

## SQL Server Utilities and the Command Line

For most people in this day and age being forced to use the command line or resort to additional utilities can seem rather strange. I guess being spoilt by the depth and breadth of the Enterprise Manager in SQL Server doesn't help.

This month I will review some of the utilities and command line tools that can get you out of a hole every now and again. I wouldn't suggest that you would be likely to use these tools all day every day, but when you do need them then they are very useful. Think of them as the RAC or AA of computing.

These utilities are available in the Back Office 4.5 Resource Kit Toolbox or come as part of the standard SQL Server product set. In some cases the utility may have been superseded by features in SQL Server 2000, but they are useful for those still using SQL Server 7.0. Probably the best example of this is Log Shipping, which made its first appearance in the Back Office Resource Kit and is now a feature of SQL Server 2000.

### **DataSizer Spreadsheet**

This is an extremely useful tool to use when designing a database and needing some estimate as to the size of the production tables. In my organisation we are increasingly being asked to build larger and larger databases, often containing many millions of rows, and customers need to know the disk storage requirements.

The spreadsheet consists of two worksheets, one for clustered tables and the other for heap tables – those tables that do not have a clustered index. These worksheets will give an estimated size for a table plus its associated clustered or non-clustered indexes.

The formulas in the spreadsheet are based on internal algorithms in SQL Server, so they do give a good estimate for the table size. That said be aware of some limitations, specifically the spreadsheet does not allow space for:

- Allocation structures. These are the internal mapping devices used by SQL Server to track space used in a table or how index pages are being used.
- The effects of random data placement. SQL Server uses Page Free Space pages to allocate any available free space and recover deleted space. This is too complex to cater for in a simple sizing spreadsheet
- Data loading using BCP (Bulk Copy Process) or inserts. The chosen data load process can affect table space as BCP and inserts do behave differently.
- The order in which rows are inserted. As a clustered index will organise the data in clustered index order submitting the data “out of order” may affect the underlying storage efficiency

Good practice will have you adding an additional 5% size margin if you are building a system with many smaller tables. Systems with few larger tables will not need this additional margin. In addition the spreadsheet does not handle text columns. Text values are loaded onto text pages using the same algorithm as data rows, so it is possible to use the heap worksheet for this calculation if you need to.

### **SQL Converter**

Many developers need to take code from SQL Server and put it into a Visual Basic application. The difference between VB code and T-SQL means that a developer has to spend time placing irritating quotes and line continuation characters in the script, which can be time consuming.

For example the following T-SQL statement will need to be reformatted:

```
UPDATE authors SET au_lname = 'England'
WHERE au_id = '999-888-7777'
IF @@ROWCOUNT = 0
    print 'Warning: No rows were updated'
```

Equivalent Visual Basic Script:

```
"UPDATE authors SET au_lname = 'England'
WHERE " _
& "au_id = '999-888-7777'
IF @@ROWCOUNT = " _
& "0
    print 'Warning: No rows were updated'
"
```

This would be enough to drive most developers bananas. The SQL Converter utility will do this for you. By typing or pasting the T-SQL into the plain SQL query screen and hitting the “convert” button VB code is created. As you would expect, it is bi-directional so VB code can be converted into T-SQL. A great tool.



**Fig 1. SQL Converter**

## **OSQL**

This utility is used on occasions when you need to run T-SQL statements from the command line. It goes without saying that it needs a knowledge of T-SQL. This utility ships with SQL Server 2000 and is a replacement for the ISQL utility, which uses DB-Library for data access (OSQL uses ODBC). This supports a more general move that Microsoft is making away from DB-Library to the more generic database access APIs. In fact DB-Library remains frozen at the SQL Server 6.5 release, so any later SQL Server functionality is not supported anyway.

I don't intend to bore you with each and every OSQL command or option; instead I'll run through some high level features of the utility.

Once started the OSQL utility will check your credentials if they are supplied, or if you are just running OSQL with no environment variables then it will use the workstation settings. In addition to T-SQL statements the following commands can be used. Be aware that OSQL is case sensitive.

### **Command**

GO	Executes all statements entered after the last GO.
RESET	Clears any statements
ED	Calls the editor. It's possible to set the default editor, for example you may wish to use notepad. In that case use the command SET EDITOR=notepad
!! (command)	Executes an OS command.
QUIT or EXIT( )	Exits from OSQL.

CTRL+C

Ends a query without exiting from OSQL.

### **SQLServr Utility**

I suppose in fairness this is rather more than a utility. In fact this is the application that starts and stops an instance of SQL Server. It can be used from the command line to help debug problems with a SQL Server installation.

Here are some of the more interesting or frequently used arguments with SQLServr:

- v Displays the server version number
- m Starts SQL Server in single user mode, often used to repair system databases
- f Starts SQL Server in minimum configuration mode
- T Trace flag set to start SQL Server with non-standard behaviour. Note that the argument is an upper case T. Lower case t will force an internal trace flag used by Microsoft for support purposes only.

### **BCP**

No discussion of command line utilities and SQL Server can afford to miss BCP or Bulk Copy Program. This is used to copy data from one file to another, normally to import data into SQL Server. BCP really shows the legacy of SQL Server as it was a Sybase utility and until recently used DB-Library bulk copy as its connectivity API. In fact prior to the graphical tools created by Microsoft it was one of the few ways of getting data into and out of SQL Server.

BCP uses the SQL Server query processor to optimise the import of data to SQL Server, planning the best way of placing the data. When using BCP the DBA has the option of having a logged or non-logged operation. Both can be rolled back on failure, but the logged operation can be rolled forward if needed. As expected logged operations will take longer to execute.

BCP is probably the most complex command line utility available to the SQL Server DBA. For that reason we will try and schedule an article on using BCP to cover it in more depth in the future. The other utilities discussed are more straightforward, and hopefully you can have a chance to experiment with them in the future.