Microsoft SQL Server 2000 Enterprise Edition

第2部 データベース作成と操作

第1章

Transact-SQL

ここでは、SQL Serverで使用するSQL言語であるTransact-SQLの機能について記述します。

Transact-SQLの概要

Oracleで SQLを使用する場合、ユーザーと対話的な操作を可能にするSQL*Plusとストアドプロシージャを記述可能なPL/SQLの2種類を使用します。

SQL Serverでは特に区別はなく、SQL Serverが実行するSQLステートメントはすべてTransact-SQLと呼びます。

Oracleで SQLを作成するにはSQL*Plusなどを使用しますが、SQL Serverではクエリアナライザという GUIツールを使用します（図1-1）。また、コマンドインタープリタとしてはosqlユーティリティがあります。

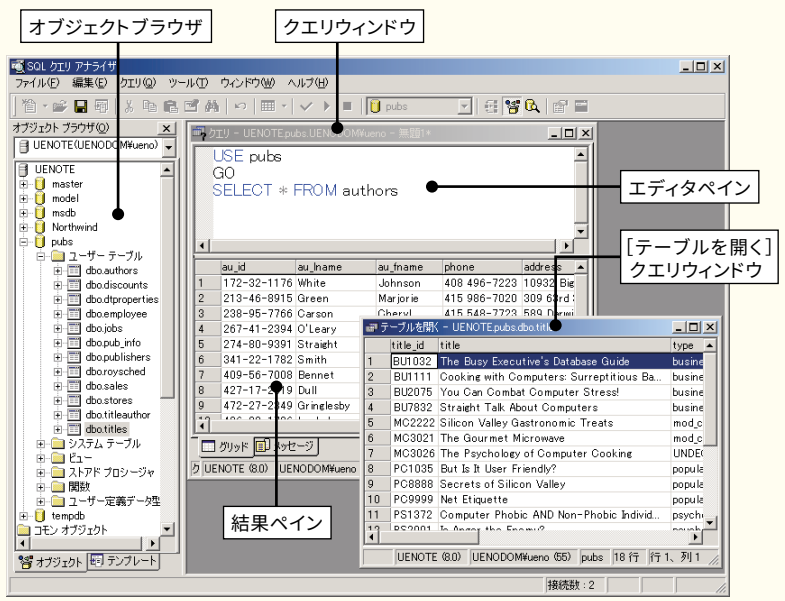

図1-1
クエリアナライザ

クエリアナライザの特徴は、以下のとおりです。

- 自由な形式のテキストエディタ（クエリウィンドウ）を使用してTransact-SQLステートメントを作成できます。
- テキストエディタ内ではキーワード（SELECT、FROMなど）、コメント、文字列などが自動的に色分けされてTransact-SQLステートメントを読みやすくします。
- Transact-SQLステートメントの実行結果を、Transact-SQLステートメントを入力したペイントは別のペイン（結果ペイン）に表示します。このと

き、Excelシートのようなグリッド表示を行うことにより、結果がより見やすくなります。

- オブジェクトブラウザにはデータベース内のオブジェクトが表示されるので、Transact-SQLステートメントを間違えることなく効率的に作成できます。
- ストアドプロシージャなどの複雑なTransact-SQLステートメントを対話的にデバッグできます。
- 実行プランをグラフィカルに表示され、Transact-SQLステートメントの分析が簡単に行えます。

Transact-SQLステートメントの実行

Transact-SQLステートメントの実行プロセスは、以下のようになります。

① **解析**　：Transact-SQLステートメントの構文が正確かどうか調べます。
② **解決**　：オブジェクト名が存在するかを確認します。
③ **最適化**　：使用するインデックスおよび結合方法を決めます。
④ **コンパイル**　：実行可能な形式（実行プラン）に変換します。
⑤ **実行**　：実行します。

通常のSQLステートメントは、実行時に毎回上記のプロセスを繰り返しますが、以下の場合はメモリ上の実行プランを再利用します。

- 1回目のバッチと同じテキストが送られた場合
- SQL Serverがパラメータと判断したもの以外が同じと判断した場合
- sp_executesqlあるいはパラメータマーカーを使用して実行した場合
- ストアドプロシージャを実行した場合

バッチ

Oracleでは、SQLステートメントの終端に区切り文字（通常はセミコロン）を記述する必要があります。通常、OracleのSQLステートメントは、ステートメントごとにコンパイルされますが、PL/SQLではブロック単位にコンパイルされます。

SQL ServerにはTransact-SQLステートメントの区切り文字はありませんが、バッチという便利な機能があり、Transact-SQLステートメントをまとめてコンパイルして、実行プランを作成できます。

バッチは以下の状況で構成されます。

- 1つのアプリケーションからまとめてSQL Serverに送られるTransact-SQLステートメントの集まり（クエリアナライザやosqlユーティリティでは、GOステートメントを使用してバッチを区切ることができます）
- 1つのストアドプロシージャまたはトリガ内のTransact-SQLステートメントの集まり
- EXECUTEステートメントまたはsp_executesqlシステムストアドプロシージャが実行するコマンド

バッチには以下のルールがあります。

- CREATE DEFAULT、CREATE PROCEDURE、CREATE RULE、CREATE TRIGGER、CREATE VIEWステートメントを、ほかのステートメントと同一のバッチ内に含めることはできません。バッチは、CREATEステートメントで始める必要があります。そのバッチのCREATE以降のステートメントはすべて、CREATEステートメントの定義の一部として解釈されます。
- テーブルを変更してから、同じバッチの中で新規の列を参照することはできません。
- EXECUTEステートメントがバッチ内の最初のステートメントである場合、EXECUTEキーワードは不要です。EXECUTEステートメントがバッチ内の最初のステートメントではない場合は、EXECUTEキーワードが必要です。

オブジェクトの参照

Oracleでのオブジェクトを参照するには、サーバー名、インスタンス名、スキーマ名、オブジェクト名が必要となります。実際のSQLステートメントで記述する場合の構文は、以下のようになります。

schema.objectname@netservicename

*netservicename*は、上記のサーバー名、ホスト名、プロトコル、リスナーポート番号、インスタンス（SID）を示します。ユーザーが接続しているデータベース内のオブジェクトを参照する場合は、*@netservicename*は省略可能で、同一スキーマのものであればスキーマ名も省略可能です。また、別スキーマへの参照の簡略化のためにシノニム（「別名」という意味です）というオブジェクトが用意されています。

SQL Serverでのオブジェクトの参照の構文は、以下のようになります。

<インスタンス名>.<データベース名>.<所有者名>.<オブジェクト名>

インスタンス名がserver01、データベース名がkeiridb、所有者名がuser1、オブジェクト名がuriageの場合は、次のように指定します。

```
server01.keiridb.user1.uriage
```

オブジェクトが、現在ログインしているインスタンスのカレントデータベース上にある場合には、インスタンス名とデータベース名を省略できます。また、ユーザー名と所有者名が同じであったり、dboが所有者のオブジェクトの場合には所有者名も省略できます。SQL Serverには、Oracleのシノニムに当たるものは存在しませんが、オブジェクトをdboが所有するかビューを使えば、同等の使い方が可能となります。

Oracleではインスタンスとデータベースは一対一になり、同一データベース上に複数のスキーマを作成することがあります。これは、オブジェクトをグループ化することによって管理を容易にしたり、テスト環境を複数作成したりするときに便利です。

SQL Serverでは、同一データベースのオブジェクトはデータベース所有者が所有することを推奨しているため、両者を比較する場合は図1-2のように考えるようにしてください。

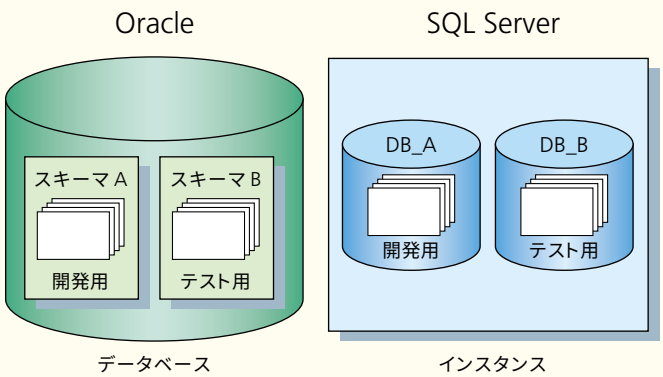

図1-2
スキーマとデータベース

データ型

SQL Serverには、Oracleより多数のデータ型を使用できます。以下に、SQL Serverで使うことのできるデータ型を示します。

分類	データ型	説明
int型	bigint	-2^{63} (-9,223,372,036,854,775,808) から $2^{63}-1$ (9,223,372,036,854,775,807) までの整数を格納するデータ型です。

（続き）

分類	データ型	説明
int型	int	-2^{31} (-2,147,483,648) から $2^{31}-1$ (2,147,483,647) までの整数を格納するデータ型です。
	smallint	-2^{15} (-32,768) から $2^{15}-1$ (32,767) までの整数データを格納するデータ型です。
	tinyint	0から255までの整数データを格納するデータ型です。
bit型	bit	1または0の整数データを格納するデータ型です。
decimal型とnumeric型	decimal	$-10^{38}+1$ から $10^{38}-1$ までの固定長の有効桁数と小数点部桁数の数値データを格納するデータ型です。
	numeric	decimal型と同じです。
money型とsmallmoney型	money	通貨単位の10,000分の1までの精度で、-2^{63} (-922,337,203,685,477.5808) から $2^{63}-1$ (+922,337,203,685,477.5807) までの金額データ値を格納するデータ型です。
	smallmoney	通貨単位の10,000分の1までの精度で、-214,748.3648から+214,748.3647までの金額データ値です。
概数型	float	-1.79E+308から1.79E+308までの浮動小数点数のデータを格納するデータ型です。
	real	-3.40E+38から3.40E+38までの浮動小数点数のデータを格納するデータ型です。
日付時刻型	datetime	300分の1秒、つまり3.33ミリ秒の精度で、1753年1月1日から9999年12月31日までの日付と時刻データを格納するデータ型です。
	smalldatetime	分単位の精度で、1900年1月1日から2079年6月6日までの日付と時刻データを格納するデータ型です。
文字列型	char	8,000文字以内の固定長のUnicode以外の文字データを格納するデータ型です。
	varchar	8,000文字以内の可変長のUnicode以外のデータを格納するデータ型です。
	text	$2^{31}-1$ (2,147,483,647) 文字以内の可変長のUnicode以外のデータを格納するデータ型です。

（続き）

分類	データ型	説明
Unicode文字型	nchar	4,000文字以内の固定長のUnicodeデータを格納するデータ型です。
	nvarchar	4,000文字以内の可変長のUnicodeデータを格納するデータ型です。sysnameはシステム提供のユーザー定義データ型です。これは、nvarchar(128)と同機能で、データベースオブジェクト名を参照するときに使用します。
	ntext	2^{30}-1（1,073,741,823）文字以内の可変長のUnicodeデータを格納するデータ型です。
バイナリ型	binary	8,000バイト以内の固定長のバイナリデータを格納するデータ型です。
	varbinary	8,000バイト以内の可変長のバイナリデータを格納するデータ型です。
	image	2^{31}-1（2,147,483,647）バイト以内の可変長のバイナリデータを格納するデータ型です。
その他のデータ型	cursor	カーソルへの参照を格納するデータ型です。
	sql_variant	このデータ型には、text型（テキスト型）、ntext型、timestamp型、sql_variant型を除き、SQL Serverでサポートしている各種データ型の値が格納されます。
	table	後続の処理に備えて結果セットを格納しておくための特別なデータ型です。
	timestamp	行が更新されるたびに更新される、データベース内で一意な番号を格納するデータ型です。
	uniqueidentifier	グローバル一意識別子（GUID）を格納するデータ型です。

変数

SQL Serverでユーザーが自由に扱える変数のことを、ローカル変数と呼びます。ローカル変数は、必ず変数名の前に@を付けます。変数を宣言するには、DECLAREステートメントを使います。変数に値を代入するには、SETステートメントを使います。次の例では、変数@xと@yをint型で定義し、それぞれに100と500を代入しています。

```
DECLARE @x int, @y int
SET @x = 100
SET @y = 500
```

SQL Serveでは、グローバル変数（先頭に@@が付く）も使うことができます。SQL Server 7.0からは、グローバル変数はシステム関数として扱われています（ただし、これは呼び名が変わっただけで機能は同じです）。ローカル変数と同じように使用できますが、値を参照することしかできません。代表的なものを以下に示します。

- **@@ERROR**
  直前のTransact-SQLステートメントのエラー番号を返します。OracleのPL/SQLでは、例外処理をEXCEPTIONキーワード以下に記述します。SQL Serverでは、@@errorの値を判断して例外処理を行います。
- **@@IDENTITY**
  直前のINSERT文で追加された行のID値を返します（詳細については、あとで説明するIDENTITYプロパティを参照）。
- **@@ROWCOUNT**
  直前のTransact-SQLステートメントで処理された行数を返します。
- **@@TRANCOUNT**
  現在の接続に対してアクティブなトランザクションの数を返します（詳細については、あとで説明するトランザクションを参照）。

SQLステートメントを記述するときの注意

OracleおよびSQL Serverには、さまざまなSQLステートメントの書き方があります。ここでは、よく使うSQLステートメントで特に注意が必要なものを紹介します。

日付の取り扱い

SQL Serverで日付に関連する操作を行う場合は、DATEADD、DATEDIFF、DATENAME、DATEPART、DAY、MONTH、YEARといった関数を使って行います。各関数の詳細については、SQL Server Books Onlineを参照してください。

また、現在時刻を取得するには、OracleではSYSDATE関数を使います。SQL ServerではGETDATE関数を使います。

2桁の年の扱い

Oracleでは2桁の年を4桁に変換する場合、RR書式またはYY書式によって50年単位の書式を設定します。

SQL Serverでは、1年単位で2桁年の解釈を決めます。設定はサーバーのオプションで行います。図1-3の設定はデフォルトで、2桁の年は1950年から2049年の間に存在すると解釈します。

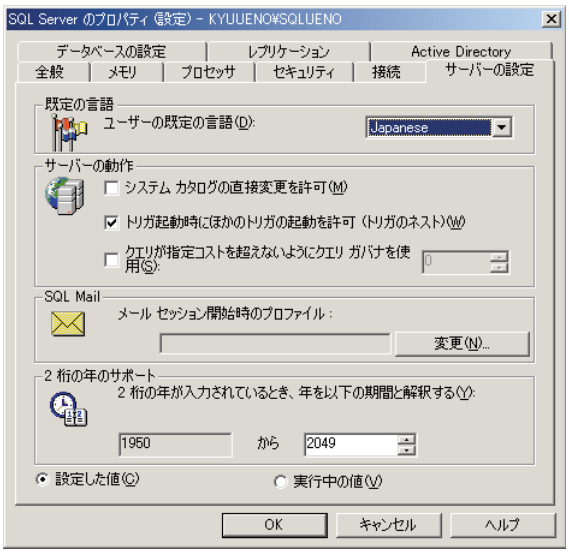

図1-3
2桁年の扱い
(SQL Server Enterprise Manager)

FROM句の省略

Oracleで SELECTステートメントを使用して結果を表示する場合、FROM句を省略できないため、DUALというダミーテーブルを使用します。

SQL Serverの場合はFROM句を省略できるため、今日の日付を出力したい場合は以下のように記述できます。

```
SELECT GETDATE()
```

PRINTステートメントの場合にも同等の記述が可能です。

変数への値の代入

SQL Serverでは、以下のように変数に値を入れることもできますが、これは旧バージョンでの記述方法です。現在では、SETステートメントを使った代入方法が推奨されています。

```
SELECT @x = 12345
```

文字列の連結

Oracleでは文字列の連結には || 記号を使いますが、SQL Serverの場合は + 演算子を使います。以下のコードは、employeeテーブルのfname列とlname列を連結したものを選択しています。

```
SELECT fname + lname FROM employee
```

テーブルの結合

Oracleでテーブルの内部結合を行う場合、テーブル名はFROM句の後ろに複数指定して、結合条件はWHERE句に記述します。

```
SELECT CategoryName, ProductName
  FROM Categories C, Products P
 WHERE C.CategoryID = P.CategoryID AND
       ProductName = 'Ikura'
```

SQL Serverでも旧バージョンとの互換性から同等の記述が可能ですが、結合条件と行の選択条件を明確に区別し、内部結合を明記して以下のように記述します。

```
SELECT CategoryName, ProductName
  FROM Categories C INNER JOIN Products P
    ON C.CategoryID = P.CategoryID
 WHERE ProductName = 'Ikura'
```

Oracleの外部結合の記述方法は、以下のとおりです。

```
SELECT ProductName, Quantity
  FROM Products P, Orders O
 WHERE P.ProductID (+) = O.ProductID
```

SQL Serverの場合は、Oracleの(+)の代わりに*=や=*を記述すれば同等の記述が可能ですが、SQL-92規格に準拠した外部結合構文の使用が推奨されています。以下のように記述します。

```
SELECT ProductName, Quantity
  FROM Products P LEFT OUTER JOIN Orders O
    ON P. ProductID = O.ProductID
```

SQLステートメントの暗号化

SQL Serverでは、ビュー、ストアドプロシージャ、トリガ、ユーザー定義関数を定義するSQLステートメントはシステムテーブルに保存されます。これらを暗号化して保存すれば、SQLステートメントを隠すことができます。

オブジェクトの作成

Oracleのオブジェクト作成には、CREATE OR REPLACEステートメントを使用できます。「OR REPLACE」は、オブジェクトが存在しなければ新規に作成し、存在していれば再作成することを意味しています。

SQL ServerではOR REPLACE句を指定することはできないため、CREATEステートメントを実行する前にオブジェクトの存在を確認する必要があります。以下のコードは、テーブルの存在を確認し、存在した場合には削除しています。

```
IF EXISTS (SELECT * FROM dbo.sysobjects
           WHERE id = object_id('テーブル名'))
DROP TABALE テーブル名
```

選択した行数の制限

Oracleでは、フェッチした各行に行番号（順序番号とも言います）が割り振られ、ROWNUMキーワードで表すことができます。ROWNUMをWHERE句に使用して選択した行の制限を行います。

SQL ServerにはOracleのROWNUMキーワードに当たるものはないので、以下のように行数を制御します。

- **SET ROWCOUNT *n***

  セッションごとに選択された行を制限できます。
- **TOPキーワード**

  SELECT文ごとに返された行を制限できます。たとえば、以下のTransact-SQLステートメントの場合、発注テーブルから発注個数が多い順に10行選択できます。

```
SELECT TOP 10 * FROM orders
ORDER BY quantity DESC
```

また、WITH TIES句を使用すると、10番目の行と同じ発注個数の行も選択できます。

```
SELECT TOP 10 WITH TIES * FROM orders
ORDER BY quantity DESC
```

さらに、行数ではなく全体の割合で選択することもできます。以下のコードは、全体の1%の行だけ選択します。

```
SELECT TOP 1 PERCENT * FROM orders
ORDER BY quantity DESC
```

ブロック

SQL Serverの流れ制御でブロックが必要になることがあります。ブロックは、BEGINキーワードとENDキーワードの間に記述されたTransact-SQLステートメントをブロックとして定義します。たとえば、以下のTransact-SQLステートメントにブロックを作成しなかった場合、SQL Serverは最初のPRINTステートメントだけしか実行しません。つまり、IF文などの条件式の中にSQLステートメントは1つしかないと解釈するのです。複数のステートメントを条件式に入れる場合は、必ずブロックを作成してください。

```
IF (SELECT AVG(price) FROM titles
    WHERE type = 'mod_cook') < $15
BEGIN
   PRINT 'The following titles are excellent mod_cook '
   PRINT 'books: '
   SELECT SUBSTRING(title, 1, 35) AS Title
   FROM titles
   WHERE type = 'mod_cook'
END
ELSE
   IF (SELECT AVG(price) FROM titles
       WHERE type = 'mod_cook') > $15
BEGIN
   PRINT 'The following titles are expensive mod_cook '
   PRINT 'books: '
   SELECT SUBSTRING(title, 1, 35) AS Title
   FROM titles
   WHERE type = 'mod_cook'
END
```

一時テーブル

SQL Serverでは、ユーザーの暗黙的な作業領域としてtempデータベースが使用されます。これはORDER BY句などを使用した並べ替えで使用されます。明示的に作業テーブルが必要な場合は、以下の方法を使用します。

- **ローカル一時テーブル**
  テーブル名の先頭に#記号を付けると、SQL Serverはtempデータベースにテーブルを割り当てます。このテーブルは、作成したセッションだけが使用可能です。明示的に削除するか、テーブルを作成したセッションが切断されるまで保持されます。
- **グローバル一時テーブル**
  テーブル名の先頭に##記号を付けると、SQL Serverはtempデータベースにテーブルを割り当てます。このテーブルは別のセッションでも使用可能です。明示的に削除するか、テーブルを参照しているセッションすべてが切断されるまで保持されます。
- **パーマネントテーブルの作成**
  データベースにテーブルを作成します。ただし、テーブルの作成権限が必要です。また、明示的に削除する必要があります（ただし、tempデータベースに作成した場合はシステム起動時に削除されます）。

一時テーブルがよく使われるのは、新規にテーブルを作成し、既存のテーブルからデータをコピーするときです。これには、SELECT INTOステートメントを使用します（Oracleの場合は、CREATE TABLE AS SELECTステート

メント）。作成するテーブルが一時的な作業で使用するテーブルの場合は、一時テーブルを指定すると便利です。しかし、作成するテーブルがパーマネントテーブルの場合、このステートメントの実行はトランザクションログに記録されないため、データベースの復旧モードが完全（フル）の場合は実行できません。

IDENTITY プロパティ

Oracle でテーブルの列に自動的に番号を生成するには、順序（SEQUENCE）を作成します。順序は、テーブルとは別のオブジェクトとして存在します。

SQL Server では、IDENTITY プロパティを使用します。IDENTITY プロパティはテーブルの列定義として存在します。以下の Transact-SQL ステートメントは、order テーブルの order_no を 10000 から始めて 5 ずつ増加させる番号を生成しています。

```
CREATE TABLE orders
(order_no    INT IDENTITY(10000,5) NOT NULL
 order_date DATETIME NOT NULL
 customer    VARCHER(50) NOT NULL)
```

制約

Oracle と SQL Server の制約に関する違いを、以下に示します。

項目	Oracle	SQL Server
PRIMARY KEY 制約	同等	同等
UNIQUE 制約	同等	同等
UNIQUE 列の NULL 値	複数行可能	1 行のみ
制約の無効と有効	DISABLE、ENABLE キーワード	NOCHECK、CHECK キーワード
PRIMARY KEY、UNIQUE 制約削除時の参照制約の連鎖削除	CASCADE オプション	事前に参照制約を削除
DEFAULT 制約	同等	同等
CHECK 制約	ほかの列の参照不可	同一テーブル内の列の参照が可能
FOREIGN KEY 制約	同等	同等
FOREIGN KEY による連鎖変更	CASCADE DELETE	CASCADE DELETE、CASCADE UPDATE

表：Oracle と SQL Server の制約