データベース更新の競合問題を考える
編集されたときに即座に書き戻す仕組み

大澤 文孝
OSAWA, Fumitaka

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ☑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2003
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  - ADO.NET
  - MSDE

Level

★ ★ ★ ☆ ☆

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの¥DOTNET¥FEATUREディレクトリに収録しています。

¥DBSAMPLE
本稿で使用しているサンプルデータベース

¥SAMPLESOURCE
本稿で取り上げたサンプルプログラムの全ソース

・README.TXT
サンプルソースを使用する際の注意事項

SQL Serverにサンプルデータベースを取り込む際は、DBSAMPLEディレクトリのデータを、MSDEを使用する場合はSAMPLESOURCEディレクトリの「SAMPLEDB.SQL」を使用してください。

はじめに

よく見かけるサンプルアプリケーションとして、「Windowsフォームにおけるデータベース開発入門」では、[読み込み]ボタンや、[書き込み]ボタンを用意し、前者が押されたときには、SqlDataAdapterのFillメソッドを呼び出してデータベースから読み込み、後者が押されたときには、SqlDataAdapterのUpdateメソッドを呼び出してデータベースに書き戻す処理について説明しました。

この方法は、サンプルとしてはわかりやすいのですが、実務で使うことを考えると、「[書き込み]ボタンを押さずにアプリケーションが終了されてしまい、編集後の結果がデータベースに反映されない」という危険性があります。

もちろん、フォームのClosingイベントでUpdateメソッドを呼び出す処理をして、フォームが閉じられるときに、強制的にデータベースに書き戻す方法もとれます。しかし、途中でアプリケーションが異常終了したときには対応できません。

たとえば、ユーザーが大量のレコードを登録したあと、アプリケーションが異常終了して、それがデータベースに書き戻されないなど、許されることではありません。

そこで本稿では、DataSet上のデータが変化したとき、それを即座にデータベースに書き戻す仕組みを説明します。

更新と読み込みの問題

Windowsフォームでデータバインディングを使ったときの編集は、DataSetを使った非接続型データベースアクセスとなります。

つまり、一度メモリに読み込み、編集したのちに、データベースに書き戻します。すなわちユーザーが編集しているのは、データベースの情報ではなく、SqlDataAdapterのFillメソッドを呼び出したときに得た、メモリ上の情報です。

マルチユーザー環境では、他のユーザーがデータベースを変更するため、

データベース上のデータと、ユーザーが編集しているDataSet内のデータが、合致しないことがあります。

そのため、SqlDataAdapterのFillメソッドとUpdateメソッドを使う場合には、いくつかの点に注意しなければなりません。

競合とは何か

データベースアプリケーションで、もっとも問題となるのは、「データベースに書き戻そうとしたときに、すでに誰かが該当レコードの値を変更したり、削除していた」という状況です。この問題が、「競合」です。

誰かが変更していた場合に、そのまま書き戻すと、その変更を上書きしてしまうことになります。また、削除していたものを書き戻すと、誰かが削除したはずのものが復活してしまいます。

そのため、競合が発生したときには、その状態をユーザーに告げ、再編集させる必要があります[注1]。

ADO.NETでは、SqlDataAdapterのUpdateメソッドを呼び出したときに、「更新対象となるレコードがデータベース上に存在するか否か」で、競合が発生したかどうかを判断しています。競合が発生すると、DBConcurrencyException例外が発生します[注2]。

Updateメソッドを呼び出したときには、SqlDataAdapterのInsertCommand、UpdateCommand、DeleteCommandの各プロパティに設定した、いずれかのSQLクエリーが実行されます。

リスト1：UpdateCommandプロパティに設定されるSQLクエリーの例

```
UPDATE PRODUCTS
SET  productname = @productname, price = @price, picture = @picture
WHERE
 (id = @Original_id) AND (price = @Original_price) AND
 (productname = @Original_productname);

SELECT id, productname, price, picture
FROM  PRODUCTS
WHERE (id = @id)
```

たとえば、「Windowsフォームにおけるデータベース開発入門」のようにしてSqlDataAdapterコントロールを構成したときのUpdateCommandプロパティに設定されるSQLクエリーは、リスト1のようになっています。

リスト1を見るとわかるように、WHERE句には、id列、price列、productname列が設定されています。そのため、データベース上で、これらの列が変更された場合には、WHERE句の条件に合致するレコードが見つからないので、競合とみなされます。

リスト1で注意したいのは、WHERE句にバイナリ型の列は含まれていないという点です。

これは、サーバーエクスプローラからテーブルをドラッグしてSqlDataAdapterコントロールを作った場合には、WHERE句にバイナリ型の列は含まれない仕様になっています。そのため、「Windowsフォームにおけるデータベース開発入門」で説明した、image型のpicture列は、WHERE句に登場しません。

すなわちユーザーがpicture列の画像を編集しても、それは競合とはみなされず、更新時には、かまわず上書きされます。

この状態を避けたいなら、timestamp列など、更新のたびに異なる値を格納する列を用意し、それをWHERE句に含めるようにします。

ちなみにリスト1では、UPDATE文の後ろにSELECT文が付いていますが、これは、データベース上のレコードを再読み込みするためのものです。その意味は、すぐあとで説明します。

サーバー側で設定される値の取得

IDENTITY列のように、サーバー側で値が設定される場合には、データベースに書き戻したあと、サーバー上で割り当てられた値をDataSetに書き戻す必要があります。

たとえば、「Windowsフォームにおけるデータベース開発入門」で説明したPRODUCTSテーブルでは、id列がIDENTITY列であるため、この値はサーバー上にレコードを挿入したときに確定します。

DataSetにIDENTITY列が含まれている場合、次のように動作します。

注1）もちろん再編集させるのではなく、「強制的に上書きするかどうか」を尋ねて、上書きして書き戻すという方法をとってもかまいません。

注2）ContinueUpdateOnErrorプロパティをTrueに設定すると、例外を発生させることなく処理を継続させることもできます。ただし、競合が発生したレコードの更新が飛ばされるだけであり、いかなる場合でも、競合が発生したレコードがデータベースに書き戻されることはありません。