WHITE PAPER

Transact-SQL 外部ジョインのセマンティックと互換性

概要

Transact-SQL の外部ジョインは、特殊な比較演算子 '* =' と '= *' を使用して指定します。1 つのクエリでこれらの演算子を 1 つ (または複数) 使用すると、From 句の 2 つのテーブル間の外部ジョインを指定できますが、そのようなクエリのセマンティックは不明瞭な場合があります。これは、TSQL 外部ジョインのセマンティックは正式に定義されたことがなく、旧バージョンの Adaptive Server Enterprise および SQL Anywhere Studio の Adaptive Server Anywhere では、TSQL 外部ジョイン・クエリの結果がオプティマイザで選択されたアクセス・プランに左右されることがありうるためです。この iAnywhere Solutions テクニカル・ホワイトペーパーでは、TSQL 外部ジョインのセマンティックについて詳しく説明し、特に ASE と ASA の TSQL 外部ジョインのサポート上の相違点を概説します。この相違点があるため、iAnywhere Solutions では、お客様のアプリケーションで Transact-SQL 外部ジョイン構文ではなく ANSI 外部ジョイン構文を使用することをお奨めします。

E-BUSINESS ANYWHERE

目次

1 序章

左右 2 つの外部ジョインは非常に便利な関係演算子で、その特性は、内部ジョインと類似しているものの、実はかなり異なります。

例 1
次のシンプルな左外部ジョイン・クエリについて考えます。

```
Select *
From R Left Outer Join S On ( C )
```

C は検索条件を示します。この左外部ジョインの結果は次のように説明できます。

1. R の各ローに関して、条件 C を満たすローを S から検索します。

2. *S* にそのようなローが 1 つ以上見つかった場合、*R* の中の該当するローとそれに対応する *S* の中のローの組み合わせをそれぞれ出力します。

3. *C* が *true* となるローが *S* の中に存在しない場合は、*R* の属性で構成される結果ローを出力し、該当のない *S* の各々の属性にNULLを代用します。

R のローはすべて結果に表示されるため、テーブル *R* を*保護テーブル*といいます。同様に、*S* の中のローに条件 *C* を満たすものがない場合は NULL 値が *S* のローに代入されるため、テーブル *S* を *null 入力テーブル*といいます。左外部ジョインは、オペランドをリバースするだけで右外部ジョインと等価になります。

上記の簡単な例では、ANSI 定義のジョイン構文を使用しています。一方、Transact-SQL (TSQL) 外部ジョインは、クエリの Where 句で *= または =* の比較演算子を使用して指定します。この比較演算子は、クエリの From 句にある 2 つの限定詞 (ベース・テーブルまたはビュー) 間の左右の外部ジョインをそれぞれ示しています。Transact-SQL 外部ジョインはもともと、初期にリリースされた Sybase SQL Server (当時の名前) で実装されていましたが、これは、外部ジョインのセマンティックが理論的見地から十分に理解される前のことです。

Transact-SQL 外部ジョインの表記には、次の 4 つの具体的な問題があります。

1. 構文で On 条件の述部と Where 句の述部を (ANSI 用語を使用して) 区別できないため、セマンティックがユーザにとって不明瞭な場合がある。

2. 外部ジョイン条件は等式条件でなければならず、その他の比較演算子は使用できない。

3. ネスト外部ジョインを、ビューを参照せずに指定することができない (これは、ASE 11.5 ではサポートされているものの、Adaptive Server Anywhere ではまだサポートしていない。第 8 項を参照)。

4. Transact-SQL 構文を使用してすべての外部ジョインを指定することができない。

これらのセマンティック問題の他に、TSQL 外部ジョインのセマンティックに関する混乱には次の 2 つの要因もあ

ります。

- バージョン 12.0 より前の ASE およびバージョン 7.0 より前の ASA では、TSQL 外部ジョインを含むクエリで返される結果が、クエリ用に選択した最適化方式によって異なる場合がある。

- ASE および ASA のさまざまなリリースで実装内容が変更されたことにより、実行可能で有効な TSQL 外部ジョイン・クエリのクラスが拡張されると同時に制限されてきた。

このテクニカル・ホワイトペーパーでは、ASE および ASA のさまざまなリリースに対する Transact-SQL 外部ジョインの動作概要を、具体例を用いて説明します。明示的な On 条件で `Left Outer Join` (左外部ジョイン) または `Right Outer Join` (右外部ジョイン) を使用して、TSQL 外部ジョインを等価の ANSI 構文に変換することにより、セマンティックを説明します。このような書き換えは、ASE 12 以上および ASA 7 以上のリリースで現在 TSQL 外部ジョインに使用されている処理と全く同じです。この書き換えは最適化前に実行されるため、クエリ結果はオプティマイザが選択するアクセス・プランの影響を受けません。ただし、この 2 つの製品の間には、特定クラスのクエリの変換方法に、TSQL 外部ジョインの動作のセマンティック上の違いが依然としてあります。

ANSI/ISO の SQL-99 規格に定義される ANSI セマンティックを (大まかに) まとめると、次の句 (一部オプション)

```
Select [Distinct ]
From
[Where ]
[Group By ]
[Having ]
```

を含むクエリ仕様のセマンティックは、次の処理を行います。

1. `From` 句の各テーブル式の拡張直積を計算します。ただし、ANSI ジョイン構文を使用すると、ANSI 外部ジョインが `From` 句に表示されるため、ANSI 外部ジョインの評価もこの手順に含まれます。

2. 抽出されたローがそれぞれ `Where` 句で指定されている検索条件を満たす (つまり、条件が *true* となる) ように、(1) の結果を制限します。*null を許容しない*述部[1]が null 入力テーブルの属性を参照する場合、`Null` 入力ローが結果から削除されます。この場合、クエリは内部ジョインを使用して書き換えられたクエリと等価になります。

3. `Group By` が指定されている場合は、`Group By` 句に含まれている式から一意に組み合わせたものとして各グループを定義して、(2) の結果をローのグループに分割します。`Group By` 句が指定されていない場合は、(2) の結果は単一の「グループ」として処理されます。クエリの `Select` 句や `Having` 句に含まれている集合関数を、各グループに関して評価できます。

4. `Having` 句が存在する場合は、各グループの属性値が `Having` 句で指定されている検索条件を必ず満た

[1] null を許容しない述部とは、入力内容に Null が 1 つでも含まれていると *true* と評価できない述部です。比較、Like、In など、ほとんどの SQL 述部は null を許容しません。null を許容する述部の例として、IsNull や、null を許容する真理値テストで修飾された述部 *p* (*p* Is Not True など) があります。

すように、(3) の結果を制限します。

5. クエリの Select リストにある属性に結果を反映します。

6. Distinct が指定されている場合は、重複している結果ローを削除します。

2 単純な TSQL 外部ジョイン

Transact-SQL 外部ジョインの最も重要な問題は、null 入力テーブルを参照する追加の Where 句の述部の処理を決定することです。これらの述部が外部ジョインの On 条件の一部として、またはクエリの Where 句の一部として処理されるかどうかは、いくつかの要因によって決まります。この議論は同じブロック (クエリまたはビュー) で発生する述部のみに該当します。セマンティック上は、Transact-SQL はすべてのビューを実体化されたテーブルとして処理します。

TSQL 外部ジョインのもともとの実装では、Where 句内の述部を外部ジョインの On 条件の一部として処理するかどうかは、オプティマイザが決定するクエリのアクセス・プランへの述部の配置方法によって異なります。このことは、12.0 より前のバージョンの ASE および 7.0 より前のバージョンの ASA に該当します。一般的に、このアプローチでは一貫した結果が保証されませんでした。ただし、実際は、両方のクエリ・オプティマイザの 2 つの特性により、ほとんどのクエリで一貫性が維持されていました。

1. 有効なアクセス・プランでは、保護テーブルが null 入力テーブルの前になければならない。

2. 述部はできる限り早い時期に評価される。つまり、述部はプランの中で評価可能な最も早い時点に配置され、(別のジョインの後などに) 遅延されない。

単純な TSQL 外部ジョイン・クエリのセマンティックを説明するために、次のスキーマを使用する例について考えます。

```
Create Table T (aIntPrimary Key,b Int,c Int );
Create Table R (xIntPrimary Key,y Int,z Int );
Create Table S (lIntPrimary Key,m Int,n Int );
Create Table W (dIntPrimary Key,e Int,Int );

Insert Into T Values(1,2,3 );
Insert Into T Values(2,4,5 );
Insert Into T Values(3,4,5 );
Insert Into R Values(3,4,5 );
Insert Into S Values(3,0,0 );
Insert Into W Values(3,4,5 );
```

例 2
次のクエリについて考えます。

```
Select *
From R,S
Where R.x * = S.l and S.m > 5.
```

この、TSQL 外部ジョイン・クエリの典型的な例は、次の ANSI 表現とセマンティック上は等価です。

```
Select *
From R Left Outer Join S On(R.x =S.l and S.m > 5).
```

この書き換えられたクエリには Where 句がないことに注目してください。*S.m* に関する追加条件が、外部ジョインの On 条件の一部となっています。*R* に関する追加の結合条件[2]はすべて、Where 句の一部とみなされます。ただし、*R* 上の述部の中に、*S* 上の述部と共に離接句に参加しているものがある場合は、その離接句は On 条件の一部となります。

例 3
次のクエリ

```
Select *
From R,S
Where R.x = S.l and R.y = 15
```

は、次のクエリと等価です。

```
Select *
From R Left Outer Join S On (R.x =S.l )
Where R.y =15.
```

[2] *離接句*は、*or* を使用して結合されている個々の述部から成る論理式であり、*離接条件*とは離接句の中の条件です。同様に、*結合句*は、*and* を使用して結合されている個々の述部から成る論理式であり、*結合条件*とは結合句の中の条件です。

例 4
次のクエリ

```
Select *
From R,S
Where R.x = S.l and ( S.m > 5 or R.x = 15 )
```

は、次のクエリと等価です。

```
Select *
From R Left Outer Join S On ( R.x = S.l and ( S.m > 5 or R.x = 15 ) ).
```

ASE および ASA の初期バージョンでは評価が可能になるとすぐに述部がアクセス・プランに配置されていたことに気づいていれば、これら 2 つの例は直感的に理解できます。TSQL 外部ジョインの null 入力側のテーブルに隣接して述部が配置された場合、述部はその外部ジョインの On 条件の一部となります。したがって、個々の述部、または句の中で、null 入力テーブルのカラムを参照しないものは、ANSI 表現で外部ジョインの On 条件の一部となることはありません。

3 TSQL 外部ジョインとより複雑な検索条件

より複雑な条件を使用すると、旧バージョンの ASE および ASA では、クエリに他のテーブルが関連している場合に動作が不確定になる可能性があります。これは、述部の配置に基づいて TSQL 外部ジョインのセマンティックを定義する処理にあいまいさが残っているためです。この現象が発生する可能性のあるケースの数を制限するため、ASE 11.0 以上では、TSQL 外部ジョインと内部ジョインの両方に関連するテーブルを含むクエリは拒否されます。

例 5
次のクエリ

```
Select *
From R,S,T
Where R.x = S.l and S.m = T.a
```

は、ASE 11 の全リリースでエラー (303,16) を、ASE 12.0 ではエラー (11054,16) を返します。ASA 7 および 8.0で は SQLCODE -680 を返します。ほとんどの場合、このエラーが妥当なのは、誤解を防ぐ場合のみです。正当であれば、上記のクエリは内部ジョインのみを含むクエリと等価です。これは、テーブル *S* とテーブル *T* の間に null を許容しないジョインの述部があるためです。

ただし、null 入力テーブルに対する結合内部ジョイン条件は許可されていなくても、離接句で null 入力テーブルが参照されているときに不確定な動作が発生する可能性はあります。

例 6
次のクエリについて考えます。

```
Select *
From R,S,T
Where R.x = S.l and R.x =T.a and ( T.b = 0 or S.m = 3 ).
```

オプティマイザが選択するジョイン方法によって、Where 句の最後の離接述部が On 条件の一部になる場合とならない場合があります。この例では、ジョイン方法が次の ASE 12.0 抽象プラン表記に対応する *R S T* である場合、

```
1   ( nl _ g _ join
2     ( t _ scan R )
3     ( t _ scan S )
4     ( t _ scan T )
5   )
```

最後の離接条件は Where 句の一部として処理されます。ANSI 構文では、これは次の構文と等価になります。

```
Select *
From R Left Outer Join S On( R.x = S.l ), T
Where R.x = T.a and ( T.b = 0 or S.m = 3 ).
```

***R*、*S*、*T* の各テーブルに上記のように入力すると、ASE 11.5 および 11.9.2 はジョイン方法 *R S T* を選択します。上記のような Where 句を使用した場合、クエリは空のセットを返します。ただし、カラム *T.a* に冗長な条件を追加するようにこのクエリを変更すると、次のクエリ**

```
Select *
From R, S, T
Where R.x * = S.l and R.x = T.a and
      T.a = 3 and ( T.b = 0 or S.m = 3 )
```

の結果は、次のようになります。

結果

x	y	z	l	m	n	a	b	c
3	4	5	NULL	NULL	NULL	3	4	5

これは、このクエリに関して ASE 11.5 および 11.9.2 はジョイン方法 *T R S* を選択し、この方法が次の ASE 12.0 抽象プラン表記に対応するためです。

```
6    ( nl _ g _ join
7      ( t _ scan T )
8      ( t _ scan R )
9      ( t _ scan S )
10   )
```

実際に、ASE 11 はこのクエリを次のように解釈します。

```
Select *
From T, R Left Outer Join S On( R.x = S.l and ( T.b = 0 or S.m = 3 ) )
Where R.x = T.a and T.a = 3.
```

これには、外部ジョインの On 条件の *T* に対する外部参照を導入するという効果があります。

ASE 12.0 は、常にテーブルを含む離接句を、クエリの Where 句の一部として外部ジョインされている離接句とは別に処理します。したがって、ASE 12.0 でもこのクエリの空のセットが作成されます。一方、ASA 8.0 は、離接述部と結合述部を区別しないため、例 5、例 6 の両方のクエリで、結果を生成せずに SQLCODE -680 を返します。ASA 6 サーバ (またはそれ以前のバージョン) では、12.0 より前のバージョンの ASE と同様に、あいまいな結果になります。ただし、さまざまな要因により、アクセス・プランは時間が経つにつれて変わる場合があるため、結果は予期できません。

ここまでの例から、TSQL 外部ジョインを含む SQL 文の作成方法がクエリのセマンティックに影響することが容易に確認できます。ASE 12.0 と ASA 8.0 の両方で、TSQL 外部ジョインを等価の ANSI 表現に変換するための分析は、元の文の Where 句に基づいて行われます。これは、述部の正規化または追加条件の推論を行う前に実行されます。つまり、同じ Where 句を 2 種類に変形した場合、検索条件が表面上はセマンティック的に等価であっても、結果が異なる可能性があるということです。

例 7
一見すると、次のクエリ

```
Select *
From R, T
Where R.x * = T.a and
     ( ( T.b = 2 and R.y = 1 ) or ( T.b = 4 and R.y = 1 ) )
```

は、次のクエリとセマンティック的に等価であるように思われます。

```
Select *
From R, T
Where R.x = T.a and R.y = 1 and ( T.b = 2 or T.b = 4 ).
```

しかし、現在リリースされているすべての ASE および ASA では、1 つ目のクエリは

結果

x	y	z	a	b	c
3	4	5	NULL	NULL	NULL

という結果を返すのに対して、2 つ目のクエリは空のセットを返します。なぜでしょうか？ この現象が発生するのは、1 つ目のクエリが

```
Select *
From R Left Outer Join T On ( R.x =T.a and
      ( ( T.b = 2 and R.y = 1 ) or ( T.b = 4 and R.y = 1 ) ) )
```

と解釈されるのに対して、2 つ目のクエリは次のように解釈されるからです。

```
Select *
From R Left Outer Join T On ( R.x = T.a and ( T.b = 2 or T.b = 4 ) )
Where R.y = 1.
```

結合条件 *R.y* = 1 が 1 つ目の例から抽出されるとき、この条件は保護テーブルのみを参照します。したがって、この条件は外部ジョインの On 条件の一部とはみなされません。

4 TSQL 外部ジョインの式に使用可能な形式

一般的な TSQL 外部ジョインの述部は、ベース・テーブルのカラムまたはビューのカラムを参照します。ASA 6.0.3 ビルド番号 2829 より前は、比較対象のいずれかが複合式となっている * = (または = *) TSQL 外部ジョイン述部を含むクエリを、ASA はすべて拒否していました。現在は、それぞれの式が 1 つのテーブルだけを参照し、その式にサブクエリが含まれていなければ、ASA では、任意の式に対して TSQL 外部ジョインの条件を付けることができます。一方、ASE では、外部ジョインの述部で複合式を使用できるだけでなく、保護テーブルを参照するサブクエリを式に含めることもできます。

例 8
次のクエリ

```
Select *
From R, S
Where S.l = * ( R.x + ( Select T.a From T Where T.a = R.x ) )
```

は、ASE では正当な外部ジョインです。一方、

```
Select *
From R, S
Where R.x * = ( S.l + ( Select T.a From T Where T.a = R.x ) )
```

を実行すると、 ASE 11 ではユビキタス外部ジョイン構文エラー (303,16) が、ASE 12.0 ではエラー (11055,16) が返されます。

もちろん、TSQL 外部ジョインとサブクエリの両方を含むクエリがすべて不正というわけではありません。ネストされたクエリに関しては、下記の例 11 および 12 も参照してください。ただし、ASA 7 以上では、例 8 のどちらのクエリを実行しても SQLCODE -680 が返されます。これは、 ASA ではどちらの比較式にもサブクエリを含めることができないためです。

ASE でも ASA でも、TSQL 外部ジョイン比較述部に定数または変数を含む算術式を使用できます。ただし、TSQL 外部ジョイン比較の両側の式がいずれも単一のテーブルを参照しなければならないという点に注意してください。

例 9
次のクエリ

```
Select *
From R, S
Where R.x = ( R.y + S.l )
```

は、構文エラー (ASE 11 では (301,16)) を返すため、次のような書き方にしてください。

```
Select *
From R,S
Where ( R.x - R.y ) = S.l
```

これが正当なクエリです。

複数の外部ジョイン述部が同じテーブルを参照している場合は、次のような結合外部ジョイン条件を含む単一の外部ジョインがあると考えられます。

例 10
次のクエリ

```
Select *
From R, S
Where R.x = S.l and R.y = S.m
```

は、ANSI の次のクエリと等価です。

```
Select *
From R Left Outer Join S On ( R.x = S.l and R.y = S.m ).
```

5 TSQL 外部ジョインとネストされたクエリ

Transact-SQL 外部ジョインのセマンティック問題のもう 1 つの原因として、相関サブクエリの存在があります。特に、サブクエリの相関属性が、1 つまたは複数の null 入力テーブルから得られたものである場合です。下記の例 11 および 12 には、Exists を使用した、存在限定サブクエリ述部が含まれていますが、In、Some、Any、All のいずれかを使用する、その他の形式の限定サブクエリ述部でも同様の問題が発生します。

例 11
次のクエリについて考えます。

```
Select *
From R, S
Where R.x * = S.l and
     Exists ( Select *From T Where T.a = S.l and T.b = R.y )
```

この形式の相関サブクエリは、ASE 11.5 以前の旧バージョンの ASE で使用できました。しかし、述部の配置が原因で、例 6 に示すようなセマンティック問題が発生していました。さらに、オプティマイザを使用してサブクエリをジョインに変換すると、書き換えられた次のクエリ

```
Select *
From R, S, T
Where R.x * = S.l and T.a = S.l and T.b = R.y
```

は、例 5 の場合と同様にエラー (303,16) になります。

11.5 より後のバージョンの ASE は、TSQL 外部ジョインの null 入力側から得られた相関変数がサブクエリに含まれている場合、外部ジョイン構文エラー (303,16) を返します (ASA 7 以上は SQLCODE -680 を返す)。この制限は、どのタイプの限定サブクエリ (Any、In、Some、All) にも該当します。このような方法でサブクエリを制限すると、ネストされたクエリをジョインとして書き換える際に発生する可能性のある問題を排除することもできます。

興味深いことに、サブクエリが限定述部の一部でない場合は、この制限はネストされた相関クエリのみに適用されます。

例 12
次のクエリ

```
Select *
From T, R
Where T.a * = R.x and R.y = ( Select S.m From S Where S.l = 3 )
```

は、次の結果を返します。

結果

x	y	z	a	b	c
1	2	3	NULL	NULL	NULL
2	4	5	NULL	NULL	NULL
3	4	5	NULL	NULL	NULL

ASE の各バージョンでは、この形式のクエリを使用できます。この場合は、サブクエリ比較述部が外部ジョインの On 条件の一部となります。サブクエリをジョインに変換すると (つまり、サブクエリの Where 句に、テーブル *T* のプライマリ・キーに対する明示的または暗黙的な等価条件が含まれる)、ASE 11 はサブクエリを内部ジョインに変換するため、オプティマイザが選択したジョイン順に依存する不確定な動作が (再び) 発生します。一方、ASE 12.0 は常にサブクエリ (またはそれに相当するもの) を、外部ジョインの On 条件の一部とします。ただし、ネストされたクエリ・ブロックに相関を追加すると、再びエラー (303,16) が発生します。ASA 7 および 8.0 では、このようなクエリはすべて構文エラー -680 で拒否されます。

6 離接 TSQL 外部ジョインの条件

ASE の各バージョンでは、次の例に示すように、TSQL 外部ジョインの条件を離接句に含めることができます。

例 13
次のクエリについて考えます。

```
Select *
From R, S
Where R.x * = S.l or R.y > 5.
```

このクエリは、ANSIの次のクエリと等価です。

```
Select *
From R Left Outer Join S On ( R.x = S.l or R.y > 5 ).
```

TSQL 外部ジョインを離接句に含めることは、ASA バージョン 7 および 8 でもサポートされています。それより前のバージョンの ASA では、結合 TSQL 外部ジョインの述部のみが考慮されていました。離接句は、保護テー

ブル上の述部を外部ジョインの On 条件の一部にすることのできる唯一のメカニズムです。

7 スター外部ジョイン

ASE も ASA も*スター*外部ジョインをサポートしています。スター外部ジョインを使用すると、テーブル参照の相関名が同じであれば、単一のテーブルに対する複数の参照が同一のテーブル参照として解釈されます。

例 14
ASA では、次のクエリを発行できます。

```
Select *
From R, R
Where R.x > 15.
```

このクエリの From 句は、それ自体では *R* の直積を指定しません。もちろん ANSI 構文でもありません。ANSI の SQL-99 規格によると、このクエリは、*R* に対する 2 つの参照の相関名が同じであるため、構文エラーになります。一方、ASA は R に対する 2 つ目の参照を同一のテーブル参照として扱うため、このクエリは次のクエリと等価になります。

```
Select *
From R
Where R.x > 15.
```

このように、ASA で重複したテーブル参照を非 ANSI 処理すると、スター・ジョイン (内部および外部) の構成が、ビューに頼る必要なく、より簡単になります。次の例で説明するように、通常は、「スター」の中央にあるテーブルは複数の外部ジョインの保護テーブルです。

例 15
次のクエリ

```
Select *
From R Left Outer Join S On ( R.x = S.l ),
          R Left Outer Join T On ( R.x = T.a )
```

を、ANSI 規格の次のクエリの代わりに使用できます。

```
Select *
From (R Left Outer Join S On ( R.x = S.l ) )
       Left Outer Join T On ( R.x = T.a )
```

この機能は、外部キーの関係や同じ名前のカラムが複数あるためにあいまいさが発生する Key ジョインや Natural ジョインを使用している場合に特に便利です。

例 16
同様に、ASE と ASA の両方で、TSQL 外部ジョインを使用するスター・ジョインを指定できます。

```
Select *
From R, S, T
Where R.x * = S.l and R.x * = T.a
```

ここでは、*R* に対する参照が 1 つしか存在しないことは From 句から明らかですが、関連付け対象のテーブルが異なる 2 つの外部ジョイン述部があるため、外部ジョインは 2 つ存在します (例 10 のクエリと比較)。

複数回参照されるテーブルが外部ジョインの null 入力テーブルである場合は、ASE と ASA ではサポートが異なります。

例 17
次の例について考えます。

```
Select *
From S Left Outer Join R On ( R.x = S.l ),
           T Left Outer Join R On ( R.x = T.a )
```

この外部ジョインには問題があります。テーブル *R* に対する参照が重複していますが、2 つの On 条件はそれぞれ独立したままです (このクエリにはまだ外部ジョインが 2 つあります)。したがって、ASA では、このクエリは構文エラー (SQLCODE -137、テーブル *'R'* にはユニークな相関名が必要) を返します。

例 18
ASE では、上記の例と等価である Transact-SQL 構文

```
Select *
From R,S,T
Where S.l * = R.x and T.a * = R.x
```

は、次に示す等価の ANSI 表現として処理 (ASE 11 の場合) されるか、または等価の ANSI 表現に変換 (ASE 12.0 の場合) されます。

```
Select *
From ( S Cross Join T ) Left Outer Join R On ( R.x = S.l and R.x = T.a )
```

これには外部ジョインが 1 つしか含まれていないため、例 17 のクエリとは明らかに異なります。このようなクエリは、Adaptive Server Anywhere のどのバージョンでもサポートされていません。

8 連鎖 (ネスト) 外部ジョイン

どの From 句にも重複したテーブル参照が存在する場合があるため、ASE と ASA では、同じテーブルが同じクエリ・ブロックで保護テーブルと null 入力テーブルの両方になっている外部ジョイン・サイクルが存在するかどうか確認する必要があります。ASA の各バージョンは、サイクルが存在する場合に特定のメッセージ (SQLCODE -136) を返すのに対して、バージョン 11 以前の ASE は標準の外部ジョイン構文エラー (301,16) を返します。

ASE 11.02 では、ネスト外部ジョインをビューを使用せずに指定することはできません。ビューでは、ネスト外部ジョインが左右のどちら側が深いネストかを特定する明示的なメカニズムが提供されています。ASE 11.5 では、*連鎖* TSQL 外部ジョインが導入され、1 つのテーブルが保護テーブル、null 入力テーブルの両方として 2 つ以上の外部ジョインに関与できるようになりました。次の例に示すように、外部ジョインは左側が深いネストと考えられます。

例 19
Adaptive Server Enterprise 11.5 以上のバージョンでは、

```
Select *
From R, S, T
Where R.x * = S.l and S.m * =T.b
```

を、次の ANSI 表現と等価であると解釈します。

```
Select *
From ( R Left Outer Join S On ( R.x = S.l ) )
        Left Outer Join T On ( S.m = T.b ).
```

上記の例から、連鎖外部ジョインを使用すると、テーブル *S* を参照する条件の配置によってあいまいさが生じるため、セマンティック・エラーが発生する可能性が大幅に高くなることは明らかです。このため、ASA は連鎖外部ジョインをサポートせず、このようなクエリに対しては SQLCODE -680 を返します。ASE 12.0 は、このようなクエリを ANSI 構文に変換し、該当する述部をネスト外部ジョイン式の「最も深い」レベルに配置して、ASE 11 で使用されている述部のプッシュダウン・テクニックと同様の動作を一貫して行っています (つまり、選択した最適化方式は結果に反映されません)。

9 まとめ

現在リリースされている ASA および ASE のバージョンでは、Transact-SQL 外部ジョインのセマンティックはオプティマイザが選択したアクセス・プランの影響を受けませんが、iAnywhere Solutions は、TSQL 外部ジョインの使用をお奨めしません。その理由は、セマンティックの定義がまだ不十分である (また、ほとんどの部分が文書化されてない) ことと、ASE 12 が ANSI ジョイン構文をサポートすることです。iAnywhere Solutions は、TSQL 外部ジョインに対する異議をまだ発表していませんが、この発表に先立って、アプリケーションで使用する構文を ANSI ジョイン構文に切り替えることを、すべてのお客様に強くお奨めします。

A ANSI 構文への自動変換

Rewrite 関数 [1] を使用すると、ASA サーバが TSQL 外部ジョインを ANSI 構文に変換する方法を憶測で特定する必要がなくなります。Adaptive Server Anywhere 7.0 で導入されたこの関数は、SQL 文を引数として取り、書き換えられた SQL 文をその結果として返します。書き換えられた SQL 文の構文は次のように変形されます。

1. On 条件を利用して Key ジョインと Natural ジョインを ANSI ジョイン構文に変換する。

2. TSQL 外部ジョインを ANSI ジョイン構文に変換する。

3. 重複している相関名を削除し、From 句にネストされたテーブル式を含める。

ASA 8.0.1 から、Rewrite 関数にオプションのキーワード・パラメータ 'ANSI' を付けると、元の SQL 文に対して行う一連の変換が上記の 3 つに制限されます。このパラメータを使用しない場合、サーバは元の文に対して次の変形も適用します。

1. ネストされたクエリをジョインとして書き換える。重複を排除する手順も追加される場合もある (キーワード Distinct)。

2. ビューや抽出テーブルを参照元のクエリにマージする

3. 結果にユニークなローが含まれることが確実な場合は Distinct キーワードを排除する

4. Group by または Union を含むビューまたは抽出テーブルに述部をプッシュダウンする

5. 不要なジョインを排除し、必要に応じて他の述部を変更または挿入する

6. 内部ジョインが同じ結果を返す場合に外部ジョインを内部ジョインに変換する

7. Where 句、Having 句、On 条件の各検索条件を、完全にまたは部分的に結合正規形に正規化し、元の条件を簡略化して、クエリ実行中に取得可能な場合は推測される述部を追加する

これらのセマンティック変形は、サーバが SQL クエリや Insert 文、Update 文、Delete 文を最適化するたびに、自動的に適用されます。

例 20
例 7 のクエリに Rewrite 関数を適用した場合、

```
Select Rewrite( 'Select *
               From R, T
               Where R.x * = T.a and R.y = 1
                    and ( T.b = 2 or T.b = 4 )',
              'ANSI ' )
```

結果は次の SQL 文になります (文字列として返される)。

```
Select *
From R Left Outer Join T On ( R.x = T.a and ( T.b = 2 or T.b = 4 ) )
Where R.y = 1.
```

参考資料

[1] iAnywhere Solutions、オンタリオ州 Waterloo。『Adaptive Server Anywhere 8.0 Reference Manual』、2001 年 12 月。

法的注意

iAnywhere、iAnywhere Solutions、Adaptive Server、SQL Anywhereは、iAnywhere Solutions, Inc.またはSybase, Inc.とその系列会社の商標です。その他の商標はすべて各社に帰属します。

Mobile Linkの技術には、Certicom,Inc.より供給を受けたコンポーネントが含まれています。これらのコンポーネントは特許によって保護されています。

本書に記載された情報、助言、推奨、ソフトウェア、文書、データ、サービス、ロゴ、商標、図版、テキスト、写真、およびその他の資料(これらすべてを"資料"と総称する)は、iAnywhere Solutions,Inc.とその供給元に帰属し、著作権や商標の法律および国際条約によって保護されています。また、これらの資料はいずれも、iAnywhere Solutions,Inc.とその供給元の知的所有権の対象となるものであり、iAnywhere Solutions,Inc.とその供給元がこれらの権利のすべてを保有するものとします。

資料のいかなる部分も、iAnywhere Solutionsの知的所有権のライセンスを付与したり、既存のライセンス契約に修正を加えることを認めるものではないものとします。

資料は無保証で提供されるものであり、いかなる保証も行われません。iAnywhere Solutions,Inc.は、資料に関するすべての陳述と保証を明示的に拒否します。これには、商業性、特定の目的への整合性、非侵害性の黙示的な保証を無制限に含みます。

iAnywhere Solutionsは、資料自体の、または資料が依拠していると思われる内容、結果、正確性、適時性、完全性に関して、いかなる理由であろうと保証や陳述を行いません。iAnywhere Solutionsは、資料が途切れていないこと、誤りがないこと、いかなる欠陥も修正されていることに関して保証や陳述を行いません。ここでは、「iAnywhere Solutions」とは、iAnywhere Solutions, Inc.またはSybase,Inc.とその部門、子会社、継承者、および親会社と、その従業員、パートナー、社長、代理人、および代表者と、さらに資料を提供した第三者の情報元や提供者を表します。

＊本書は、米国iAnywhere Solutions, Inc.が作成・テストしたものを日本語に翻訳したものです。

アイエニウェア・ソリューションズ株式会社
http://www.ianywhere.jp

1017447