



<http://www.progressivebusinesstechnologytraining.com/>

Batch Updates in ADO.NET 2.0 for Improved Performance

by [David Hayden](#) ([Florida .NET Developer](#))

<http://davidhayden.com/blog/dave/archive/2006/01/05/2665.aspx>

When you updated a database using the DataAdapter in .NET 1.1 each command was sent to the database one at a time. This caused a lot of roundtrips to the database.

[ADO.NET 2.0](#) has introduced the concept of **Batch Updates**, which allows you to designate the number of commands sent to the database at a given time. If used correctly, this can increase the performance of your data access layer by reducing the number of roundtrips to the database.

DataAdapter.UpdateBatchSize Property

The DataAdapter has an **UpdateBatchSize** Property that allows you to set the number of commands that will be sent to the database with each request.

- UpdateBatchSize = 1, disables batch updates
- UpdateBatchSize = X where X > 1, sends x statements to the database at a time
- UpdateBatchSize = 0, sends the maximum number of statements at a time allowed by the server

Command.UpdatedRowSource Property

When using batch mode, the **UpdatedRowSource** property of the command can only be set to either **UpdatedRowSource.None** or **UpdatedRowSource.OutputParameters**

Batch Updates Tutorial Using Northwind

You can test out **batch updates** on the Northwind Database by simulating an update to the Categories Table.

First, get the data from the Categories Table. The code below gets the information and places it in an untyped DataSet:

```
SqlConnection connection =
    new SqlConnection("...");

SqlDataAdapter adapter =
    new SqlDataAdapter("SELECT * FROM Categories",
        connection);

DataSet ds = new DataSet();

adapter.Fill(ds);
```

Simulate modification of the CategoryName of each category so there is something to update:

```
foreach (DataRow dr in ds.Tables[0].Rows)
{
    string categoryName = dr["CategoryName"].ToString();
    dr["CategoryName"] = categoryName;
}
```

Construct an update command for the SqlDataAdapter to update the data and assign it to the adapter's **UpdateCommand** property:

```
SqlCommand command = new SqlCommand();

command.CommandText = "Update Categories
    Set CategoryName = @CategoryName WHERE
    CategoryID = @CategoryID";

command.Parameters.Add(new SqlParameter
    ("@CategoryID", SqlDbType.Int)).SourceColumn
```

```

        = "CategoryID";

command.Parameters.Add(new SqlParameter
    ("@CategoryName", SqlDbType.NVarChar, 15))
    .SourceColumn = "CategoryName";

adapter.UpdateCommand = command;

```

Set the **UpdatedBatchSize** and **UpdatedRowSource** equal to the proper values. In the case of the Categories Table, there are only 8 records in it and we have changed them all. I will set the UpdatedBatchSize to 2 for the sake of testing.

```

adapter.UpdateBatchSize = 2;
command.UpdatedRowSource = UpdateRowSource.None;

```

Execute the update process:

```

adapter.Update(ds);

```

Hooking into the DataRowUpdating and DataRowUpdated events of the DataAdapter

If I hook into the **DataRowUpdating** and **DataRowUpdated** events of the DataAdapter as so:

```

adapter.RowUpdating +=
    new SqlRowUpdatingEventHandler(adapter_RowUpdating);
adapter.RowUpdated +=
    new SqlRowUpdatedEventHandler(adapter_RowUpdated);

private void adapter_RowUpdated(object sender,
    SqlRowUpdatedEventArgs e)
{
    _countUpdated++;
}

void adapter_RowUpdating(object sender,
    SqlRowUpdatingEventArgs e)

```

```
{  
    _countUpdating++;  
}
```

Without Batch Updates, both events will fire 8 times, one for each row being updated.

However, with Batch Updates, RowUpdated will only be called once for each batch update (8 rows / 2 updates per batch = 4 times). RowUpdating will be called the usual 8 times.

Conclusion

Batch updates can improve the performance of your data access layer by reducing the number of roundtrips to the database.

Source: [David Hayden](#) ([Florida .NET Developer](#))

Related ADO.NET Articles

- [Typed DataSet and Sorting Filtering and Searching a DataTable in ADO.NET - Custom Expressions - ShoppingCart DataSet](#)
- [DataTable - Load from IDataReader - CreateDataReader - ReadXml WriteXml - Free ADO.NET Tutorials](#)
- [.NET 2.0 Provider Model - Polymorphism - Factory Method - ADO.NET 2.0 Data Providers](#)
- [ObjectDataSource Control and Concurrency in Multiuser ASP.NET Web Applications - OverwriteChanges - CompareAllValues](#)
- [Paging Records Using SQL Server 2005 Database - ROW_NUMBER Function](#)
- [SQL Server Transaction Savepoints - Rollback Part of Transaction - SqlTransaction.Save - SqlTransaction.Rollback](#)
- [Sql Server Transactions - ADO.NET 2.0 - Commit and Rollback - Using Statement - IDisposable](#)
- [SQL Server - Optimistic Concurrency Database Updating - Pessimistic Concurrency - High Performance ASP.NET Websites](#)