SQL Serverでどーんといってみよう!

必ず役立つ現場のノウハウ

百田 昌馬
HYAKUTA, Shoma
Supported by 松本 美穂
http://www.sqlquality.com/

最終回 SQL Server 2005で強化されたTransact SQL

はじめに

SQL Server 2005は、SQLCLR（.NET Framework統合）が取り沙汰されることが多いが、すべてのコード（ストアドプロシージャやトリガ、ユーザー定義関数）をSQLCLRで置き換えるべきというわけではない。単純なデータアクセスや単純な演算処理であればTransact SQLのほうがパフォーマンスが良いので、従来通りTransact SQLを利用したほうが良い。あくまでもSQLCLRのメリットは、開発生産性の高さと、複雑な処理を何度も呼び出すような場合のパフォーマンス向上である。

そこで、最終回となる今回は、SQL Server 2005で大幅に強化されたTransact SQLの新機能について説明する。例外処理（TRY…CATCH）のサポートやTOP句での変数、PIVOT、ROW_NUMBER、RANK、CTE（共通テーブル式）など、多くの便利な機能が追加されているのでぜひ活用してほしい。

レベル >>> Level
1 2 3 4 5

言語 >>> Language
- T-SQL

ツール >>> Tool
- SQL Server 2005

例外処理 ～TRY CATCH

まずは、例外処理機構（TRY…CATCHのサポート）からみていく。これは、Transact SQLの新機能の中で筆者が一番気に入っている機能である。以前のバージョンでは、ストアドプロシージャ内のエラー処理は、インラインエラー処理しかサポートされず、次のように@@ERRORシステム関数を使ってひとつひとつのステートメントごとに行なわなければならなかった。

```
BEGIN TRANSACTION
  INSERT INTO t VALUES(1, 999)
    IF @@ERROR <> 0
      BEGIN
        ROLLBACK TRANSACTION
        RETURN
      END
  INSERT INTO t VALUES(2, 888)
    IF @@ERROR <> 0
      BEGIN
        ROLLBACK TRANSACTION
        RETURN
      END
COMMIT TRANSACTION
```

しかし、SQL Server 2005からTRY…CATCHによる構造化エラー処理がサポートされたので、次のようにまとめてエラー処理を記述できるようになった。

```
BEGIN TRY
  BEGIN TRANSACTION
```

```
    INSERT INTO t VALUES(1, 999)
    INSERT INTO t VALUES(2, 888)
  COMMIT TRANSACTION
END TRY
BEGIN CATCH
  SELECT ERROR_NUMBER(), ERROR_MESSAGE()
        ,ERROR_SEVERITY(), ERROR_STATE()
  ROLLBACK TRANSACTION
END CATCH
```

これにより、BEGIN TRYからEND TRYまでのステートメントでエラーが発生した場合は、BEGIN CATCHへジャンプし、END CATCHまでのステートメントが実行されるようになる。

✷ERROR_MESSAGEでエラーメッセージも取得可能

@@ERRORは、エラー番号は取得できても肝心のエラーメッセージを取得することができなかったが、SQL Server 2005ではERROR_xxxで始まるいくつかの関数が追加され、エラーに関する周辺情報を取得できるようになった。主なものは次のとおり。

・ERROR_NUMBER：エラー番号
・ERROR_MESSAGE：エラーメッセージ
・ERROR_SEVERITY：重大度レベル
・ERROR_STATE：状態番号

TOP句の変数

以前のバージョンでは、次のようにTOP句で変数を記述することはできなかった。

```
CREATE PROCEDURE topN
  @n int
AS
  SELECT TOP @n * FROM t
```

したがって、TOP句の値を変数化したい場合は、次のように動的SQLを利用するか、SET ROWCOUNTステートメントを利用するしかなかった。

```
EXEC('SELECT TOP ' + @n + ' * FROM t')
```

または、

```
SET ROWCOUNT @n
SELECT * FROM t
SET ROWCOUNT 0
```

しかし、SQL Server 2005からTOP句での変数がサポートされたので、わざわざこういった記述をする必要はない。次のようにTOP句の後にカッコを付けて変数を与えるだけでよい（関数のように利用できる）。

```
CREATE PROCEDURE topN
  @n int
AS
  SELECT TOP(@n) * FROM t
```

✷TOP句は更新系でもOK

SQL Server 2005のTOP句は、更新系のステートメントでも利用できる。たとえば、DELETEステートメントの場合は、次のように記述できる。

```
DELETE TOP(5) FROM t
```

今までは、大量のデータ削除時のロック競合を回避する方法として、SET ROWCOUNTステートメントを使って行数を制限して少しずつデータを削除していくしかなかったが、このようなケースでTOP句を利用できるようになる。なお、SET ROWCOUNTステートメントは、今後のバージョンで削除される予定なので、今のうちからTOP句に切り替えておくことをお勧めする。

PIVOT ～クロス集計

以前のバージョンでは、クロス集計を行なうにはCASE関数（AccessでいうところのIIF、OracleでのDECODE）を使って図1のように記述しなければならなかった。しかし、SQL Server 2005からはPIVOT演算子がサポート