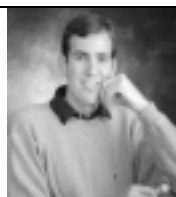


Fundamentals of DTS

Brian Knight, Alltel

Introduction

Data Transformation Services (DTS) is a built-in component of SQL Server that provides programmers and database administrators a way to move data from any source to any destination. In the past, you would have to use third party expensive products such as Data Junction to perform a similar task. Data is transformed in a workflow system where one action (known as steps in DTS) cannot occur until another completes, succeeds or fails. Microsoft has added an enormous amount of features into SQL Server 2000 (SS200) for DTS. This session provides an overview of how to use DTS and the new features that have been added in SQL Server 2000 for DTS. Afterwards, you will be able to use DTS to convert data from and to almost any data source.



Brian Knight has been working with SQL Server for about 5 years, during which he spent most of his time integrating SQL Server with web applications such as Cold Fusion and ASP. For the few past years he has been working with Alltel in Jacksonville, Florida, performance tuning enterprise servers and aiding in the creation of financial applications for the Web. He has written several columns with SQL Server Magazine, co-runs the SQL Server area on SWYNK.COM and runs SQLServerCentral.com.

DTS Terminology

Before we move into the detailed discussion of DTS, we must explain some of the terms that are used when running DTS. The primary component to DTS is a package. It is what you save or open when programming in DTS. At the next level, packages contain tasks, steps, connections and global variables. Tasks are an action that you are going to perform, such as running an ActiveX script or transforming data. Steps tell DTS which order to execute the tasks in. Connections contain information on how to connect to any OLE-DB compliant database or text file. Finally, global variables provide a way of passing variables to tasks and making your packages dynamic.

Weighing Coolness Versus Performance

One of the most common questions I receive is from developers asking where they should save their packages. To complicate matters slightly, there are now four places where you can save your packages in SQL Server 2000: in Visual Basic (VB) format, in COM-structured files, locally on the SQL Server, or in the SQL Server Meta Data Repository. Where you'd like to save your packages depends on how fast you'd like it to perform or the amount of features.

There is no better way to learn the DTS object model than to create a package as you would normally and then save it as a VB formatted file. Before you do this, however, make sure you also save it as a COM-structured file (".DTS") so you have a backup of your file. Once saved into a .BAS file, you can no longer view it in Designer, and have to edit it through Visual Basic or a text editor. This method of saving your package is perfect for taking snippets of code and placing them into a larger VB program or learning the DTS object model. Files with a ".DTS" extension load faster than other DTS package types. For large packages your execution time can be lowered significantly because the package loads faster. They also allow you to perform version control through Visual Source Safe, which is better than the crude version control system built into SQL Server Enterprise Manager. Visual Source Safe offers a way to check in and out DTS packages like any of your other source code. As you save packages, SQL Server will still version control the COM-structured files and you may need to purge them occasionally. DTS doesn't provide a means to check in and out packages without a third part product like Visual Source Safe. By default, the last user to save a package wins—the other user's changes are saved as a past version. Saving them as ".DTS" files also doesn't take up any space in SQL Server.

Saving packages locally and into Meta Data Services uses the MSDB database. Meta Data Services packages are far by the slowest of all the package types. There is also a greater chance of packages stored in this fashion being corrupted in SQL Server 7.0 before Service Pack 1 (which saw the problem being resolved). In my testing in SQL Server 2000, the corruption problem has not occurred. Meta Data Service is cool however because it self-documents your package, producing a searchable list of connections and tasks for you.

Copy Database Wizard (CDW)

CDW gives you the ability to copy or move database files and their accompanying logs to a new server. Other SQL Server items that will be transferred are objects like triggers, indexes, and stored procedures. It also transfers logins, error messages, and the job history associated with the database.

CDW is perfect for migrating data from development to production. It's also useful for migrating databases that are on an Alpha-machine to Intel-based machines. It can be executed from any machine that can connect to the source server. Because the source is always going to be a SQL Server 2000 machine, changes in sort order aren't a concern (SS2000 supports collation at the database and column level).

CDW uses many of the Operating System's APIs to perform the copy. It copies the files to a temporary share automatically, and then attaches itself to the destination server. Because it uses OS functionality, your account that is running the CDW must be a member of the sysadmin role on both the source and destination server.

There are two key restrictions to remember. There cannot be a database on the destination server with the same name as the database you're transferring. Also, you must be a member of the sysadmin role to perform this action. Also system databases cannot be transferred through CDW. CDW uses DTS on the backend, taking advantage of a number of new DTS tasks:

- Transfer Databases
- Transfer Logins
- Transfer Jobs
- Transfer Master Stored Procedures
- Transfer Error Messages

Import/Export Wizard

Another great tool for moving data is the Import/Export Wizard. This tool allows you to copy the data from any OLE DB provider to any other OLE DB provider. You can execute the transfer immediately or after (scheduled for later execution through SQL Agent). Processes that are performed often can be saved as a DTS package.

A few items that were nice in the beta have been removed from the final release of SS2000. You cannot create the Primary Key and Foreign Key information easily. You must either transfer into a table that already has those constraints or create them after the transfer.

Tool Enhancements

The DTS tools have been improved substantially to make a DTS programmer's life easier. In the DTS Designer, you can execute individual steps. Previously in SQL Server 7.0, execution could only be done at the package level. To execute an individual step, you'd have to disable every other step in the package. DTS can now cache packages automatically, so load time for a large package is minimized after it's cached – this option is on Windows 2000 machines only. You can now enable Visual Studio integration with DTS to debug your scripts. These options are not enabled by default however. To enable these options, you must click on Data Transformation Services with your right mouse button in Enterprise Manager and select properties.

There is also a disconnected edit mode for those who may not be connected to the network and don't want DTS to validate connections. This is perfect for DTS programmers that are working on the plane without connectivity to the main server. It allows you to update every DTS property from a single screen that resembles the Dynamic Properties task. The problem with this is that no validation is provided so it is easy to make a mistake and not notice it until runtime.

Microsoft provides you with an added tool in SQL Server 2000 called `DTSRunUI.EXE` that simplifies scheduling and executing a package. This tool can write the command line execution strings. It can also encrypt your commands and pass global variables into a DTS package. The tool is located typically in `C:\Program Files\Microsoft SQL Server\80\Tools\Binn`.

Lastly, extra logging abilities have been added to this release. You can log very granular details at the step level in system tables. A GUI is provided to view these logs by clicking your right

mouse button on a package. You must setup logging in the Package Properties under the logging tab. Logging will show you when a step was executed and when it completed.

Logging system tables in MSDB database:

- Sysdtspackagelog
- Stsdtsteplog
- Sysdtstasklog

The Multiphase Data Pump

The largest update to the old tasks is the multiphase data pump, which allows you to separate your transformation into several steps or to customizable them. Previously in SQL Server 7.0, there was one phase, the row transform phase. With the multiphase version, you can add extra error handling to trap constraint violations and send those records into an audit table. It also allows you to have row-level restartability. If a transformation fails after transforming a million records, you can add functions to trap the errors and continue with the good records. The multiphase data pump in SQL Server 2000 now consists of six phases:

- Pre Source – Functions placed in this phase are executed before the first record is fetched from the source. You can use functions placed here to write Meta data out to a file, reserve memory for later use, or initialize connections.
- Row Transform – This was the only phase in version 7.0. It's now the default phase in SQL Server 2000. This is the phase that actually transforms the columns.
- Post Row Transform – For each row that is transformed, the Post Row Transform phase is executed. This phase is divided into three sub phases: Transform Failure, Insert Success, and Insert Failure. Each row that's transformed can only have one outcome.
- On Batch Complete – By default, there is only one batch per data pump. You can change the batch size in the options tab under the Transform Data task. This phase is perfect for reporting the status of a large data load. Larger packages perform better when they have smaller batches. A general rule of thumb is a batch of about 10,000.
- Post Source Data – Unlike the On Pump Complete phase, the Post Source Data phase *can* access data. Ideally, you use this phase to write footer rows to the data you created in the Pre Source phase.
- On Pump Complete – After all rows have been transformed, the On Pump Complete functions will execute. This phase cannot access the data but it is ideal for freeing up memory that was taken during previous phases.

The multiphase data pump is, by default, not exposed to the Designer. To enable this, you must click your right mouse button on Data Transformation Services in Enterprise Manager and go to properties. Once in the Designer, phases are set up under the transformation tab for either Data Driven Query or Transform Data Tasks.

After creating the mapping as normal, you can create the logic in a simple VBScript as shown below. This script will present a message box as the package starts, completes, and processes each batch. For the `BatchCompleteMain` function to work, however, you will have to set the insert batch size in the options tab. By default, DTS will insert all records in a single batch, slowing the load as the batch grows. A good batch size for large loads is 10,000.

```

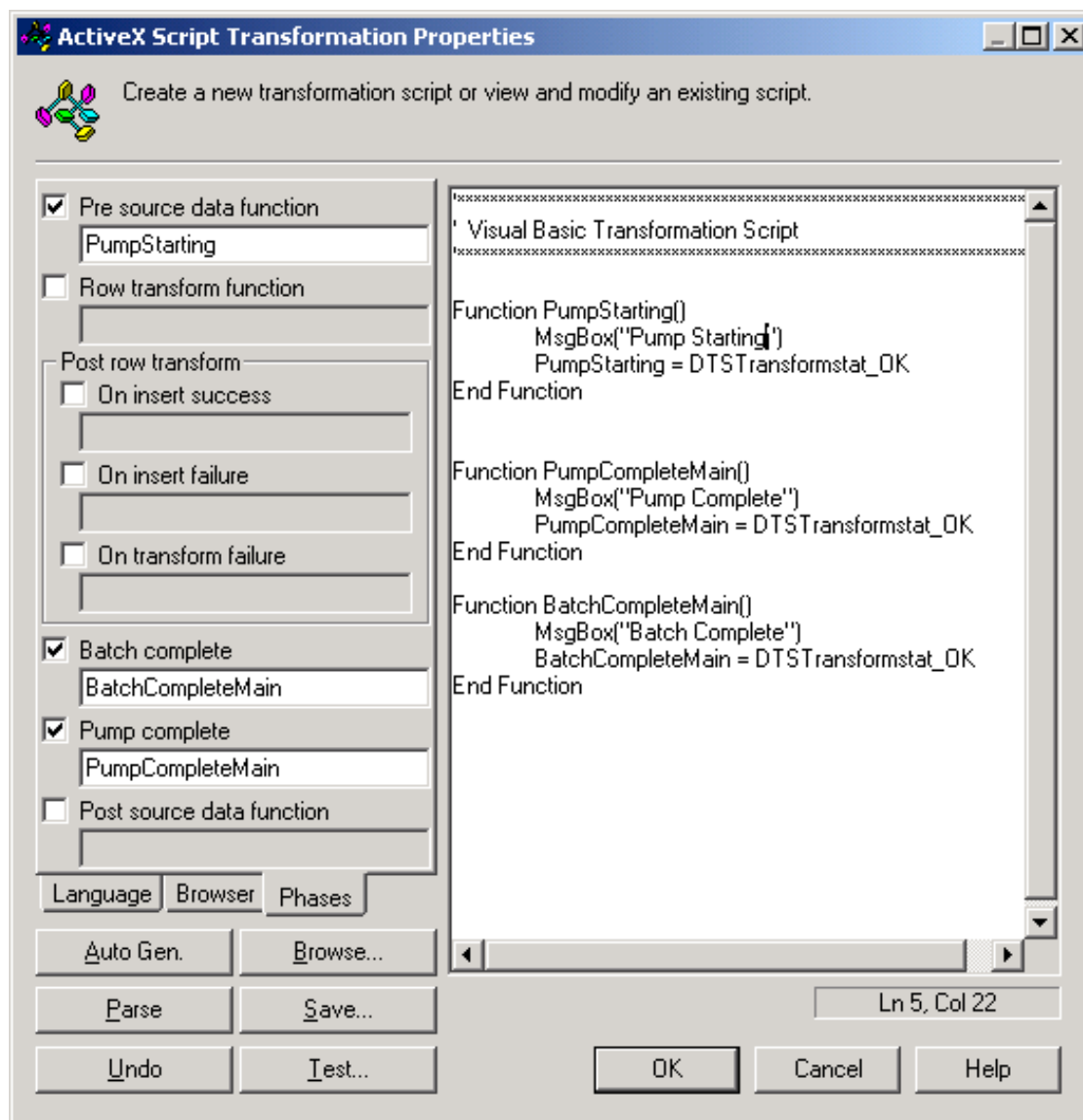
Function PumpStarting()
    MsgBox("Pump Starting")
    PumpStarting = DTSTransformstat_OK
End Function

Function PumpCompleteMain()
    MsgBox("Pump Complete")
    PumpCompleteMain = DTSTransformstat_OK
End Function

Function BatchCompleteMain()
    MsgBox("Batch Complete")
    BatchCompleteMain = DTSTransformstat_OK
End Function

```

It is also important to set the entry point for each function as shown in the screen shot below:



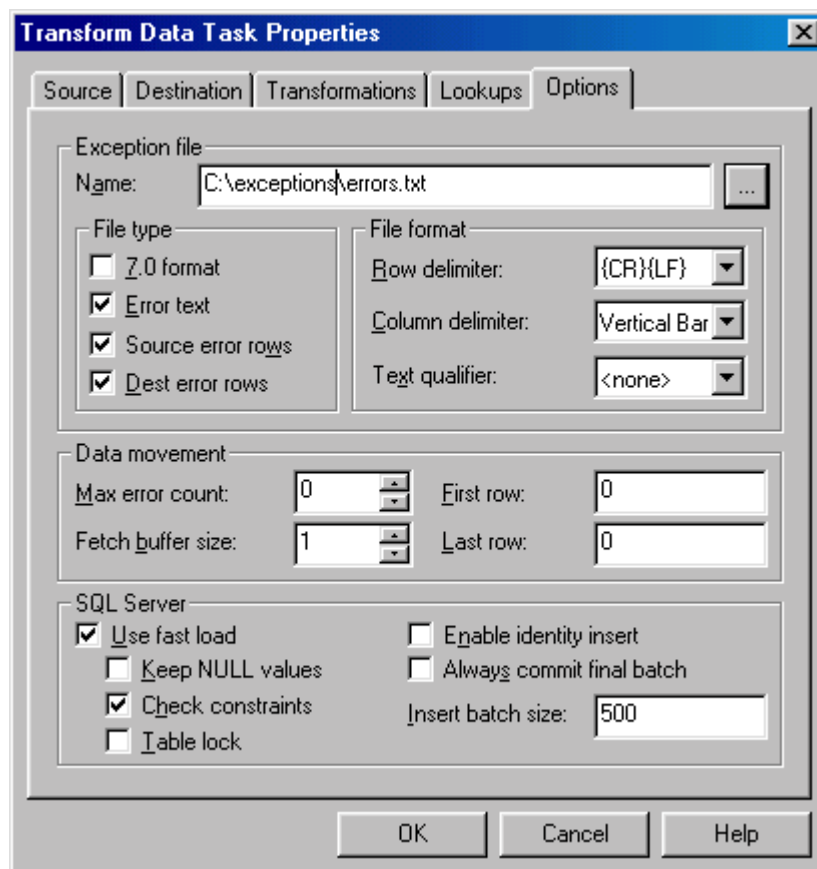
Other Task Improvements

Now, you also have the ability to pass variables into and out of the Execute SQL task. You can pass variables into the data pump tasks too, something that can be done by using a question mark placeholder in your queries. Then click the Parameters button to define which global variables the parameters will be coming from.

The Execute SQL Task now supports input and output parameters. If your Execute SQL outputs a rowset, any column can be set to a global variable in the Output Parameter tab. You can also set the entire rowset to be outputted to a single global variable. This can be used to cache rowsets until you need them in another task or package. For example, if you need to scan metadata from a source table to find out what table needs to be inserted into, you could use output parameters to store the destination table's name as a global variable and then dynamically configure the data pump task.

Tasks that used to be web downloads are now included in DTS after you install Analysis Services. These new tasks will allow you to control actions in Analysis Services, such as reprocessing a cube or training a mining model.

Another addition to the data pump tasks is the ability to write any exceptions to an error log. Exceptions occur when a row cannot be transformed. If this occurs, the Data Pump task can be configured to write the record before the transformation and after the transformation attempt to a file. You can also configure how you'd like the file to be delimited. This can all be done through the options tab in the task. Lastly, you can pipe out the actual error that occurred.



New DTS Tasks

A number of the new tasks Microsoft has added replace complex scripts and COM components that you had to previously write in version 7.0.

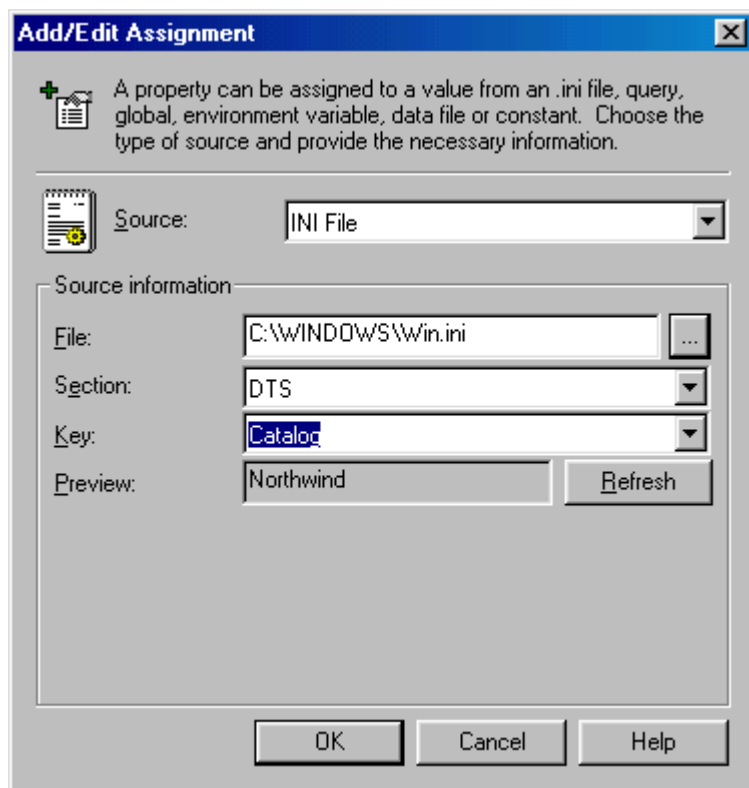
The major new tasks are:

- MSMQ – Pass MSMQ messages to and from packages
- Dynamic Properties – Dynamically set properties of any DTS object
- FTP – Receive files from an FTP server
- Execute Package – Make your packages modular by executing a package within a package

MSMQ allows you to send or receive data files, global variables, or strings. It gives you the ability to send messages between packages on any server. You can also communicate with any application that relies on MSMQ. Typically, you would use this task to make the parent package wait until all the children packages "check-in" during a data load process. Because of this, you can spread out a package's data load procedure over many servers and divide the work amongst them. MSMQ can sustain small to mid-size applications well. Large applications may experience bottlenecks if they use MSMQ and should use a more customized COM approach.

The Dynamic Properties task is one of the best upgrades to DTS. Previously, DTS programmers had to write complex ActiveX code in the ActiveX Script task to perform the same action. It allows almost any DTS property that is accessible to the DTS object model to be set to a:

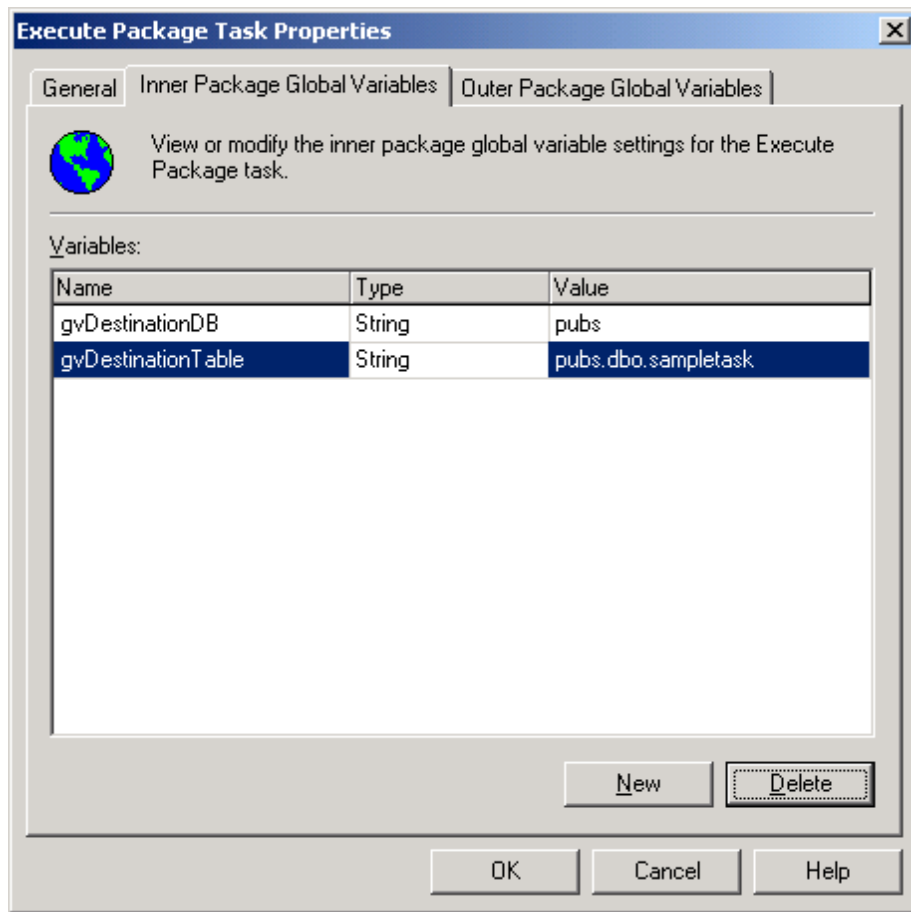
- Global variable.
- Constant that is typically used to assign the default value of an item, such as `DTSStepExecStat_Completed`. This is one way of resetting a value back to the default setting after changing it in another task.
- INI file that supports a single line property value as shown in the screen shot overleaf.
- Data file that can support multiple lines, but lacks in control, because it is an "all-or-nothing" option.
- Query that uses the first row returned. It is for this reason that it is recommended that you design your query in such a way that it only returns one row, like count, sum, or average queries.
- Environment variable from the System or User variables. These can be set in the control panel under system. Typical environment variables include `COMPUTERNAME`, `TEMP`, and `PATH`.



Setting the properties is not done until runtime. This leads us into the next hardest thing to program into DTS: the Execute Package task. Previously you had to do this in ActiveX script as well. Now you can execute a remote package and pass it parameters from the parent package or set parameters manually from the task.

The benefit of the Execute Package task is that you can now make your packages modular, up to sixteen packages deep. This allows you to easily have a standard company data pump task that is dynamically configured as you pass it parameters. It also allows you to subdivide parts of the package that are high security away from the main package. Only users that need access to the high security package can get it, but you don't hinder them when programming the main package. Programmers have often created a standard auditing package that they call from other packages as well.

There are two types of parameters that you can pass to the child package: inner and outer. Inner variables allow you to set the variables in the child package manually. The inner global variables are set manually at the task level, as shown in the screen shot overleaf. The outer global variables can only be set at the parent package level, not the task level. Outer global variables allow you to pass a parent package's global variable to the child. If the global variable doesn't exist on the child, it will be temporarily created in that instance and then deleted when the package closes.



It is extremely important to note that the task is version based. If you happen to change the child package, you must update the parent task to reflect the new version GUID. Otherwise, you task will continue to execute the old package version. Eventually when the version is purged, the task will fail.

Never use the Execute Package task to open its own package. This will cause an Access Violation (AV) and an endless loop.

FTP is a welcomed addition to DTS. It allows you to receive files from an internal or external FTP server. It does not allow you to push files to another server. The FTP task was previously done with the Execute Process task (executing the command line FTP). DTS programmers often created their own FTP components to plug into DTS. If you wish to push files, you will still have to do this.

There are quite a few DTS tasks that are tied into the CDW process, but they have little benefit outside the wizard. These tasks allow you to copy the following items that are tied to a database:

- Logins
- Jobs
- Databases
- Master Stored Procedures
- Error Messages

Summary

In SQL Server 2000, the capabilities of DTS have grown enormously. Many tasks have been added to make a DTS programmer's life easier. Previously you would have to create custom scripts to perform some of the actions that are built into DTS now like the Execute Package task. Additionally, all of the SQL Server 7.0 tasks have been modified to allow for multiple phases in data pumps and the acceptance of parameters. All of these new features allow you to create DTS packages faster and make them more customizable.

SQL Server 2000 Resources

Web: <http://www.microsoft.com/sql>
<http://msdn.microsoft.com/sqlserver/>
<http://microsoft.com/technet/sql/>
<http://www.swynk.com>

Newsgroups: microsoft.public.sqlserver.server
microsoft.public.sqlserver.tools