

Richard Astbury's Blog

[GitHub](#) [Twitter](#) [SoundCloud](#) [Projects](#) [About](#) [Contact](#)

Bulk inserting data into SQL Server

*Nov 28, 2017**4 minute read*

Let's say we have a simple table in SQL Server


```
CREATE TABLE [dbo].[test] (  
    [Value] [varchar](36) NOT NULL  
)
```

...in which we want to insert thousands of values (Guids) from a .NET application.

One by one

A naive approach is to insert one at a time, like this:

```
public void OneByOne()  
{  
    const string sql = "INSERT INTO [Test] ([Value]) Values  
    for (var i = 0; i < count; i++)  
    {  
        connection.Execute(sql, new { Value = Guid.NewGuid()  
    }  
}
```



(note this code uses Dapper)

Inserting 10,000 records on a local SQL Express database takes 54,533ms, which is **183 records per second**. Slow :-(

1000 at a time

SQL Server allows you to insert multiple records in a single insert statement, in fact we can insert up to 1,000 at a time.

```
public void BatchOf1000()
{
    foreach (var batch in Enumerable.Range(0, count).Chunk(1)
    {
        if (batch.Length == 0) continue;
        var sql = "INSERT INTO [Test] ([Value]) VALUES \r\n"
        connection.Execute(sql);
    }
}
```

Inserting 1,000,000 records on a local SQL Express database takes 22,256ms, which is **44,931 records per second**. Must faster.

Bulk Copy

Can we go any faster? Of course we can.

SQL Server (and SQL Database in Azure) supports [bulk insert](#), you may have used bcp in the past.

The `SqlBulkCopy` class provides easy access to this from .NET.

```
public void BulkCopy()
{
    var table = new DataTable();
    table.Columns.Add("Value", typeof(string));

    for (var i = 0; i < count; i++)
    {
        table.Rows.Add(Guid.NewGuid().ToString());
    }

    using (var bulk = new SqlBulkCopy(this.connection))
    {
        bulk.DestinationTableName = "test";
        bulk.WriteToServer(table);
    }
}
```

Inserting 1,000,000 records on a local SQL Express database takes 9,315ms, which is **107,353 records per second**. Even faster.

Notes on SqlBulkCopy

The above code samples shows that in C# you must first create a `DataTable`, and then tell it the schema of the destination table. You then add values according to their position in the table.

This code is a little fragile to changes in the table structure, so another approach is to load the table structure from the database, and then insert the values by name rather than position:

```
public void BulkCopy()
{
    var table = new DataTable();

    // read the table structure from the database
    using (var adapter = new SqlDataAdapter($"SELECT TOP 0 *
    {
        adapter.Fill(table);
    });

    for (var i = 0; i < count; i++)
    {
        var row = table.NewRow();
        row["Value"] = Guid.NewGuid().ToString();
        table.Rows.Add(row);
    }

    using (var bulk = new SqlBulkCopy(this.connection))
    {
        bulk.DestinationTableName = "test";
        bulk.WriteToServer(table);
    }
}
```

I've noticed a small overhead in using `SqlBulkCopy`, so I wouldn't use it for insert a small number of records (i.e. less than 100), but your mileage may vary.

Happy inserting!

1 Comment **richorama.github.io****Login** ▾ **Recommend**  **Tweet**  **Share****Sort by Best** ▾

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS **Lynn Nicholson** • 19 days ago

Thanks Richard for posting this. Tonight was my first time playing with SqlBulkCopy where my first approach was to create the DataTable and add the column and column types explicitly. However, trying to insert 7,700 records into an 8-column table always threw a Format/Type Conversion error. After implementing your example in my code, everything ran beautifully. And it is fast!

^ | ▾ • Reply • Share ›

</> available on [Github](#).