SQL Server 2008 - The Power of Merge

Introduction

In Part IV of my *Sales Order Workshop*, I had presented a stored procedure which saves a sales order information into *Order Header* and *Order Detail* tables. If you have ever worked on an order processing application, you would realize that saving a modified sales order is little tricky. There may be new rows added to the sales order. There may be rows which are updated and there may be rows that should be deleted. If you have the freedom to delete all the rows from order details table and re-insert everything, you are lucky. But many of the times you cannot simply delete the order details table because there may be additional information like Quantity-picked etc, which is updated from other parts of the application. In those scenarios, you need to perform a *DELETE-UPDATE-INSERT* operation to save the information correctly. The following is the *pseudo code* that I had used in my stored procedure.

```
 1 /*
 2     Pseudo code used for saving sales order information with SQL Server 2005
 3
 4     -- save order header information
 5     If OrderNumber found in OrderHeader
 6         Update the information
 7     Else
 8         Insert the information
 9     end
10
11     -- save order detail information
12     Delete from Order detail table all items not in the order info
13     Update Order detail for all items present in the order info
14     Insert into order details all new items in the order info
15 */
```

SQL Server 2008 introduces ***MERGE***, a new keyword which performs *INSERT*, *UPDATE* and *DELETE* operations at one go. With SQL Server 2008, you can perform the above operation as simple as the following pseudo code.

```
 1 /*
 2     Pseudo code for saving the same order with the MERGE statement of SQL
Server 2008
 3
 4     -- save order header information
 5     MERGE order info to Order Header table
 6
 7     -- save order detail information
 8     MERGE order info to order detail table
 9 */
```

No, I did not miss anything. You can write the code in just 2 lines. The rest of this article presents a *SQL Server 2008* stored procedure which demonstrates this.

The Data

Here is the structure of the order data that we have. This XML has the order header and detail information. Our stored procedure needs to store the information in both tables. Some rows need to be updated, some inserted and some deleted.

```
<OrderInfo>
  <OrderHeader OrderNumber="20070101" CustomerNumber="J0001"
OrderDate="2007-07-08" />
  <ItemInfo>
    <Item ItemNumber="A001" Qty="10" Rate="100" />
    <Item ItemNumber="A002" Qty="11" Rate="200" />
    <Item ItemNumber="A003" Qty="12" Rate="300" />
    <Item ItemNumber="A004" Qty="13" Rate="400" />
    <Item ItemNumber="A005" Qty="14" Rate="500" />
  </ItemInfo>
</OrderInfo>
```

Tables

We need two tables to store the order information. Here is the script to create the tables.

```
CREATE TABLE [dbo].[OrderHeader](
    [OrderNumber] [varchar](20) NULL,
    [CustomerNumber] [varchar](20) NULL,
    [OrderDate] [datetime] NULL
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[OrderDetails](
    [OrderNumber] [varchar](20) NULL,
    [ItemNumber] [varchar](20) NULL,
    [Qty] [int] NULL,
    [Rate] [money] NULL
) ON [PRIMARY]

GO
```

Enter the Dragon

Let us see the Stored Procedure which uses the MERGE keyword.

```
CREATE PROCEDURE [dbo].[MergeSalesOrder]
(
    @OrderInfo XML
)
AS

SET NOCOUNT ON

/*
I am not using a TRY..CATCH or BEGIN TRAN to simplify the code.
*/

/*
Code to save order header. I am creating a CTE over the XML data to simplify
the code.
*/

;WITH OrderInfo AS (
    SELECT
        x.h.value('@OrderNumber', 'VARCHAR(20)') AS OrderNumber,
        x.h.value('@CustomerNumber', 'VARCHAR(20)') AS CustomerNumber,
        x.h.value('@OrderDate', 'VARCHAR(20)') AS OrderDate
    FROM @OrderInfo.nodes('/OrderInfo/OrderHeader') AS x(h)
)
MERGE OrderHeader AS h
USING OrderInfo AS o
ON (h.OrderNumber = o.OrderNumber)
WHEN MATCHED THEN
    UPDATE SET h.CustomerNumber = o.CustomerNumber, h.OrderDate = o.OrderDate
WHEN NOT MATCHED THEN
    INSERT (OrderNumber, CustomerNumber, OrderDate)
```

```
    VALUES (o.OrderNumber, o.CustomerNumber, o.OrderDate)
;

/*
Save Order Detail Information
*/

;WITH OrderInfo AS (
    SELECT
        x.h.value('(../../OrderHeader/@OrderNumber)[1]', 'VARCHAR(20)') AS
OrderNumber,
        x.h.value('@ItemNumber', 'VARCHAR(20)') AS ItemNumber,
        x.h.value('@Qty', 'INT') AS Qty,
        x.h.value('@Rate', 'MONEY') AS Rate
    FROM @OrderInfo.nodes('/OrderInfo/ItemInfo/Item') AS x(h)
)
MERGE OrderDetails AS d
USING OrderInfo AS o
ON (d.OrderNumber = o.OrderNumber AND d.ItemNumber = o.ItemNumber)
WHEN MATCHED THEN
    UPDATE SET
        d.ItemNumber = o.ItemNumber,
        d.Qty = o.Qty,
        d.Rate = o.Rate
WHEN NOT MATCHED THEN
    INSERT (OrderNumber, ItemNumber, Qty, Rate)
    VALUES (o.OrderNumber, o.ItemNumber, o.Qty, o.Rate)
WHEN SOURCE NOT MATCHED THEN
    DELETE
;

/*
Points to note:
1. MERGE statement should be terminated with a semi colon
2. The JOIN (USING...ON) should not result in duplicate records.
3. When the records in the SOURCE and TARGET matches, MATCHED becomes true
4. When the record is not in the TARGET, NOT MATCHED becomes true
5. When the record is not in the SOURCE, SOURCE NOT MATCHED becomes true.
*/
```

Execute the code

It is time to execute the code. Use the following code to execute the stored procedure.

```
EXECUTE [MergeSalesOrder] '
<OrderInfo>
  <OrderHeader OrderNumber="20070101" CustomerNumber="J0001"
OrderDate="2007-07-08" />
  <ItemInfo>
    <Item ItemNumber="A001" Qty="10" Rate="100" />
    <Item ItemNumber="A002" Qty="11" Rate="200" />
    <Item ItemNumber="A003" Qty="12" Rate="300" />
    <Item ItemNumber="A004" Qty="13" Rate="400" />
    <Item ItemNumber="A005" Qty="14" Rate="500" />
  </ItemInfo>
</OrderInfo>
'

/*
Let us check the results
*/
SELECT * FROM OrderHeader
SELECT * FROM OrderDetails

/*
OrderNumber         CustomerNumber       OrderDate
-------------------- -------------------- -----------------------
20070101             J0001                2007-07-08 00:00:00.000

(1 row(s) affected)
```

```
 26
 27 OrderNumber         ItemNumber           Qty         Rate
 28 -------------------- -------------------- ----------- ---------------------
 29 20070101             A001                 10          100.00
 30 20070101             A002                 11          200.00
 31 20070101             A003                 12          300.00
 32 20070101             A004                 13          400.00
 33 20070101             A005                 14          500.00
 34
 35 (5 row(s) affected)
 36 */
```

The above code shows that the order is saved correctly. Now lets us modify the order info. Let us delete a row, add a new row and modify an existing row. Here is the code. Note that Item *A005* is deleted. Item *A006* is added and item *A001* is modified. Let us execute the code and see the results.

```
  1 EXECUTE [MergeSalesOrder] '
  2 <OrderInfo>
  3   <OrderHeader OrderNumber="20070101" CustomerNumber="J0001"
OrderDate="2007-07-08" />
  4   <ItemInfo>
  5     <Item ItemNumber="A001" Qty="15" Rate="150" />
  6     <Item ItemNumber="A002" Qty="11" Rate="200" />
  7     <Item ItemNumber="A003" Qty="12" Rate="300" />
  8     <Item ItemNumber="A004" Qty="13" Rate="400" />
  9     <Item ItemNumber="A006" Qty="16" Rate="600" />
 10   </ItemInfo>
 11 </OrderInfo>
 12 '
 13
 14 SELECT * FROM OrderHeader
 15 SELECT * FROM OrderDetails
 16
 17 /*
 18 OUTPUT:
 19
 20 OrderNumber         CustomerNumber       OrderDate
 21 -------------------- -------------------- -----------------------
 22 20070101             J0001                2007-07-08 00:00:00.000
 23
 24 (1 row(s) affected)
 25
 26 OrderNumber         ItemNumber           Qty         Rate
 27 -------------------- -------------------- ----------- ---------------------
 28 20070101             A001                 15          150.00
 29 20070101             A002                 11          200.00
 30 20070101             A003                 12          300.00
 31 20070101             A004                 13          400.00
 32 20070101             A006                 16          600.00
 33
 34 (5 row(s) affected)
 35
 36 */
```

Conclusions

I found the ***MERGE***
keyword very powerful and friendly. It reduces the complexity of the code and provides a simple interface to perform a complex operation. I like it and I suppose many of you around there would like it too.