベースサーバが Sales Manager と Accountant の作業を同時に処理しているため、知らない間に間違った情報を受け取っています。

次のステップ

次のレッスンに進みます。

1.10.8.2.2 レッスン 2: スナップショットアイソレーションを使用してダーティリードを避ける

独立性レベルをスナップショットアイソレーションに設定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

スナップショットアイソレーションは、他のデータベースがクエリに対応してコミットされたデータだけに接続できるようにして、ダーティリードの発生を防ぎます。

Accountant は、スナップショットアイソレーションを使用して、コミットされたデータがクエリに影響を与えないようにすることができます。

手順

1. Sales Manager として次の文を実行し、データベースのスナップショットアイソレーションを有効にします。

   ```
   SET OPTION PUBLIC.allow_snapshot_isolation = 'ON';
   ```

2. Sales Manager として、すべての T シャツの価格を 0.95 ドル上げます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

a. 次の文を実行して価格を更新します。

```
UPDATE GROUPO.Products
SET UnitPrice = UnitPrice + 0.95
WHERE Name = 'Tee Shirt';
```

b. Sales Manager 用の新しい T シャツ価格を使用して、全在庫品の小売り価格の合計を計算します。

```
SELECT SUM( Quantity * UnitPrice )
 AS Inventory
   FROM GROUPO.Products;
```

次の結果が返されます。

Inventory
6687.15

3. Accountant として次の文を実行し、全在庫品の小売り価格の合計を計算します。このトランザクションは snapshot 独立性レベルを使用するため、データベースにコミットされたデータのみで結果が計算されます。

```
SET OPTION isolation_level = 'Snapshot';
SELECT SUM( Quantity * UnitPrice )
 AS Inventory
   FROM GROUPO.Products;
```

次の結果が返されます。

Inventory
6538.00

4. Sales Manager として次の文を実行し、データベースに対する変更をコミットします。

```
COMMIT;
```

5. Accountant として次の文を実行し、現在の在庫の更新後の小売り価格を表示します。

```
COMMIT;
SELECT SUM( Quantity * UnitPrice )
 AS Inventory
   FROM GROUPO.Products;
```

次の結果が返されます。

Inventory
6687.15

Accountant のトランザクションで使用されるスナップショットは最初の読み込み操作で開始するため、COMMIT を実行してトランザクションを終了し、スナップショットトランザクションの開始後に加えられたデータの変更を Accountant が確認できるようにする必要があります。

6. Sales Manager として次の文を実行し、T シャツの価格の変更を取り消し、SQL Anywhere サンプルデータベースを元の状態に復元します。

```
UPDATE GROUPO.Products
SET UnitPrice = UnitPrice - 0.95
WHERE Name = 'Tee Shirt';
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
COMMIT;
```

結果

Accountant は、スナップショットアイソレーションを有効にして、ダーティリードを防ぐことができました。

次のステップ

(省略可) サンプルデータベース (*demo.db*) を元の状態にリストアします。

1.10.8.3 チュートリアル: 繰り返し不可能読み出しの知識

このチュートリアルでは、複数のトランザクションを同時に実行するときに発生する可能性のある矛盾である、繰り返し不可能読み出しを説明します。

前提条件

SELECT ANY TABLE、UPDATE ANY TABLE、SET ANY SYSTEM OPTION のシステム権限が必要です。

このチュートリアルでは、Sales Manager および Accountant としてサンプルデータベースに接続していると仮定します。

コンテキスト

このシナリオでは、小規模商社の 2 人の従業員が会社のデータベースに同時にアクセスします。1 人は会社の Sales Manager で、もう 1 人は Accountant です。

Sales Manager はプラスチックバイザーに新しい価格を設定すると仮定します。Accountant は最近注文があったいくつかの品目の価格を調べるとします。

この例では、独立性レベル 0 ではなく 1 を使って、両方の接続を開始します。独立性レベル 0 は SQL Anywhere サンプルデータベースのデフォルト値です。独立性レベルを 1 に設定することで、ダーティリードの可能性を減らします。

> **i 注記**
>
> このチュートリアルを正常に機能させるには、Interactive SQL のデータベースロックの自動解放オプションをオフにする必要があります。このオプションの設定を確認するには、 *[ツール]* > *[オプション]* をクリックし、左ウィンドウ枠の *[SQL Anywhere]* をクリックします。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

レッスン 1: 繰り返し不可能読み出しの作成 [815 ページ]
Sales Manager によって変更され、同じトランザクション中に 2 つの異なる結果を取得するローを Accountant が読み出す、繰り返し不可能読み出しを作成します。

レッスン 2: スナップショットアイソレーションを使用して繰り返し不可能読み出しを避ける [818 ページ]
スナップショットアイソレーションを使用すると、ブロックを避けることができます。

関連情報

独立性レベルと一貫性 [763 ページ]
スナップショットトランザクションの理解 [768 ページ]
典型的な矛盾のケース [772 ページ]
クエリ時のロック [791 ページ]
チュートリアル:独立性レベルのチュートリアルのシナリオ設定 [807 ページ]

1.10.8.3.1 レッスン 1: 繰り返し不可能読み出しの作成

Sales Manager によって変更され、同じトランザクション中に 2 つの異なる結果を取得するローを Accountant が読み出す、繰り返し不可能読み出しを作成します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次の文を実行して、Accountant の接続の独立性レベルを 1 に設定します。

```
SET TEMPORARY OPTION isolation_level = 1;
```

2. 次の文を実行して、Sales Manager のウィンドウに独立性レベル 1 を設定します。

```
SET TEMPORARY OPTION isolation_level = 1;
```

3. Accountant として、次の文を実行し、サンバイザーの価格のリストを表示します。

```
SELECT ID, Name, UnitPrice
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
FROM GROUPO.Products;
```

ID	Name	UnitPrice
300	Tee Shirt	9.00
301	Tee Shirt	14.00
302	Tee Shirt	14.00
400	Baseball Cap	9.00
401	Baseball Cap	10.00
500	Visor	7.00
501	Visor	7.00
...	...	...

4. Sales Manager として、次の文を実行し、プラスチック製サンバイザーの新しい販売価格を導入します。

```
SELECT ID, Name, UnitPrice FROM GROUPO.Products
WHERE Name = 'Visor';
UPDATE GROUPO.Products
SET UnitPrice = 5.95 WHERE ID = 501;
COMMIT;
SELECT ID, Name, UnitPrice FROM GROUPO.Products
WHERE Name = 'Visor';
```

ID	Name	UnitPrice
500	Visor	7.00
501	Visor	5.95

5. Sales Manager ウィンドウでのバイザーの価格と Accountant ウィンドウでの価格を比較してみてください。Accountant として SELECT 文をもう一度実行すると、Sales Manager の新価格が表示されます。

```
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```

ID	Name	UnitPrice
300	Tee Shirt	9.00
301	Tee Shirt	14.00
302	Tee Shirt	14.00
400	Baseball Cap	9.00
401	Baseball Cap	10.00
500	Visor	7.00
501	Visor	5.95
...	...	...

この矛盾を「繰り返し不可能読み出し」と呼んでいます。Accountant が 2 回目も同じトランザクションで同じ SELECT 文を実行した後で、同じ結果が得られないためです。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

もちろん、Accountant が SELECT を再度使用する前に、COMMIT や ROLLBACK 文などを発行してトランザクションを終了した場合には、状況は違ってきます。データベースは複数のユーザが同時に使用でき、Accountant のトランザクションの前後に他の人が値を変更できます。他の人が行った変更によって矛盾が生じるのは、Accountant のトランザクションの途中で変更が行われるためです。そうしたイベントにより、スケジュールは直列化不可能となります。

6. Accountant はこれに気づき、以降は価格参照中にほかからの変更を防ぐことにします。繰り返し不可能読み出しは、独立性レベル 2 で削除されます。Accountant として、次の文を実行します。

```
SET TEMPORARY OPTION isolation_level = 2;
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```

7. Sales Manager は、明日受注するはずの大口注文に値下げを適用しなくて済むように、プラスチックバイザーの安売りを来週に延期することを決定します。Sales Manager として、次の文を実行します。文の実行が始まりますが、ウィンドウは実行中のまま終了しません。

```
UPDATE GROUPO.Products
SET UnitPrice = 7.00
WHERE ID = 501;
```

データベースサーバは独立性レベル 2 では繰り返し可能読み出しを保証する必要があります。Accountant は独立性レベル 2 を使用するため、データベースサーバは Accountant が読み込む Products テーブルの各ローに読み込みロックをかけます。Sales Manager が価格を元に戻そうとすると、そのトランザクションは、Products テーブルのプラスチックバイザーのローに書き込みロックをかける必要があります。書き込みロックは排他ロックであるため、Accountant のトランザクションが読み込みロックを解放するまで待機しなければなりません。
8. Accountant が価格の閲覧を終了しました。データベースを誤って変更するのを防ぐために、ROLLBACK 文でトランザクションを完了します。

```
ROLLBACK;
```

データベースサーバがこの文を実行すると、Sales Manager のトランザクションが完了します。

ID	Name	UnitPrice
500	Visor	7.00
501	Visor	7.00

9. Sales Manager もトランザクションを終了できるようになりました。Sales Manager は、変更をコミットして、元の価格をリストアします。

```
COMMIT;
```

結果

Accountant は、同じトランザクション中に異なる結果を受け取ります。したがって、スナップショット独立性レベル 2 を有効にして、繰り返し不可能読み出しを防ぎます。Accountant がデータベースを変更すると、Sales Manager はデータベースを変更できなくなります。

Accountant の独立性レベルを 1 から 2 に更新したときに、データベースサーバは、前に誰もロックをかけたことのない場所で読み込みロックを使用しました。それ以降、選択内容に適合するローごとに読み込みロックをかけました。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

前述のチュートリアルでは、UPDATE 文の実行中に Sales Manager のウィンドウがフリーズしました。データベースサーバは UPDATE 文の実行を開始し、Sales Manager が変更を必要としているローに Accountant のトランザクションが読み込みロックをかけていることを発見しました。この時点で、データベースサーバは UPDATE の実行を一時停止します。Accountant が ROLLBACK でトランザクションを終了すると、データベースサーバは自動的にロックを解放します。妨げがなくなると、データベースサーバが Sales Manager の UPDATE を最後まで処理します。

次のステップ

次のレッスンに進みます。

関連情報

ブロックオプション [779 ページ]
レッスン 2: スナップショットアイソレーションを使用して繰り返し不可能読み出しを避ける [818 ページ]

1.10.8.3.2 レッスン 2: スナップショットアイソレーションを使用して繰り返し不可能読み出しを避ける

スナップショットアイソレーションを使用すると、ブロックを避けることができます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

スナップショットアイソレーションを使用するトランザクションはコミットされたデータだけを認識するため、Accountant のトランザクションは、Sales Manager のトランザクションをブロックしません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. Accountant として次の文を実行して、データベースのスナップショットアイソレーションを有効にし、スナップショットアイソレーションレベルを使用することを指定します。

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
SET TEMPORARY OPTION isolation_level = 'snapshot';
```

2. Accountant として、次の文を実行し、サンバイザーの価格のリストを表示します。

```
SELECT ID, Name, UnitPrice
FROM GROUPO.Products
ORDER BY ID;
```

ID	Name
300	Tee Shirt
301	Tee Shirt
302	Tee Shirt
400	Baseball Cap
401	Baseball Cap
500	Visor
501	Visor
...	...

3. Sales Manager として、次の文を実行し、プラスチック製サンバイザーの新しい販売価格を導入します。

```
UPDATE GROUPO.Products
SET UnitPrice = 5.95 WHERE ID = 501;
COMMIT;
SELECT ID, Name, UnitPrice FROM GROUPO.Products
WHERE Name = 'Visor';
```

4. Accountant はクエリをもう一度実行しますが、最初の読み込み時にコミットされたデータがトランザクションで使用されるため、価格の変更を認識しません。

```
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```

5. Sales Manager として、プラスチックバイザーを元の価格に戻します。

```
UPDATE GROUPO.Products
SET UnitPrice = 7.00
WHERE ID = 501;
COMMIT;
```

データベースサーバは、Accountant が読み込み中の Products テーブルのローに読み込みロックをかけません。これは Sales Manager が Products テーブルに変更を加える前に作成された、コミットされたデータのスナップショットを Accountant が閲覧しているためです。

6. Accountant が価格の閲覧を終了しました。データベースを誤って変更するのを防ぐために、ROLLBACK 文でトランザクションを完了します。

```
ROLLBACK;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

結果

スナップショットアイソレーションを使用して、繰り返し不可能読み出しを防ぐことができました。

1.10.8.4 チュートリアル: 幻ローの知識

このチュートリアルでは、複数のトランザクションを同時に実行するときに発生する可能性のある矛盾である、幻ローを再現してみます。

前提条件

SELECT ANY TABLE、INSERT ANY TABLE、DELETE ANY TABLE、SET ANY SYSTEM OPTION のシステム権限が必要です。

このチュートリアルでは、Sales Manager および Accountant としてサンプルデータベースに接続していると仮定します。

コンテキスト

このシナリオでは、小規模商社の 2 人の従業員が会社のデータベースに同時にアクセスします。1 人は会社の Sales Manager で、もう 1 人は Accountant です。

Sales Manager は、新しく Foreign Sales と Major Account Sales の部署を作りたいと考えています。Accountant は、会社に存在するすべての部署を確認したいと考えています。

この例では、独立性レベル 0 ではなく 2 を使って、両方の接続を開始します。独立性レベル 0 は SQL Anywhere サンプルデータベースのデフォルト値です。独立性レベルを 2 に設定することで、ダーティリードと繰り返し不可能読み出しの可能性を減らします。

> **注記**
>
> このチュートリアルを正常に機能させるには、Interactive SQL のデータベースロックの自動解放オプションをオフにする必要があります。このオプションの設定を確認するには、 *[ツール]* › *[オプション]* をクリックし、左ウィンドウ枠の *[SQL Anywhere]* をクリックします。

このセクションの内容:

[レッスン 1: 幻ローの作成 [821 ページ]](#)
: Sales Manager がローを挿入し、Accountant が隣接するローを読み出すことで、幻ローを作成します。これにより、新しいローが幻として表示されます。

[レッスン 2: スナップショットアイソレーションを使用して幻ローを防ぐ [824 ページ]](#)
: スナップショットアイソレーションレベルを使用すると、独立性レベル 3 と同じレベルで一貫性を維持することができ、ブロックが発生しません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

独立性レベルと一貫性 [763 ページ]
スナップショットトランザクションの理解 [768 ページ]
典型的な矛盾のケース [772 ページ]
クエリ時のロック [791 ページ]
チュートリアル:独立性レベルのチュートリアルのシナリオ設定 [807 ページ]

1.10.8.4.1 レッスン 1: 幻ローの作成

Sales Manager がローを挿入し、Accountant が隣接するローを読み出すことで、幻ローを作成します。これにより、新しいローが幻として表示されます。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次の文を実行して、Sales Manager と Accountant のウィンドウにそれぞれ独立性レベル 2 を設定します。

```
SET TEMPORARY OPTION isolation_level = 2;
```

2. Accountant として次の文を実行し、すべての部署のリストを表示します。

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing
500	Shipping

3. Sales Manager は、主に外国市場を担当する新しい部署の設置を決めました。EmployeeID 129 の Philip Chin を新しい部署の責任者にします。Sales Manager として、次の文を実行し、新しい部署用のエントリを作成します。この新しいエントリは、Sales Manager のウィンドウのテーブル下部に新しいローとして表示されます。

```
INSERT INTO GROUPO.Departments
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

```
    ( DepartmentID, DepartmentName, DepartmentHeadID )
    VALUES( 600, 'Foreign Sales', 129 );
    COMMIT;
```

4. Sales Manager として次の文を実行し、すべての部署のリストを表示します。

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing
500	Shipping
600	Foreign Sales

5. しかし Accountant は、新しい部署が作成されたことに気づきません。独立性レベル 2 では、データベースサーバはローを変更しないようにロックをかけますが、他のトランザクションが新しいローを挿入するのを防止するロックはかけていません。

   Accountant は SELECT 文を再実行した場合にだけ、新しいローを発見できます。Accountant として、SELECT 文を実行して、テーブルに追加された新しいローをもう一度表示します。

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing
500	Shipping
600	Foreign Sales

   新しく追加されたローは、幻ローと呼ばれます。これは、Accountant の観点から見ると、このローがどこからともなく出現した幻のように映るためです。Accountant は独立性レベル 2 で接続されます。このレベルでは、サーバは使用中のローにだけロックをかけます。他のローにはロックがかけられないため、Sales Manager が新しいローを挿入するのを妨げるものはありません。

6. Accountant は今後そうしたことが起こらないように、現在のトランザクションの独立性レベルを 3 に上げることにします。Accountant として、次の文を実行します。

```
SET TEMPORARY OPTION isolation_level = 3;
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

7. Sales Manager は、大企業に対して営業活動を行う新しい部署を追加したいと考えています。Sales Manager として、次の文を実行します。

```
INSERT INTO GROUPO.Departments
 ( DepartmentID, DepartmentName, DepartmentHeadID )
   VALUES( 700, 'Major Account Sales', 902 );
```

Accountant のロックが文をブロックするため、Sales Manager のウィンドウは実行中に一時停止します。ツールバーで、*[停止]* をクリックして、このエントリを中止します。

Accountant が独立性レベルを 3 に上げ、Departments テーブルのすべてのローを再度選択した場合、データベースサーバはテーブルの各ローに対挿入ロックを設定し、新しいローの挿入を防止するためにテーブルの最後にもう 1 つ幻ロックを設定します。Sales Manager がテーブルの最後に新しいローを挿入しようとすると、その文はこの最後のロックによってブロックされます。

Sales Manager は独立性レベル 2 で接続されているにもかかわらず、Sales Manager の文はブロックされました。データベースサーバは独立性レベルと各トランザクション文の要求に応じ、読み込みロックと同様に対挿入ロックを設定します。一度対挿入ロックが設定されると、このロックは同時に実行される他のすべてのトランザクションに適用されます。

8. SQL Anywhere サンプルデータベースが変更されないようにするため、Major Account Sales 部署のローを挿入する未完了トランザクションをロールバックし、2 つ目のトランザクションを使用して Foreign Sales 部署を削除してください。
   a. Accountant として、次の文を実行して独立性レベルを引き下げ、ローのロックを解除することで、Sales Manager がデータベースに対する変更を取り消すことができるようにします。

   ```
   SET TEMPORARY OPTION isolation_level=1;
   ROLLBACK;
   ```

   b. Sales Manager として、次の文を実行して現在のトランザクションをロールバックし、以前挿入されたローを削除し、この操作をコミットします。

   ```
   ROLLBACK;
   DELETE FROM GROUPO.Departments
   WHERE DepartmentID = 600;
   COMMIT;
   ```

結果

Accountant は、SELECT 文が実行されるたびに異なる結果を受け取ります。したがって、スナップショット独立性レベル 3 を有効にして、幻ローを防ぎます。Accountant がデータベースを変更すると、Sales Manager はデータベースを変更できなくなります。

次のステップ

次のレッスンに進みます。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

ロックの仕組み [782 ページ]

1.10.8.4.2 レッスン 2: スナップショットアイソレーションを使用して幻ローを防ぐ

スナップショットアイソレーションレベルを使用すると、独立性レベル 3 と同じレベルで一貫性を維持することができ、ブロックが発生しません。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Sales Manager の文はブロックされず、Accountant は幻ローを認識しません。

手順

1. Accountant として、次の文を実行してスナップショットアイソレーションを有効にします。

```
SET OPTION PUBLIC. allow_snapshot_isolation = 'On';
SET TEMPORARY OPTION isolation_level = 'snapshot';
```

2. 次の文を実行して、すべての部署をリストします。

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

DepartmentID	DepartmentName
500	Shipping

3. Sales Manager は、主に外国市場を担当する新しい部署の設置を決めました。EmployeeID 129 の Philip Chin を新しい部署の責任者にします。Sales Manager として、次の文を実行し、新しい部署用のエントリを作成します。この新しいエントリは、Sales Manager のウィンドウのテーブル下部に新しいローとして表示されます。

```
INSERT INTO GROUPO.Departments
   ( DepartmentID, DepartmentName, DepartmentHeadID )
   VALUES( 600, 'Foreign Sales', 129 );
COMMIT;
```

4. Sales Manager として次の文を実行し、すべての部署のリストを表示します。

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

DepartmentID	Department Name
100	R & D
200	Sales
300	Finance
400	Marketing
500	Shipping
600	Foreign Sales

5. Accountant はクエリをもう一度実行できます。また、トランザクションがコミットされていないため、新しいローを認識しません。

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing
500	Shipping

6. Sales Manager は、大企業に対して営業活動を行う新しい部署を追加したいと考えています。Sales Manager として、次の文を実行します。

```
INSERT INTO GROUPO.Departments
 ( DepartmentID, DepartmentName, DepartmentHeadID )
   VALUES( 700, 'Major Account Sales', 902 );
```

Accountant がスナップショットアイソレーションを使用しているため、Sales Manager による変更はブロックされません。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

7. Sales Manager がデータベースにコミットした変更を認識するために、Accountant はスナップショットアイソレーションを終了する必要があります。

```
COMMIT;
   SELECT * FROM GROUPO.Departments
   ORDER BY DepartmentID;
```

Accountant は Foreign Sales 部署を認識するようになりました。ただし、Major Account Sales 部署は認識しません。

DepartmentID	Department Name
100	R & D
200	Sales
300	Finance
400	Marketing
500	Shipping
600	Foreign Sales

8. SQL Anywhere サンプルデータベースが変更されないようにするため、Major Account Sales 部署のローを挿入する未完了トランザクションをロールバックし、2 つ目のトランザクションを使用して Foreign Sales 部署を削除してください。
   a. Sales Manager として、次の文を実行して現在のトランザクションをロールバックし、以前挿入されたローを削除し、この操作をコミットします。

```
ROLLBACK;
DELETE FROM GROUPO.Departments
WHERE DepartmentID = 600;
COMMIT;
```

結果

スナップショットアイソレーションを使用して、幻ローを防ぐことができました。

1.10.8.5 チュートリアル: 幻ロックの知識

幻ロックを使用して幻ローを防ぎます。

前提条件

SELECT ANY TABLE、INSERT ANY TABLE、DELETE ANY SYSTEM のシステム権限が必要です。

このチュートリアルでは、Sales Manager および Accountant としてサンプルデータベースに接続していると仮定します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

> **注記**
>
> このチュートリアルを正常に機能させるには、Interactive SQL のデータベースロックの自動解放オプションをオフにする必要があります。このオプションの設定を確認するには、 *[ツール]* *[オプション]* をクリックし、左ウィンドウ枠の *[SQL Anywhere]* をクリックします。

コンテキスト

このチュートリアルでは、幻ロックについて説明します。幻ロックは幻ローを防ぐためにインデックススキャン位置に設定される共有ロックです。独立性レベル 3 のトランザクションが指定された基準を満たすローを選択すると、データベースサーバは対挿入ロックを設定し、他のトランザクションが基準を満たすローを挿入することを禁止します。設置するロック数は、検索基準とデータベースの設計によって異なります。

Accountant と Sales Manager はどちらも SalesOrder テーブルと SalesOrderItems テーブルに関わるタスクがあります。Accountant は、販売担当者に支払ったコミッションの小切手額を確認する必要があります。Sales Manager は、失われた注文がいくつかあることに気づき、それを追加したいと思っています。

手順

1. 次の文を実行して、Sales Manager と Accountant のウィンドウにそれぞれ独立性レベルを 2 に設定します。

```
SET TEMPORARY OPTION isolation_level = 2;
```

2. 毎月、販売担当者には、その月の各人の売り上げ高に対し、一定の割合のコミッションが支払われます。Accountant は、2001 年 4 月分のコミッションの小切手を準備しています。彼の最初のタスクは、その月の各担当者の売り上げ合計額を計算することです。価格、注文情報、従業員データがそれぞれ別のテーブルに保存されます。外部キー関係を使用してこれらのテーブルをジョインすることにより、必要な情報を結合します。

   Accountant として、次の文を実行します。

```
SELECT EmployeeID, GivenName, Surname,
   SUM( SalesOrderItems.Quantity * UnitPrice )
      AS "April sales"
FROM GROUPO.Employees
   KEY JOIN GROUPO.SalesOrders
   KEY JOIN GROUPO.SalesOrderItems
   KEY JOIN GROUPO.Products
WHERE '2001-04-01' <= OrderDate
   AND OrderDate < '2001-05-01'
GROUP BY  EmployeeID, GivenName, Surname
ORDER BY EmployeeID;
```

EmployeeID	GivenName	Surname	April sales
129	Philip	Chin	2160.00
195	Marc	Dill	2568.00
299	Rollin	Overbey	5760.00

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

EmployeeID	GivenName	Surname	April sales
467	James	Klobucher	3228.00
...	...	...	...

3. Sales Manager は、Philip Chin の多額の売り上げがデータベースに入力されていないことに気づきました。Philip はコミッションの迅速な支払いを希望しているため、Sales Manager は 4 月 25 日の彼の漏れていた注文を入力します。

   Sales Manager として、次の文を実行します。1 つの注文に多くの品目を含めることができるため、受注と品目は別のテーブルに入力されます。品目を追加する前に、売り上げ注文にエントリを作成します。参照整合性を維持するために、データベースサーバは注文がすでに存在する場合にかぎり、トランザクションが品目を注文に追加することを許可します。

```
INSERT into GROUPO.SalesOrders
VALUES ( 2653, 174, '2001-04-22', 'r1',
      'Central', 129 );
INSERT into GROUPO.SalesOrderItems
VALUES ( 2653, 1, 601, 100, '2001-04-25' );
COMMIT;
```

4. Accountant は、Sales Manager が新しい注文を追加したことを知りません。新しい注文がもっと前に入力された場合は、Philip Chin の 4 月の売り上げ計算に含まれます。

   Accountant のウィンドウで、4 月の売り上げ合計を再計算します。同じ文を使用しますが、Philip Chin の 4 月の売り上げ額が 4560.00 ドルに変更されていることに注意してください。

EmployeeID	GivenName
129	Philip
195	Marc
299	Rollin
467	James
...	...

   ここで、Accountant は 4 月の注文すべてにマークを付けて、コミッションが支払い済であることを示します。Sales Manager が入力したばかりの注文が 2 度目の検索で見つかりましたが、たとえ Philip の 4 月の売り上げ合計に含まれていなかったとしても、支払い済みのマークが付いています。

5. 独立性レベル 3 では、データベースサーバは対挿入ロックを設定し、検索または選択条件に合うローを他のトランザクションが追加できないようにします。

   Sales Manager として次の文を実行し、新しい注文を削除します。

```
DELETE
FROM GROUPO.SalesOrderItems
WHERE ID = 2653;
DELETE
FROM GROUPO.SalesOrders
WHERE ID = 2653;
COMMIT;
```

6. Accountant として、次の文を実行します。

```
ROLLBACK;
SET TEMPORARY OPTION isolation_level = 3;
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

7. Accountant として、次のクエリを実行します。

```
SELECT EmployeeID, GivenName, Surname,
   SUM( SalesOrderItems.Quantity * UnitPrice )
      AS "April sales"
FROM GROUPO.Employees
   KEY JOIN GROUPO.SalesOrders
   KEY JOIN GROUPO.SalesOrderItems
   KEY JOIN GROUPO.Products
WHERE '2001-04-01' <= OrderDate
   AND OrderDate < '2001-05-01'
GROUP BY  EmployeeID, GivenName, Surname;
```

独立性レベルを 3 に設定したため、データベースサーバは自動的に対挿入ロックを設定し、Accountant がトランザクションを終了するまで、Sales Manager が 4 月の注文品目を挿入できないようにします。

8. Sales Manager として次の文を実行して、Philip Chin の漏れていた注文の入力を試みます。

```
INSERT INTO GROUPO.SalesOrders
VALUES ( 2653, 174, '2001-04-22',
         'r1','Central', 129 );
```

Sales Manager のウィンドウは応答を停止し、操作は完了しません。ツールバーの *SQL* 文の中断をクリックしてこのエントリを一時停止します。

9. Sales Manager は 4 月の注文を入力できませんが、5 月分には入力できると考えます。

文の日付を 5 月 5 日に変更し、再実行します。

```
INSERT INTO GROUPO.SalesOrders
VALUES ( 2653, 174, '2001-05-05', 'r1',
      'Central', 129 );
```

Sales Manager のウィンドウは再度応答を停止します。ツールバーの *SQL* 文の中断をクリックしてこのエントリを一時停止します。データベースサーバでは挿入を防止するために必要となるロック以外は設定しませんが、これらのロックによって、多数のトランザクションの処理が妨げられる可能性があります。

データベースサーバはテーブルインデックスにロックをかけます。たとえば、インデックスに幻ロックを設定し、インデックスの直前に新しいローを追加できないようにします。ただし、適切なインデックスが存在しない場合、テーブルのすべてのローにロックをかける必要があります。ある状況では、対挿入ロックによって特定テーブルへの挿入のみをブロックできます。

10. サンプルデータベースの変更を防ぐために、SalesOrders テーブルに対する変更をロールバックする必要があります。Sales Manager ウィンドウと Accountant ウィンドウの両方で、次の文を実行します。

```
ROLLBACK;
```

11. Interactive SQL の両方のインスタンスを終了します。

結果

幻ロックの動作方法を理解するチュートリアルを完了しました。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

独立性レベルと一貫性 [763 ページ]
スナップショットトランザクションの理解 [768 ページ]
典型的な矛盾のケース [772 ページ]
クエリ時のロック [791 ページ]

1.10.9 ユニークな値を生成するためのシーケンスの使用

シーケンスを使用すると、複数のテーブル間でユニークな値、つまり一連の自然数と異なる値を生成できます。

シーケンスは、CREATE SEQUENCE 文を使用して作成します。シーケンス値は BIGINT 値として返されます。

各接続で、最後に使用された「次の値」が現在の値として保存されます。

シーケンスを作成すると、データベースサーバがメモリに保持しているシーケンス値の数が定義に含められます。この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ます。データベースサーバで障害が発生した場合、キャッシュに保持されていたシーケンス値がスキップされることがあります。

シーケンス値の取得

シーケンスの現在の値を取得するには、次の文を使用します。

```
SELECT sequence-name.CURRVAL;
```

シーケンスの次の値を取得するには、次の文を使用します。

```
SELECT sequence-name.NEXTVAL;
```

シーケンス値と **AUTOINCREMENT** 値からの選択

AUTOINCREMENT の動作	シーケンスの動作
テーブルの単一のカラムに定義される	データベースオブジェクトとして格納され、式が許される場所であればどこでも使用できる
カラムのデータ型は整数データ型または真数値データ型	値は式が使用できる場所であればどこからでも参照でき、カラムのデフォルト値に適合しなくてもよい
値は 1 つのテーブルの単一のカラムでのみ使用できる	値は複数のテーブルで使用できる
値は一連の自然数 (1、2、3、...) の一部である	一連の自然数ではない値を生成できる
値は増分される	値は増分または減分できる

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

AUTOINCREMENT の動作	シーケンスの動作
カラムの以前の最大値より 1 大きいユニークな値がデフォルトで生成される sa_reset_identity システムプロシージャを使用して、次に挿入するローの AUTOINCREMENT 値を変更できる	増分の単位を指定できる
生成される次の値がカラムに格納できる最大値を超えている場合、NULL が返される	最大値または最小値に達した後に値を生成できるように選択するか、または NO CYCLE を指定することでエラーを返す

シーケンスの例

顧客ホットラインのインシデント番号を生成するために使用されるシーケンスの場合を考えてみます。顧客からの苦情電話には 2 種類のタイプ (請求が間違っている、または荷物が行方不明) があると想定します。

```
CREATE SEQUENCE incidentSequence
   MINVALUE 1000
   MAXVALUE 100000;
```

```
CREATE TABLE reportedBillingMistake(
   incidentID INT PRIMARY KEY DEFAULT (incidentSequence.nextval),
   billNumber INT,
   valueOnBill NUMERIC(10,2),
   expectedValue NUMERIC(10,2),
   comments LONG VARCHAR );
```

```
CREATE TABLE reportedMissingShipment(
   incidentID INT PRIMARY KEY DEFAULT(incidentSequence.nextval),
   orderNumber INT,
   comments LONG VARCHAR );
```

incidentSequence.nextval を incidentID カラムに使用することによって、incidentID が 2 つのテーブルでユニークであることが保障されます。顧客から再度問い合わせの電話があり、インシデント値が提供されたときに、請求の問題か荷物の出荷の問題かわからなくなることはありません。

請求の間違いを挿入する場合、次の文は同義です。

```
INSERT INTO reportedBillingMistake VALUES( DEFAULT, 12345, 100.00, 75.00, 'Bad
bill' );
INSERT INTO reportedBillingMistake
   SELECT incidentSequence.nextval, 12345, 100.00, 75.00, 'Bad bill';
```

直前に挿入された incidentID を見つけるには、挿入を実行した (上の 2 つの文のいずれかを使用) 接続で次の文を実行できます。

```
SELECT incidentSequence.currval;
```

このセクションの内容:

シーケンスの作成 [832 ページ]
SQL Central を使用してシーケンスを作成します。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

シーケンスの変更 [833 ページ]
SQL Central を使用してシーケンスを変更します。

シーケンスの削除 [833 ページ]
SQL Central を使用してシーケンスを削除します。

関連情報

オートインクリメントデフォルト [735 ページ]
GLOBAL AUTOINCREMENT デフォルト [735 ページ]

1.10.9.1 シーケンスの作成

SQL Central を使用してシーケンスを作成します。

前提条件

CREATE ANY SEQUENCE または CREATE ANY OBJECT のシステム権限が必要です。

手順

1. SQL Central で、*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. 左ウィンドウ枠でシーケンスジェネレータを右クリックし、新規 > シーケンスジェネレータをクリックします。
3. シーケンスジェネレータの作成ウィザードの指示に従います。

結果

シーケンスを作成しています。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.10.9.2 シーケンスの変更

SQL Central を使用してシーケンスを変更します。

前提条件

その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

手順

1. SQL Central で、*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. シーケンスジェネレータを右クリックして、プロパティをクリックします。

   一般タブで、シーケンスの設定を変更できます。すぐに再起動をクリックすると、ALTER SEQUENCE...RESTART WITH `n` 文が実行されます。ここで、`n` は開始値フィールドの値に対応しています。

結果

変更は直ちに有効になります。

1.10.9.3 シーケンスの削除

SQL Central を使用してシーケンスを削除します。

前提条件

その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。

- DROP ANY SEQUENCE システム権限
- DROP ANY OBJECT システム権限

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. SQL Central で、*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. シーケンスジェネレータを右クリックして、削除をクリックします。

結果

シーケンスはデータベースから削除されます。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。

1.11 SQL Anywhere のデバッガ

SQL Anywhere のデバッガは、作成した SQL のストアドプロシージャ、トリガ、イベントハンドラ、ユーザ定義関数をデバッグするために使用できます。

デバッガは以下の作業に使用することもできます。

イベントハンドラのデバッグ

イベントハンドラは SQL ストアドプロシージャの拡張機能です。次のストアドプロシージャのデバッグに関する説明は、イベントハンドラのデバッグにも同様に当てはまります。

ストアドプロシージャとクラスのブラウズ

SQL プロシージャのソースコードをブラウズできます。

実行のトレース

ストアドプロシージャのコードを 1 行ずつ実行できます。呼び出された関数のスタックを前後に検索することもできます。

ブレークポイントの設定

ブレークポイントまでコードを実行して停止します。

ブレーク条件の設定

ブレークポイントには複数行のコードが含まれますが、コードをブレークする場合、条件も指定できます。たとえば、ある行を 10 回実行された時点で停止させたり、変数が特定の値を持つ場合にだけ停止させたりできます。

ローカル変数の検査と修正

実行がブレークポイントで停止したときに、ローカル変数の値を検査して変更できます。

式を検査してブレークする

実行がブレークポイントで停止したときに、さまざまな式の値を検査できます。

ロー変数の検査と修正

ロー変数はローレベルトリガの OLD と NEW の値です。これらの値を検査して修正できます。

クエリの実行

SQL プロシージャのブレークポイントで実行が停止したときに、クエリを実行できます。これによって、テンポラリテーブルに保持される中間結果を参照したり、ベーステーブルの値をチェックしたり、クエリ実行プランを参照したりできます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

このセクションの内容:

デバッガの稼働条件 [835 ページ]
デバッガを使用するには、いくつかの条件を満たす必要があります。たとえば、一度に 1 ユーザのみがデバッガを使用することができます。
チュートリアル: デバッガの使用開始 [835 ページ]
デバッガを使用してストアドプロシージャのエラーを識別する方法を学習します。
ブレークポイント [841 ページ]
ブレークポイントでは、デバッガがソースコードの実行を中断するタイミングを制御します。
デバッガによる変数の変更 [844 ページ]
デバッガでは、コードをステップしながら変数の動作を表示して編集できます。
接続とブレークポイント [847 ページ]
SQL Central の接続タブには、データベースへの接続が表示されます。

1.11.1 デバッガの稼働条件

デバッガを使用するには、いくつかの条件を満たす必要があります。たとえば、一度に 1 ユーザのみがデバッガを使用することができます。

HTTP/SOAP 接続を介してデバッガを使用する場合は、サーバのポートのタイムアウトオプションを変更してください。たとえば、`-xs http{TO=600;KTO=0;PORT=8081}` では、ポート 8081 のタイムアウトが 10 分に設定され、キープアライブのタイムアウトがオフになります。Timeout (TO) は受信パケットと次の受信パケットの間に経過する時間であることに注意してください。キープアライブタイムアウト (KTO) は、接続が実行を許可される時間の合計時間です。KTO を 0 に設定することは、タイムアウトしないように設定することと同じです。

SQL Anywhere の HTTP/SOAP クライアントプロシージャを使用して、デバッグしている SQL Anywhere HTTP/SOAP サービスを呼び出す場合、クライアントの remote_idle_timeout データベースオプションを 150 などの大きい値 (デフォルトは 15 秒) に設定して、デバッグセッション中のタイムアウトを回避します。

1.11.2 チュートリアル: デバッガの使用開始

デバッガを使用してストアドプロシージャのエラーを識別する方法を学習します。

前提条件

SA_DEBUG システムロールが必要です。

さらに、EXECUTE ANY PROCEDURE システム権限か、システムプロシージャ debugger_tutorial に対する EXECUTE 権限が必要です。また、ALTER ANY PROCEDURE システム権限または ALTER ANY OBJECT システム権限も必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

SQL Anywhere のサンプルデータベース、*demo.db* には、debugger_tutorial という名前の意図的にエラーが含まれたストアドプロシージャがあります。debugger_tutorial システムプロシージャは、発注額の最も多い会社名とその発注額を含む結果セットを返します。プロシージャは、会社と発注をリストするクエリの結果セットをループしてこれらの値を計算します。(この結果は、プロシージャにこの論理を追加しなくとも、SELECT FIRST クエリを使用して得ることができます。このプロシージャは説明を目的としたものです)。ただし、debugger_tutorial システムプロシージャにはバグが含まれているためエラーになり、正しい結果セットが返されません。

1. レッスン 1: デバッガの起動とバグの検査 [836 ページ]
   デバッガを起動して debugger_tutorial ストアドプロシージャを実行し、バグを検査します。
2. レッスン 2: バグの診断 [838 ページ]
   debugger_tutorial ストアドプロシージャでバグを診断するには、ブレークポイントを設定してコードをステップスルーし、プロシージャの実行にともなって変数の値を監視します。
3. レッスン 3: バグの修正 [840 ページ]
   前述のレッスンで特定されたバグを修正するには、*Top_Value* 変数を初期化します。

関連情報

デバッガの稼働条件 [835 ページ]

1.11.2.1 レッスン 1: デバッガの起動とバグの検査

デバッガを起動して debugger_tutorial ストアドプロシージャを実行し、バグを検査します。

前提条件

このチュートリアルの冒頭に一覧表示されているロールと権限を持っている必要があります。

手順

1. このチュートリアルで使用するサンプルデータベースのコピーを作成します。
   a. データベースを保持するディレクトリ (`c:¥demodb` など) を作成します。
   b. 次のコマンドを実行して、データベースを作成します。

```
newdemo c:¥demodb¥demo.db
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

2. SQL Central を起動します。 スタート > プログラム > SQL Anywhere 17 > 管理ツール > SQL Central をクリックします。
3. SQL Central で、次のように demo.db に接続します。
   a. 接続 > SQL Anywhere 17 に接続 をクリックします。
   b. 接続ウィンドウで、データベースに接続するための情報を以下の項目に入力します。
      1. ユーザ ID 項目に、**DBA** と入力します。
      2. パスワード項目に、**sql** と入力します。
      3. アクションドロップダウンリストで、このコンピュータのデータベースを起動して接続をクリックします。
      4. データベースファイル項目に、**c:¥demodb¥***demo.db* と入力します。
      5. サーバ名項目に **demo_server** と入力します。
   c. 接続をクリックします。
4. モード > デバッグ をクリックします。
5. デバッグするユーザを指定してください項目に * と入力し、*OK* をクリックします。

   SQL Centrall の最下部にデバッガの詳細ウィンドウ枠が表示され、SQL Central のツールバーには一連のデバッガツールが表示されます。

   * を指定すると、すべてのユーザをデバッグできます。デバッグ対象のユーザを変更するには、いったんデバッグモードを終了して再びデバッグモードに入ります。ユーザ ID を入力すると、そのユーザ ID を使用した接続に関する情報が取得され、*[接続]* タブに表示されます。
6. SQL Central の左ウィンドウ枠で、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。
7. *debugger_tutorial (GROUPO)* を右クリックし、*Interactive SQL* から実行をクリックします。

   Interactive SQL が開き、次の結果セットが表示されます。

top_company	top_value
(NULL)	(NULL)

   この結果セットは間違っています。チュートリアルの続きで、この結果をもたらしたエラーを診断します。
8. [Interactive SQL] ウィンドウが開いている場合は閉じます。

結果

デバッガが起動して、debugger_tutorial ストアドプロシージャでバグが検出されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: デバッガの使用開始 [835 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

次のタスク: レッスン 2: バグの診断 [838 ページ]

関連情報

SQL Anywhere のデバッガ [834 ページ]

1.11.2.2 レッスン 2: バグの診断

debugger_tutorial ストアドプロシージャでバグを診断するには、ブレークポイントを設定してコードをステップスルーし、プロシージャの実行にともなって変数の値を監視します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の右ウィンドウ枠で、*debugger_tutorial (GROUPO)* をダブルクリックします。
2. *debugger_tutorial (GROUPO)* の *SQL* タブで、次の文を検索します。

```
OPEN cursor_this_customer;
```

3. ブレークポイントを追加するには、文の左側にあるグレーの縦線の領域をクリックします。ブレークポイントは、赤い円で表示されます。
4. 左ウィンドウ枠で、*[debugger_tutorial (GROUPO)]* を右クリックし、*[Interactive SQL から実行]* をクリックします。

   SQL Central の右ウィンドウ枠で、ブレークポイントの上部に黄色の矢印が表示されます。
5. **デバッガの詳細**ウィンドウで**ローカル**タブをクリックし、プロシージャ内のローカル変数とその現在の値およびデータ型のリストを表示します。*Top_Company*、*Top_Value*、*This_Value*、*This_Company* の各変数は、すべて初期化されていないため、NULL を示しています。
6. [F11] キーを押して、プロシージャをスクロールします。次の行に到達すると、変数の値が変化します。

```
IF SQLSTATE = error_not_found THEN
```

7. [F11] キーをもう 2 回押して、分岐を確認します。黄色の矢印が、次のテキストに戻ります。

```
customer_loop: loop
```

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

`IF` テストは TRUE を返しませんでした。テストが失敗したのは、NULL とどのような値を比較しても NULL が返されるためです。NULL 値によりテストは失敗し、`IF...END IF` 文内のコードは実行されませんでした。

このことから、*Top_Value* が初期化されていないことが原因とわかります。

8. プロシージャコードを変更しないで、*Top_Value* が初期化されていないことが問題であるとする仮説をテストします。
   a. **デバッガの詳細**ウィンドウで、**ローカル**タブをクリックします。
   b. *Top_Value* 変数をクリックし、**値**項目に **`3000`** と入力し、Enter を押します。
   c. *This_Value* 変数の**値**項目が 3000 よりも大きな値になるまで F11 キーを繰り返し押します。
   d. ブレークポイントをクリックして、グレーにします。
   e. [F5] キーを押して、プロシージャを実行します。

      [Interactive SQL] ウィンドウが再び表示されて、正しい結果が表示されます。

top_company	top_value
Chadwicks	8076

   f. [Interactive SQL] ウィンドウが開いている場合は閉じます。

結果

仮説が正しいことが確認されました。*Top_Value* 変数が初期化されていないことが原因でした。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: デバッガの使用開始 [835 ページ]

前のタスク: レッスン 1: デバッガの起動とバグの検査 [836 ページ]

次のタスク: レッスン 3: バグの修正 [840 ページ]

関連情報

ブレークポイント [841 ページ]
ブレークポイントの設定 [841 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.11.2.3 レッスン 3: バグの修正

前述のレッスンで特定されたバグを修正するには、*Top_Value* 変数を初期化します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. モード > 設計 をクリックします。
2. 右ウィンドウ枠で、次の文を検索します。

```
OPEN cursor_this_customer;
```

3. *Top_Value* 変数を初期化する次の行を下に入力します。

```
SET top_value = 0;
```

4. ファイル > 保存 をクリックします。
5. プロシージャを再度実行し、Interactive SQL に正しい結果が表示されることを確認します。
6. [Interactive SQL] ウィンドウが開いている場合は閉じます。

結果

バグが修正され、プロシージャが期待どおりに実行されます。これでデバッグに関するチュートリアルは終了です。

次のステップ

このチュートリアルで使用したサンプルデータベースのコピーが含まれているディレクトリ (`c:¥demodb` など) を削除します。

タスクの概要: チュートリアル: デバッガの使用開始 [835 ページ]

前のタスク: レッスン 2: バグの診断 [838 ページ]

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

関連情報

[デバッガによる変数の変更 [844 ページ]]
[変数値の表示 [845 ページ]]

1.11.3 ブレークポイント

ブレークポイントでは、デバッガがソースコードの実行を中断するタイミングを制御します。

[デバッグ] モードで実行し、接続がブレークポイントに到達した場合、その動作は選択した接続によって変わります。

- 接続を選択していない場合は、接続は自動的に選択され、プロシージャのソースコードが表示されてデバッグされます。
- 接続が選択済みで、その接続がブレークポイントに到達した接続と同じである場合は、プロシージャのソースコードが表示されます。
- 接続は選択済みですが、その接続がブレークポイントに到達した接続とは異なる場合、ウィンドウが開き、ブレークポイントに到達した接続に変更するように求めるメッセージが表示されます。

このセクションの内容:

ブレークポイントの設定 [841 ページ]
ブレークポイントを設定して、指定した行で実行を中断するタイミングをデバッガに指示します。デフォルトでは、ブレークポイントはすべての接続に適用されます。

ブレークポイントのステータスの変更 [843 ページ]
SQL Central におけるブレークポイントのステータスの変更

ブレークポイント条件の編集 [843 ページ]
ブレークポイントに条件を追加して、特定の条件または回数を満たす場合だけそのブレークポイントで実行を中断するよう、デバッガに指示を出します。

1.11.3.1 ブレークポイントの設定

ブレークポイントを設定して、指定した行で実行を中断するタイミングをデバッガに指示します。デフォルトでは、ブレークポイントはすべての接続に適用されます。

前提条件

SA_DEBUG システムロールが必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

手順

1. SQL Central で、*SQL Anywhere 17* プラグインを使用してデータベースに接続します。
2. **モード** > **デバッグ** をクリックします。
3. **デバッグするユーザを指定してください**フィールドに、* を入力してすべてのユーザをデバッグするか、デバッグ対象となるデータベースユーザの名前を入力します。

オプション	アクション
SQL Central 右ウィンドウ枠	1. 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て、プロシージャを選択します。 2. 右ウィンドウ枠で、ブレークポイントを挿入する行をクリックします。 クリックした行にカーソルが表示されます。 3. F9 キーを押します。 コード行の左側に赤い円が表示されます。
デバッグメニュー	1. **デバッグ** > **ブレークポイント** をクリックします。 2. **新規**をクリックします。 3. **プロシージャ**リストからプロシージャを選択します。 4. 必要に応じて、*[条件]* フィールドと *[カウント]* フィールドにも入力します。 条件とは、ブレークポイントが処理を中断するために TRUE に評価されなければならない SQL 式です。 カウントとは、実行が停止するまでの、ブレークポイントのヒット数です。0 を指定した場合、そのブレークポイントで常に実行が停止されます。 5. *OK* をクリックします。ブレークポイントが、プロシージャ内の最初の実行可能な文に設定されます。

結果

ブレークポイントが設定されます。

例

特定のユーザによって作成された接続に適用されるブレークポイントを設定します。それには、次のような条件を入力します。

```
CURRENT USER = 'user-name'
```


© 2016 SAP SE or an SAP affiliate company.All rights reserved.

1.11.3.2 ブレークポイントのステータスの変更

SQL Central におけるブレークポイントのステータスの変更

前提条件

SA_DEBUG システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere 17* プラグインを使用してデータベースに接続します。
2. モード > デバッグ をクリックします。
3. 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て、プロシージャを選択します。

オプション	アクション
SQL Central 右ウィンドウ枠	右ウィンドウ枠で、編集する行の左側にあるブレークポイントインジケータをクリックします。ブレークポイントがアクティブから非アクティブに変わります。
ブレークポイントウィンドウ	1. デバッグ > ブレークポイント をクリックします。 2. ブレークポイントを選択し、*[編集]*、*[無効にする]*、または *[削除]* をクリックします。 3. 閉じるをクリックします。

結果

ブレークポイントのステータスが変更されます。

1.11.3.3 ブレークポイント条件の編集

ブレークポイントに条件を追加して、特定の条件または回数を満たす場合だけそのブレークポイントで実行を中断するよう、デバッガに指示を出します。

前提条件

SA_DEBUG システムロールが必要です。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

コンテキスト

プロシージャとトリガについては、SQL 探索条件にします。

手順

1. SQL Central で、*SQL Anywhere 17* プラグインを使用してデータベースに接続します。
2. モード > デバッグ をクリックします。
3. 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て、プロシージャを選択します。
4. デバッグ > ブレークポイント をクリックします。
5. 編集するブレークポイントを選択し、*[編集]* をクリックします。
6. 条件リストで、条件をクリックします。たとえば、特定のユーザ ID からの接続にのみ適用するようにブレークポイントを設定するには、次の条件を入力します。

   ```
   CURRENT USER='user-name'
   ```

   `user-name` は、ブレークポイントをアクティブにする対象のユーザ ID です。
7. *OK* をクリックしてから、閉じるをクリックします。

結果

条件がブレークポイントに設定されます。

1.11.4 デバッガによる変数の変更

デバッガでは、コードをステップしながら変数の動作を表示して編集できます。

デバッガには、ストアドプロシージャで使用するさまざまな種類の変数を表示するデバッガの詳細ウィンドウ枠が用意されています。デバッガの詳細ウィンドウ枠は、SQL Central をデバッグモードで実行しているときに、SQL Central の最下部に表示されます。

デバッガのグローバル変数

グローバル変数はデータベースサーバによって定義され、現在の接続、データベース、その他の設定に関する情報がグローバル変数に格納されます。

グローバル変数は、グローバルタブのデバッガの詳細ウィンドウ枠に表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

ロー変数は、トリガで、トリガ元の文が対象とするローの値を保持するために使用します。ロー変数は、*[ロー]* タブの *[デバッガの詳細]* ウィンドウ枠に表示されます。

静的変数は Java クラスで使用します。静的変数は**静的**タブに表示されます。

このセクションの内容:

変数値の表示 [845 ページ]
SQL Central で変数値を表示します。

コールスタックの表示 [846 ページ]
ネストされたプロシージャをデバッグするときに、行われた呼び出しの順番を検査します。

関連情報

トリガ [107 ページ]

1.11.4.1 変数値の表示

SQL Central で変数値を表示します。

前提条件

SA_DEBUG システムロールが必要です。

さらに、EXECUTE ANY PROCEDURE システム権限、またはそのプロシージャに対する EXECUTE 権限が必要です。

手順

1. SQL Central で、*SQL Anywhere 17* プラグインを使用してデータベースに接続します。
2. **モード** > **デバッグ** をクリックします。
3. **デバッグするユーザを指定してください**フィールドに、***** を入力してすべてのユーザをデバッグするか、デバッグ対象となるデータベースユーザの名前を入力します。
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て、プロシージャを選択します。
5. 右ウィンドウ枠で、ブレークポイントを挿入する行をクリックします。

   クリックした行にカーソルが表示されます。
6. F9 キーを押します。

   コード行の左側に赤い円が表示されます。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

7. デバッガの詳細ウィンドウ枠で、ローカルタブをクリックします。
8. 左ウィンドウ枠で、プロシージャを右クリックし、*[Interactive SQL から実行]* をクリックします。
9. ローカルタブをクリックします。

結果

変数は、値とともに表示されます。

1.11.4.2 コールスタックの表示

ネストされたプロシージャをデバッグするときに、行われた呼び出しの順番を検査します。

前提条件

SA_DEBUG システムロールが必要です。

さらに、EXECUTE ANY PROCEDURE システム権限、またはそのプロシージャに対する EXECUTE 権限が必要です。

コンテキスト

プロシージャのリストは、コールスタックタブで参照できます。

手順

1. SQL Central で、*SQL Anywhere 17* プラグインを使用してデータベースに接続します。
2. モード > デバッグ をクリックします。
3. デバッグするユーザを指定してくださいフィールドに、* を入力してすべてのユーザをデバッグするか、デバッグ対象となるデータベースユーザの名前を入力します。
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て、プロシージャを選択します。
5. 右ウィンドウ枠で、ブレークポイントを挿入する行をクリックします。

   クリックした行にカーソルが表示されます。
6. F9 キーを押します。

   コード行の左側に赤い円が表示されます。
7. デバッガの詳細ウィンドウ枠で、ローカルタブをクリックします。

 © 2016 SAP SE or an SAP affiliate company.All rights reserved.

8. 左ウィンドウ枠で、プロシージャを右クリックし、*[Interactive SQL から実行]* をクリックします。
9. **デバッガの詳細**ウィンドウ枠で、**コールスタック**タブをクリックします。

結果

コールスタックタブにプロシージャの名前が表示されます。リストの一番上に現在のプロシージャが表示されます。そのプロシージャを呼び出したプロシージャがそのすぐ下に表示されます。

1.11.5 接続とブレークポイント

SQL Central の**接続**タブには、データベースへの接続が表示されます。

常に複数の接続が実行されています。あるものはブレークポイントで停止し、あるものは停止していません。

接続を切り替えるには、*[接続]* タブで接続をダブルクリックします。

ブレークポイントを設定して、単一のユーザ ID に対して実行を中断するようにすると便利です。このためには、次の形式でブレークポイント条件を設定します。

```
CURRENT USER = 'user-name'
```

SQL の特別値、CURRENT USER には接続のユーザ ID が保持されます。

関連情報

ブレークポイント条件の編集 [843 ページ]

1.12 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。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

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

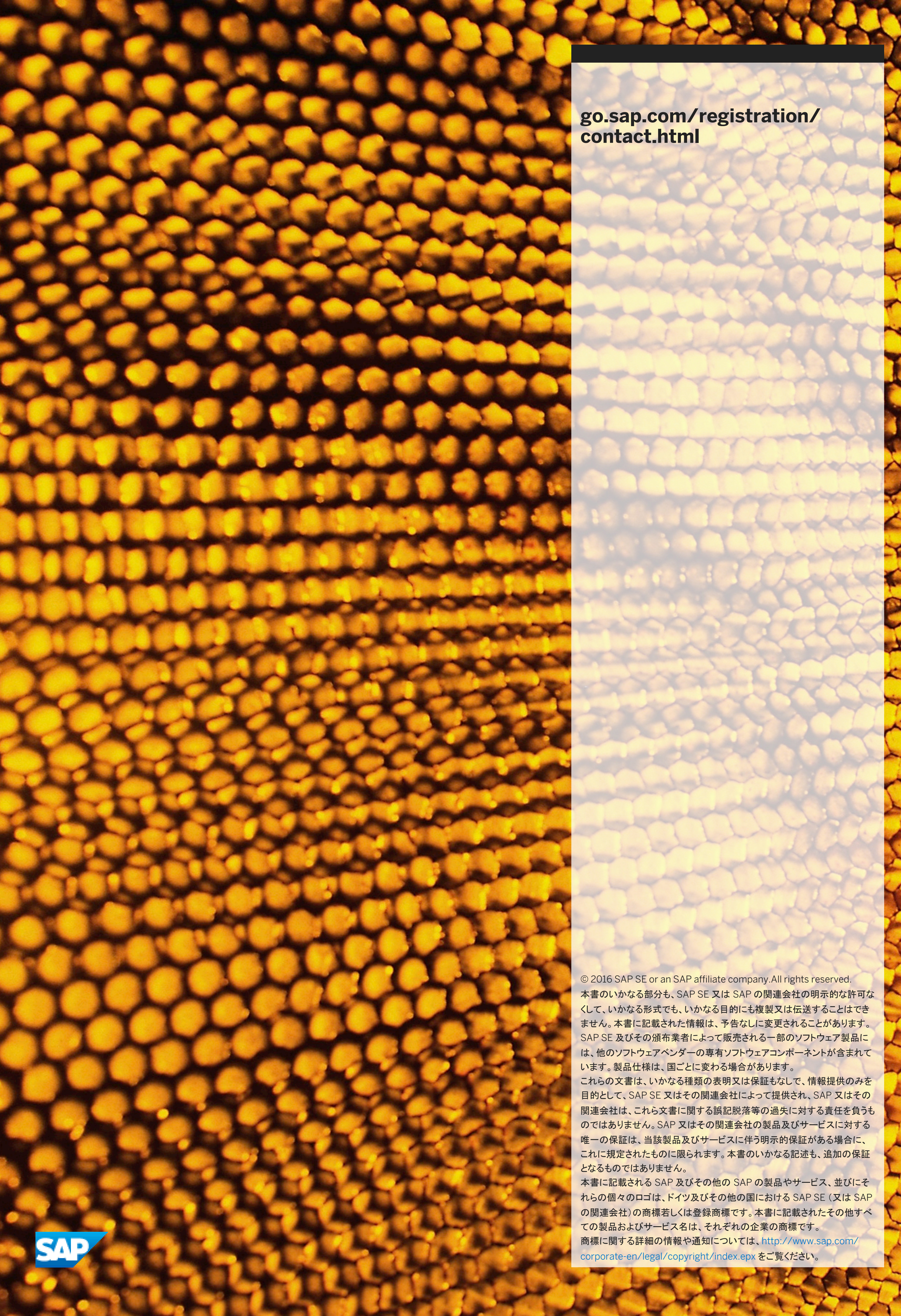
SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例:「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

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)。

© 2016 SAP SE or an SAP affiliate company.All rights reserved.

© 2016 SAP SE or an SAP affiliate company.All rights reserved.



go.sap.com/registration/contact.html

© 2016 SAP SE or an SAP affiliate company.All rights reserved.
本書のいかなる部分も、SAP SE 又は SAP の関連会社の明示的な許可なくして、いかなる形式でも、いかなる目的にも複製又は伝送することはできません。本書に記載された情報は、予告なしに変更されることがあります。
SAP SE 及びその頒布業者によって販売される一部のソフトウェア製品には、他のソフトウェアベンダーの専有ソフトウェアコンポーネントが含まれています。製品仕様は、国ごとに変わる場合があります。
これらの文書は、いかなる種類の表明又は保証もなしで、情報提供のみを目的として、SAP SE 又はその関連会社によって提供され、SAP 又はその関連会社は、これら文書に関する誤記脱落等の過失に対する責任を負うものではありません。SAP 又はその関連会社の製品及びサービスに対する唯一の保証は、当該製品及びサービスに伴う明示的保証がある場合に、これに規定されたものに限られます。本書のいかなる記述も、追加の保証となるものではありません。
本書に記載される SAP 及びその他の SAP の製品やサービス、並びにそれらの個々のロゴは、ドイツ及びその他の国における SAP SE（又は SAP の関連会社）の商標若しくは登録商標です。本書に記載されたその他すべての製品およびサービス名は、それぞれの企業の商標です。
商標に関する詳細の情報や通知については、http://www.sap.com/corporate-en/legal/copyright/index.epx をご覧ください。