Welcome to DTS

A company's data is its life. Since the evolution of the mainframe and relational databases, companies have been evolving and increasing their productivity through database products like SQL Server and Oracle. The problem with evolution, however, is that some database systems are left behind. They are just too expensive to convert to the newer systems.

In the past, as a company upgraded its database system, a programmer would have to reprogram the communication layer through complex code. To remedy the time-to-market problem this created, Microsoft invented different ways to communicate with legacy and modern relational databases via an open standard layer called OLE DB. This breakthrough allowed programmers to communicate with IBM's DB2 database using code similar to that used to communicate to a Microsoft's SQL Server database. Suddenly portability became much easier.

This solved the problem of a program's portability, but what happens when you have a DB2 database that you need to convert to a SQL Server database? For example, your company purchases another company that still uses a legacy VSAM (mainframe driven) system. Your company, however, has its corporate database infrastructure in SQL Server. Converting the VSAM system may be too expensive and time consuming. In the meantime, you need the data that a sales representative is entering into a dumb terminal (DT) from the company you bought to flow to your SQL Server system in the shipping department. In the past, this process would have taken quite a while to program and employee turnover would be so bad from the merger that by the time you finished the development of a workflow system, you would have no one to support it.

This scenario is being repeated throughout the data processing industry daily. As companies merge, they need a way to rapidly develop an application to transition data from any source to any destination. Doing this on the mainframe is expensive and the human resources to do this are dwindling.

Developers also need to be able to transform data. This means that you can have the data look one way on the source and pass the data through a cleansing process to appear a different way on the destination. We can, for example, perform calculations to generate new values for loading, combine multiple columns into a single column, or conversely, break a single column into multiple columns. Imagine that the company you purchased calls sales territories by one set of names, and you'd like to call them by another; this could cause problems when the data was queried, as the values would appear to belong to different territories. However, you can allow them to keep their current system and just scrub the data (that is, make it consistent) as it enters your tracking systems. Another common example is seen when the fields FirstName and LastName need to become a single field named Name. In either case, programmers would have had to spend months staring at their computer screen turning over line after line to perform this simple translation.

DTS to the rescue! **Data Transformation Services** (**DTS**) was first introduced with the release of SQL Server 7.0. It was immediately hailed as a revolution in data transformation, since it was, after all, built into SQL Server.

> **DTS is a collection of utilities and objects that allow you to import, export, and convert data from any data source to any data source, whether to/from another database system (heterogeneous) or to/from another SQL Server. DTS isn't just about data! DTS can move SQL Server objects, execute programs, FTP files, and even has the flexibility to be expanded with your own custom COM components.**

Data source here refers to any source in an OLE DB, ODBC, or text file format.

In this chapter we will:

- Provide an introduction to DTS
- Explain the different types of connection methods to SQL Server using OLE DB and ODBC
- Dive into the DTS Architecture
- Explore the new DTS features in SQL Server 2000
- Examine the tools we can use to create packages, concentrating on the wizards to introduce the basics of DTS

Let's start by looking at exactly what we can do with DTS.

The Next Generation in Database Solutions

In SQL Server 6.5, data movement was done through clunky utilities such as **Transfer Manager**. Transfer Manager allowed us to transfer data with a single step from one SQL Server to another SQL Server. However, we had no access to any other data source, such as Oracle or Microsoft Access.

DBAs also had access to **Bulk Copy Program** (**BCP**). BCP is a non-logged event that inserts data into SQL Server in bulk fashion without any type of transformation. A BCP file's columns can be de-limited by a character such as a comma or a tab. BCP files can also use fixed column width to assign where each column begins and ends. The problem with BCPing files was that the schema positioning on the server and in the BCP file had to exactly match. For example, Column1 in the flat file had to be Column1 in the SQL Server.

Products such as **Data Junction**, produced by Data Junction Corporation, were released to fill the void. Data Junction allowed you to transfer data from just about any Database Management System (DBMS) or flat file, to any other DBMS. It was an extremely powerful tool, but expensive.

Today, developers and DBAs have access to the same power that existed in Data Junction, but through a tool that ships with all editions of SQL Server 7.0 and 2000 (including MSDE). DTS can act independently of SQL Server using a set of programs from a command line, or integrate tightly with SQL Server. This means that Oracle and Access users for example, can use DTS for their own data movement needs without having to buy a SQL Server license. However, if you do have a SQL license, you get the benefit of using the DTS Designer, which we'll look at more later on in this chapter. DTS Designer is an integral part of Enterprise Manager and no standalone program is available. If you don't have the DTS Designer, then you will have to develop DTS workflows on a system that has a SQL Server license and then port them to the other system, or program the workflows using the DTS object model, which we will discuss later in the book, and a COM compliant programming language. Keep in mind that if you connect to SQL Server in your workflow, you'll still need to have a SQL Server Client Access License (CAL). We will go into further detail about how to make DTS standalone in Chapter 2.

DTS allows users to convert data from any OLE DB compliant data source to any data source. This is done through a workflow system where you can create a process that sequentially moves data. It also allows you to expand the functionality by adding your own COM components. These components can be in any programming language that supports COM, such as Visual C++ or Visual Basic, for example.

What is OLE DB?

OLE DB is an API that allows COM applications, such as DTS, to communicate with almost any data storage system, including non-relational databases, e-mail, directories, and DTS packages. With OLE DB, services are made available via a **provider**, which is a dynamic link library (DLL) that allows you to connect to a data source. The OLE DB provider actually contains the API that is used by OLE DB.

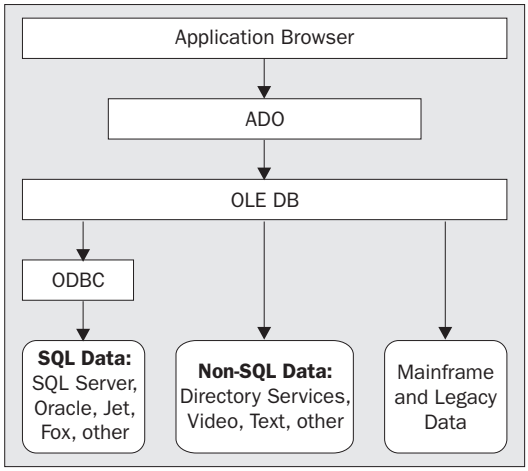
OLE DB is also extensible. This means that you can install new providers to connect to a countless number of data sources. For example, you can install the OLE DB provider for OLAP to access Microsoft Analysis Services cubes. OLE DB is the only connectivity tool that will allow you to connect to such systems. OLE DB has truly opened the door for open communication between database systems and allowed applications to become much more portable. For example, a web application could simply change its connection string in one file (`GLOBAL.ASA`) to move from connecting to Access to SQL Server.

A different type of common communication layer is **Open Database Connectivity** (**ODBC**). ODBC was Microsoft's first attempt at an open standard of data access. It was widely accepted around the industry as the data access standard years ago, and even grew acceptance in the UNIX world. An application that uses ODBC generally connects using Remote Data Objects (RDO). Most new development at Microsoft surrounds OLE DB because it offers more compatibility with other systems.

One of the OLE DB providers is the OLE DB Provider for ODBC. This substantially expands the list of database systems you can access. The OLE DB Provider for ODBC is considerably slower than the straight OLE DB one, however.

OLE DB is able to communicate with almost any system because it exposes the database's communication layer to the programmer through open COM APIs. A programmer can easily communicate to the provider through Active Data Objects (ADO). Because OLE DB uses these lower level COM API calls, you can communicate to almost any consumer.

In the diagram below, you can see the application using ADO to communicate to the database through OLE DB. Optionally, you can use the OLE DB Provider for ODBC to communicate to systems that may not have converted to OLE DB yet.



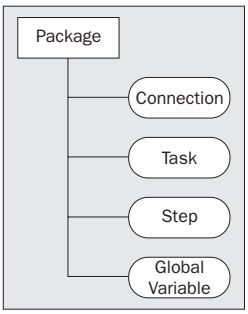
Additionally, you can install programs such as Microsoft Host Integration Services (formally SNA Server) to expand the available providers. For example, after installing Host Integration Services, you can communicate to DB2 and legacy systems. Some providers do take a little more work to configure. By default, Oracle providers are installed when you install SQL Server 2000, however, you must install Oracle tools (SQL Net) to establish connectivity.

The DTS Package

The core component of DTS is the **package**. The package is what you create and execute when you are using DTS. Every other object in DTS is a child to a package. A package is used to accomplish a particular goal using the various children that it may have. It contains all the connection objects and items that DTS will need to connect and transform the data. A package may not contain a connection to a database at all, instead executing a program or an ActiveX script. All of these objects are defined in a simple GUI called the DTS Designer. You can also design a package in any COM compliant programming language.

DTS packages can be saved into the local SQL Server, the Microsoft Repository, Visual Basic Files (SQL Server 2000 only) or COM-Structured files. You can also execute packages without saving them. It is only compulsory to save packages if you plan to schedule them for later execution.

A package has four major components. The four components, which are shown in the following figure, don't all have to exist for a package to be executable. For example, you can have a package with one task that executes an external custom program or process. In this case, you don't have any of the other three components.



Let's take a look at each of these components in turn.

Connections

DTS allows you to connect to any OLE DB compliant data source. Natively, DTS can facilitate OLE DB providers such as:

- SQL Server
- Microsoft Access
- Microsoft Excel
- Visual FoxPro
- Paradox
- dBase
- Text Files
- HTML Files
- Oracle

DTS also supports Microsoft Data Link files. Data link files are physical files that can be written to connect to a database at run time (SQL 7.0 compiles these at design time). Because they're files, they can be transported from system to system. The list of OLE DB providers can also be expanded. For example, on the Microsoft SNA CD-ROM (or HIS CD-ROM) there is an installation for OLE DB providers for IBM's DB2 database. Sybase providers are also available through the vendor's website.

More information about OLE DB providers can be found at http://www.microsoft.com/data.

We'll examine establishing connections in more detail in Chapter 2.

Tasks

The package holds a number of instructions called **tasks**. Without tasks, a package would be a car without an engine. There were 8 built-in tasks for SQL Server 7.0, which could do a number of things, such as:

- Transform data
- Bulk insert data
- Execute a SQL script
- Execute a program
- Execute an ActiveX script
- Move a database

There are 17 built-in tasks for SQL Server 2000, which have additional functionality including the ability to:

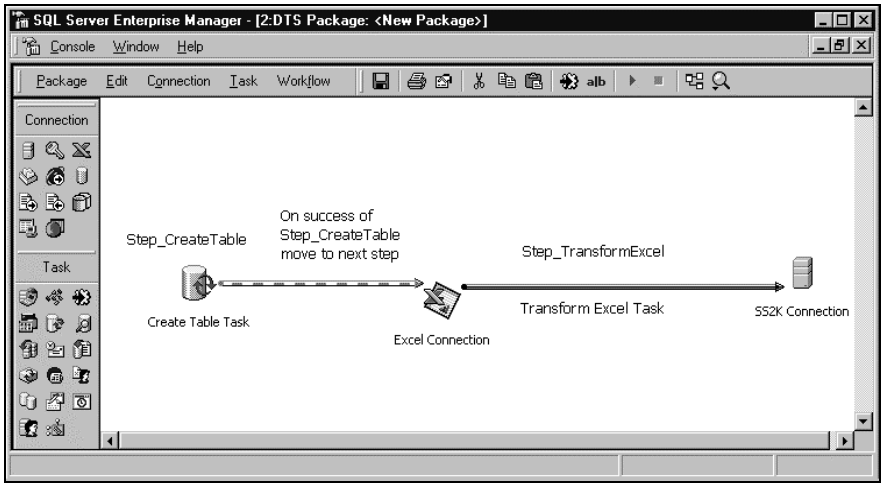
- FTP a file
- Execute another package
- Send or receive MSMQ (Microsoft Message Queue) messages from another package

We'll cover all of these tasks in Chapter 2 and 3. If you find that the built-in tasks do not fit your needs, you can expand the capabilities by registering your own custom tasks. A custom task can be written in any language that uses COM. Creating custom tasks will also be covered in a later chapter.

Steps

A **step** gives a package its logic. The step object allows you to connect tasks in a sequential order. Each task has one step associated with it that can either execute in sequential order or in parallel order, depending on how you have the package configured. The key difference between a task and a step is that the task object holds the information about the individual function that you're doing. For example, a task would contain what file you're executing. A step would tell the task when to execute.

You can also set constraints on tasks, which are called **precedence constraints**. Precedence constraints allow you to dictate if a task will be executed in the event of a failure, success, or completion of another task. For example, as shown below, step 1, which creates the table will have to execute and succeed for step 2, which transfers data from Excel to SQL Server, to execute.



Global Variables

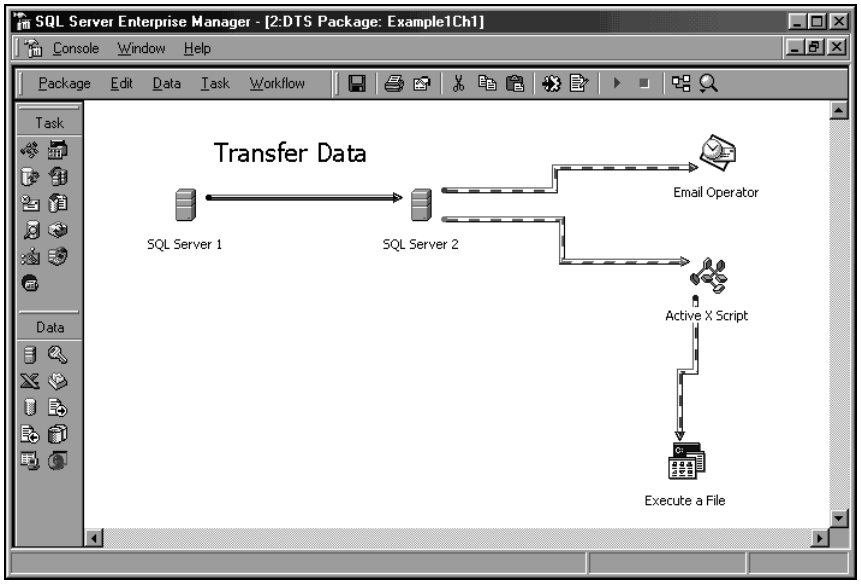
Global variables can extend the dynamic abilities of DTS. Global variables allow you to set a variable in a single area in your package, and use the variable over and over throughout the package, whether in an ActiveX script or a data transformation. They allow you to communicate with other DTS tasks and pass messages between them. Rowsets can also be stored into a global variable for later use by a different task.

An ideal case for using these variables would be to set up a dynamic database load. For example, say you receive an extract each day from a mainframe. The extract file name changes daily based on the client the extract is for and the date: for example, the filename `CLIENT1-010198.txt` would mean client 1 run on January 1 1998. By using an ActiveX script, you can read the file name, and change the global variable for the client number and run date based on the name you read. You can later read these two global variables to determine where to insert the data and what security to allow. We'll see a complete example of how to do this is in Chapter 8 (*Dynamic Configuration of Package Objects*).

> **In SQL Server 2000, DTS uses global variables to a greater extent. Various tasks have been added and modified to allow input and output parameters. Global variables act as a holding tank for these until they're needed. This is still possible in SQL Server 7.0. In SQL Server 7.0, you will have to write an ActiveX script to perform the same action that the GUI completes in one step.**

A Simple Package

Much of your package creation, as we will discuss later in the book, will be done in DTS Designer. An example of this is shown below:



In this package, data is transformed from SQL Server 1 to SQL Server 2. The execution of this task is considered a step. This is shown as a solid line with an arrowhead pointing at the destination. If that step fails, then the operator (system administrator) is e-mailed – the failure line (the one between SQL Server 2 and the Email Operator) is displayed in red in the Designer. If the transfer succeeds – the lower horizontal line, displayed in green in the Designer – then an ActiveX script is fired off to move the log files into an archive directory. Once the move is complete, a batch file is executed to send a broadcast message (using net send) to all network administrators of the package's completion.

This just displays the steps in the package – the result of the package execution is not displayed in the Designer. In the above example, the network administrators are informed of the package's completion, not its success or failure. The administrator would then have to go to the log files and determine if it was properly executed. You can also program more complex logic in ActiveX to send a message regarding the package's success or failure.

What's New in 2000?

SQL Server 2000 has expanded the power of DTS substantially. The DTS engine improvements that are discussed in Chapter 2 and 3 include:

- Ability to save packages to Visual Basic files. You can then place the `.BAS` file into your VB program.
- Integration with Windows 2000 security (Kerberos) and the ability for Windows 2000 users to cache packages.
- Ability to execute individual steps. In SQL Server 7.0, you had to disable all other steps to debug problems in one step. In SQL Server 2000, that's no longer necessary.
- Ability to run packages asynchronously.
- Support for Microsoft data link files (`.udl`). This allows you to create a `.udl` file that expresses the connection strings (the ones you define in your package to point to the source and destination) and to use it for your connection. In SQL Server 7.0, the `.udl` file was compiled at design time, now it's not compiled until runtime. This allows you to dynamically configure the `.udl` at run time.
- Addition of an HTML Web page source.
- Multi-phase data pump, which allows a failed insertion attempt to not fail the entire package, as well as giving you the ability to break a transformation into a number of stages, which are completely customizable. This is discussed in great detail in Chapter 3.
- Ability to edit a package disconnected from the network.
- Ability to save your DTS files as template (`.DTT`) files.

The most exciting addition to DTS in SQL Server 2000 is the new tasks. The complete list of tasks is discussed in Chapter 2. The added tasks are:

- File transfer protocol (FTP) transformation
- Microsoft Message Queue
- Dynamic Properties
- Execute Package

- Move Database
- Move Users
- Move Messages
- Move Jobs
- Move Master Stored Procedures

There are also some added tasks once you install Analysis Services. These include one to reprocess an OLAP cube and one to train a data mining model. Although the DTS user interface (the DTS Designer) has not had a major facelift, it does have some added functionality.

The DTS Toolbox

Microsoft has given us several tools to create our packages. A DTS programmer can create a package with:

- **Import and Export Wizards** – which automatically create the package for you after asking a series of questions.
- **DTS Designer** – a graphical user interface (GUI) that allows you to design and execute packages.
- **DTS Programming Interfaces** – series of APIs that are accessible through COM interfaces and any COM compliant programming language such as Visual Basic or C++ and scripting languages. Built into DTS is the ability to program in VBScript and JavaScript. We'll look at this much more in the Chapter 8. This can be expanded to any installed script, however, such as PerlScript.

As you might expect, the simplest way to create a package is through the wizards. As the wizards are also the easiest way to get a little hands-on experience of DTS, we'll take a look at these wizards here. However, as we'll see later, this method is also the weakest in terms of functionality. That's why, in the remainder of the book, we'll be using the wizards as little as possible and be exploring the alternatives.

In DTS, the easiest way to move data is through the built-in wizards. The wizards allow you to quickly create DTS packages or not even see the packages at all, and just execute them without saving them. They give you the basics you need to create a package to transform data, but lack depth. Most of what we discuss in the further chapters can't be done with the wizards. The wizards are only meant to transform data; any other functionality you might want is missing.

The wizards do provide a great way to create a basic package. The last step the wizard provides is the ability to save the package for later execution, and at that point, you can modify the package to add your own logic and the features that are missing. For example, you may know a flat file named `transform.txt` is going to be in a certain location daily for transformation. However, you'll need to fetch the file using the FTP task. You can use the wizard to create the transformation part of the package and then save it. After you save it, you can edit the package in the DTS Designer and add the FTP functionality.

Using the Wizards

SQL Server provides an array of wizards to guide us through exporting and importing data. There are two core wizards that help you move or copy data: The **Import/Export Wizard** and the **Copy Database Wizard** (**CDW**). The wizards are an ideal way to migrate data from development to production or to create the root of a package to add logic to later. In this section, we will work our way through examples that will teach us how to:

- Export and import data from a database with DTS Wizards (we only go through the export wizard – although the import wizard is almost identical).
- Export and import database objects such as stored procedures and triggers.
- Create packages through the wizards that you can later edit in Designer.
- Save and schedule packages for later execution.

In the examples, we will provide sections that are hands-on, so that you can begin to get a feel for DTS. Interspersed between these are sections describing the functionality of the other options available, to provide a comprehensive overview of what we can achieve with the wizards.

In SQL Server 6.5, we were able to transfer database objects using Transfer Manager. Transfer Manager is a distant relative to DTS, but it lacks some of the functionality that DTS provides. This functionality has carried over into SQL Server 2000 with the Copy objects and data between SQL Server databases option that we will discuss in a moment. Some of the differences include:

- A direct way to save jobs to add custom components into them
- Access to OLE DB compliant data sources other than SQL Server
- Customization with ActiveX scripts
- A workflow type system

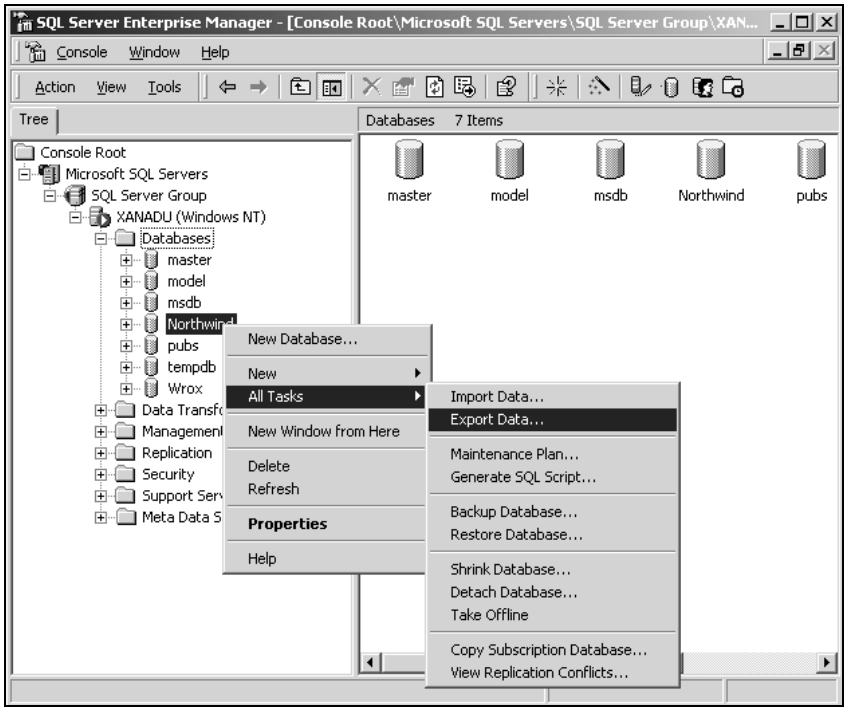
Transferring Data Using the Import/Export Wizard

Accessing the Wizard

In this example, we'll import data from the SQL Northwind database into a database called Wrox. To create the empty Wrox database, click your right mouse button in Enterprise Manager under the database tree node and select New Database. It will not matter where your database is located.

You can also transform your data in this example into the TempDB database. The next time your SQL Server stops and starts, the tables you create in this example will be purged. The TempDB database is a nice way to test transformations before you move them over to the live database, and it automatically cleans itself out after the server cycles. Never use the TempDB database for this purpose in production, however, since you could slow the performance of some queries by doing this.

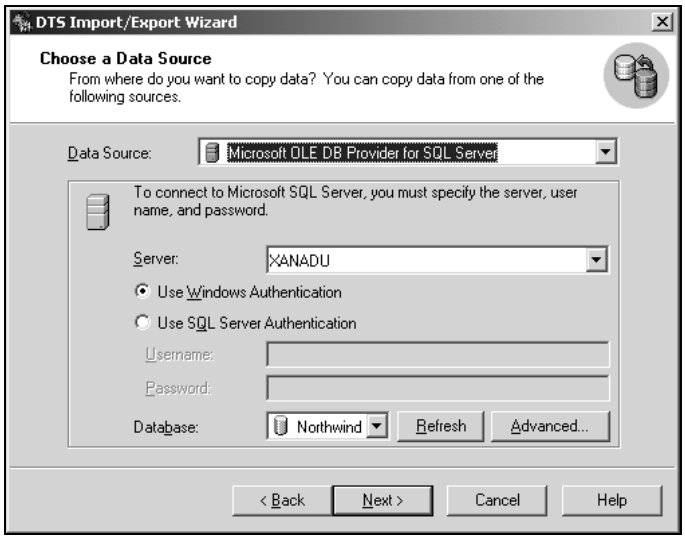
As with most wizards, there are many ways of accessing them. You can access the wizards (as long as you have highlighted Databases) through the Tools menu, Wizards, then Data Transformation Services (or Management for the Copy Database Wizard); or Tools, then Data Transformation Services. However, the primary way to access the DTS Import and Export Wizards is to open SQL Server Enterprise Manager, and click with your right mouse button on the database you want to import/export from – in this case, the Northwind database. Then select All Tasks, and Export Data:



Once you're in the wizard, make sure you read each question very carefully. The wrong answer could result in you purging records from your tables unintentionally. If you're transforming into a production database or any database that contains data you care about, make sure you backup the database before running the wizard. Note also that entries are added into the log, so that we can recover the data, if necessary.

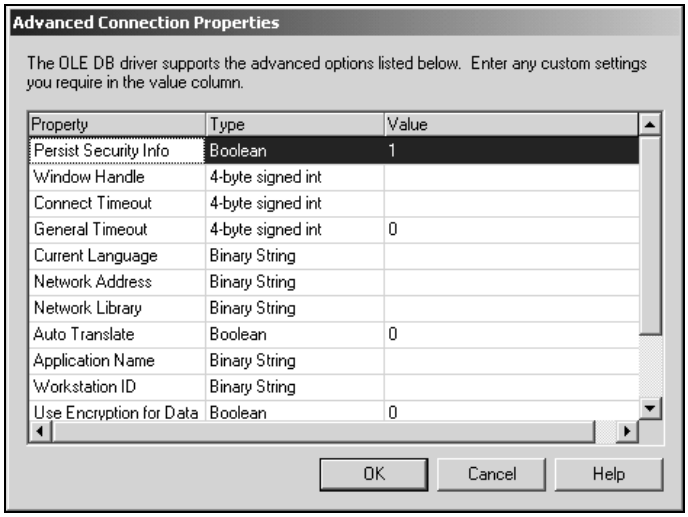
The reason we choose to access the Import/Export Wizard by clicking our right mouse button on the database is it takes a step out of the process. Since we did this, the first screen overleaf is already fully completed for us. If you came into the wizard another way, and had not highlighted the Northwind database first, you would need to select Northwind as the source connection database in the drop down box at the bottom of the screen. You will need to enter the necessary security information (username and password). As in ODBC data sources, the OLE DB provider for SQL Server supports either using standard SQL Server or Windows Authentication. Those using Windows 98 on their machines will have to click the Refresh button to see the database listings populate.

The first screen you see prompts you for your Data Source, defaulting to the OLE DB Provider for SQL Server:



Advanced Options

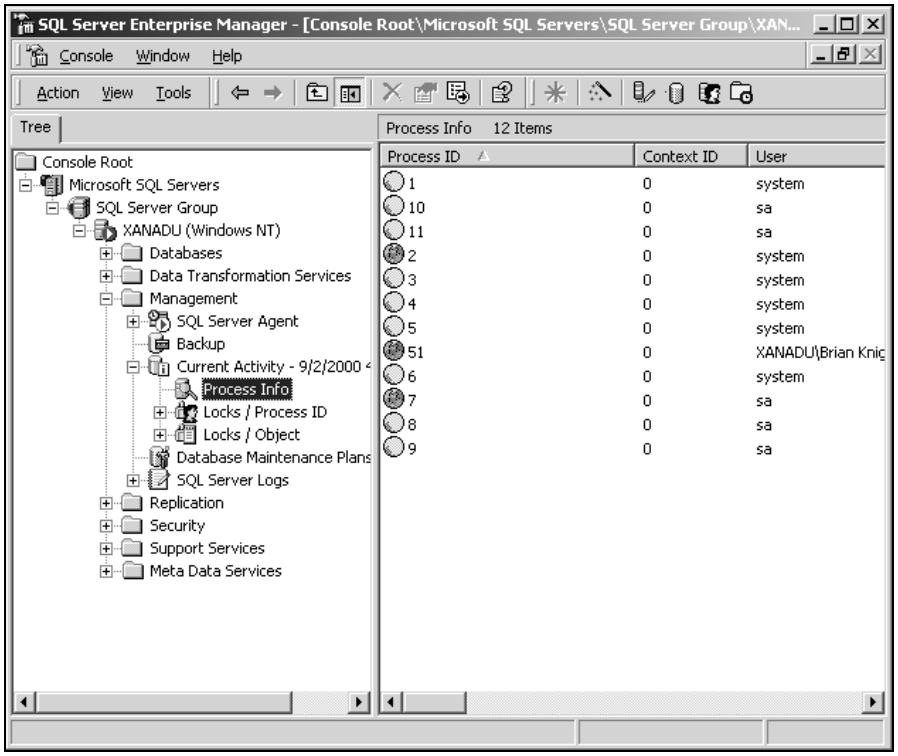
If you need more advanced OLE DB connection options in your transformation procedure, click the Advanced… button to reveal the screen below:



The dialog box that will appear will present you with a number of OLE DB connection options that are specific to the provider. In the OLE DB provider for SQL Server you will see options such as increasing the connection timeout, or specifying an IP address for the source server. Note that if you need to specify any Boolean values, they must be represented with a 0 (`False`) or 1 (`True`).

The Application Name and Workstation ID options are nice for auditing purposes. Each database management system (DBMS) has its own way of auditing who is connected, but in SQL Server if you adjust these two options, you can run a stored procedure named `SP_WHO2` from the server, and detect which application and server name is connecting.

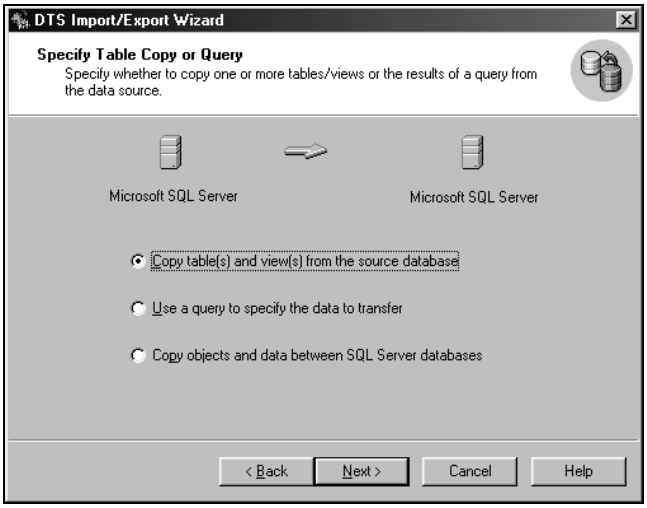
Note that you can also view connection information in Enterprise Manager. If you open Enterprise Manager and drill down into the Management node, you can select Current Activity, then Process Info to see the current connections. The Process Info item, as shown below, is the only item that contains the level of granularity to let you see with which applications users are connecting. This is a handy tool to debug connectivity problems and to determine if your connection request is even making it to the server. Active connections performing a query are lit, while sleeping connections are gray.



Choosing the Destination Database

Once you've set all the options in the Choose a Data Source page, click Next. The Wizard will then prompt you to select the destination database. Repeat the above steps for the destination database, this time selecting Wrox as the database. Again, you may have to click Refresh to see the Wrox database in the drop-down box.

On clicking Next, you'll be presented with the following options:

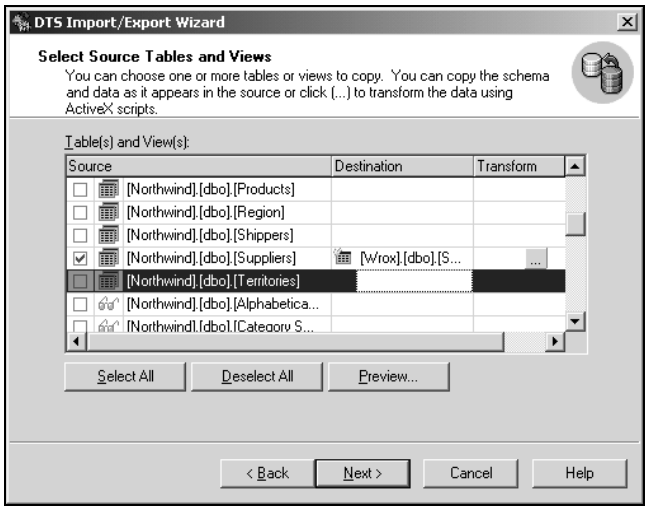


We'll look at what each of these does in turn. For the purposes of the current example, choose the Copy table(s) and view(s) from the source database option.

Copy Tables and Views Option

The easiest of the three options is to select Copy table(s) and view(s) from the source database. This option transfers data from the source database, but it does lack in selectivity. In other words, this option will select all data from any given table. If you wish to selectively transfer data, use the next option (Use a query to specify the data to transfer), which we will examine later in the chapter.

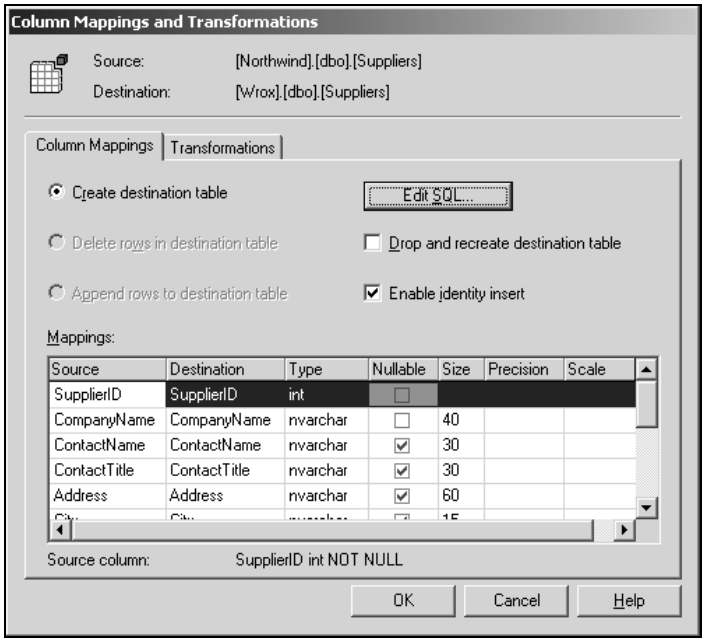
The first dialog box in this section of the wizard will ask you which tables and views you'd like to transfer into the Wrox database. For the purpose of our example, select only the Suppliers table from the source column:



The destination column will automatically be filled after you check the source. This does not mean that the table exists in the destination database. By default, if the table does not exist in the destination database, then the DTS engine will create it for you. If the table does already exist, then DTS will by default append the new data to the existing data. The destination column will also allow you to select another table from a drop-down list if you've already created one that's more suitable.

Advanced Options – Column Mappings Tab

In the Transform column, you can click on the three dots to select some more advanced options. If your destination table does not have the same schema as the source table or view, then you could specify which columns to transfer and their mapping here. You can ignore certain columns by clicking on them, then selecting the <ignore> option from the Source drop-down box. The reason you'll see a drop-down box for only the Source column is that if you adjusted the Destination, you would receive an error. The error would be generated by DTS sending data to a column that's being ignored. A typical reason you'd ignore columns is if you only wanted a subset of the data vertically, or if you wanted the destination table to assign the identity column and not take on the identity from the source.

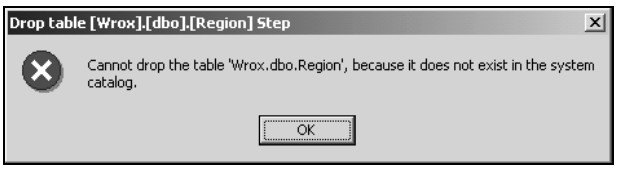


The options that are not needed are grayed out. For example, you can't append data to a table that doesn't exist. Let's have a look at options provided by the Column Mappings tab (although we will not apply any of them in this example):

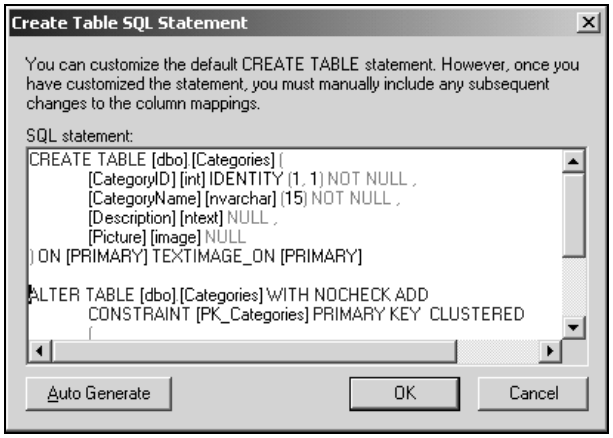
- Create destination table. This will also allow you to customize the transformation into that table. If the table already exists, you are still given the option; however, you would need to select the option to Drop and recreate the destination table. Otherwise you would receive an error on execution complaining about a name conflict in creating the table.
- Append rows to destination table. If you were receiving incremental updates from a source, you would use this option.
- Delete rows in destination table. Use this if you want to complete a data refresh without dropping and recreating the table.

- Enable identity insert. This option will turn off identity fields temporarily during the insert. This is only needed if you're inserting into a table that has an identity column (or a column that's auto numbering a column). You would receive an error upon inserting data into a table that has an identity column unless you have this checked. This is because the system wants to auto assign the new data a number. This option is automatically checked if there is a column on the destination with an identity column in it.
- Drop and recreate the destination table. Use this option if you want to delete the table and recreate it with a different schema.

As you can see, the append rows and delete rows options are not available when you have the Create Destination Table radio box selected. The Create destination table radio box is automatically selected when the table does not exist on the destination. This is because the table doesn't exist. If you select the box Drop and recreate the destination table and the table has not been created, you'll receive an error when executing the package as shown below. This is due to DTS issuing a drop command of a table that doesn't exist yet. The DTS engine will detect that the table does not exist on the destination and not allow you to select, append, or delete from the table.



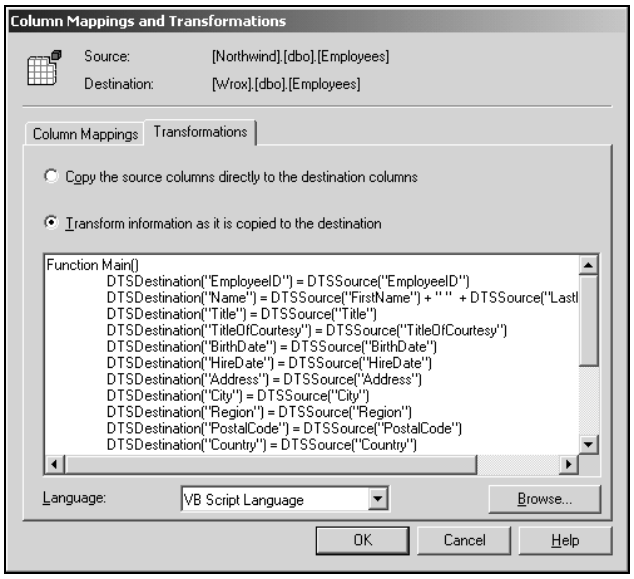
You can also click the Edit SQL button to modify the script used to create the table by hand. For example, you could add statements to this to add the necessary primary and foreign keys. You will also need to add any constraints or identity column information in here as shown below. The dialog box does not support `GO` statements, which you will receive by default if you generate a script from Enterprise Manager. There is an unusual workaround that you must do to create primary keys in this fashion. After you try to exit this dialog box and go back to the transformation screen, DTS performs a check on your SQL code. When it can't find the table, DTS will return an error since you can't issue an `ALTER TABLE` statement (which is used to create constraints and primary keys) on a table that doesn't exist yet. The workaround is to create a shell of the table on the destination with one column:



It is because of this workaround that I would recommend that you just create tables that need Primary Keys and constraints on the destination before you try to transform the data into them. You can also run through the wizard and then add the constraints afterwards. You can also select the Copy objects and data between SQL Server databases branch of the wizard, which we will cover shortly in this chapter. This feature is still nice if you'd like to modify the schema slightly on the destination. Those who participated in the Beta program will remember that this wizard had the Foreign and Primary key capabilities in it during the beta phases. It was pulled in the production, but Microsoft has promised to revisit this ability in a later release of SQL Server.

Advanced Options – Transformations Tab

You can also perform more customized transformations. By going to the Transformations tab, as shown in the following screenshot, you can write ActiveX logic to transform the data before it is committed to the destination. For more complex logic, use DTS Designer. It offers more flexibility than the wizard's transformation tab will offer. To enable this option, select the option to Transform information as it is copied to the destination.



As we will discuss in a later chapter, after you select this option, your transformation of data will slow considerably because records must pass through logic in a script versus the built-in COM object that DTS provides. You can write your script in any scripting language that is installed on the client running the package. The only exception is if you want to schedule the package. In that case, you can only use scripting languages that are installed on the server, since SQL Server Agent is actually executing the package. To change the scripting language, select the new language from the drop-down box seen above. The fastest of the scripting methods is VBScript followed by JavaScript. If you chose a language other than those two, REXX for example, you take on a slight risk because DTS may have not been tested in the scripting language.

Why would you use this feature then if it is considerably slower? You can use this to perform a number of functions that you couldn't perform with a straight copy. For example, if you received two fields on the source, `FirstName` and `LastName`, and wanted to merge the fields onto the destination as just `Name`, you could do so with a transformation script shown overleaf. Make sure you add a space between the two fields with an empty string `" "`. You can separate fields with the plus sign (+).

```
DTSDestination("Name") = DTSSource("FirstName") + " " + DTSSource("LastName")
```

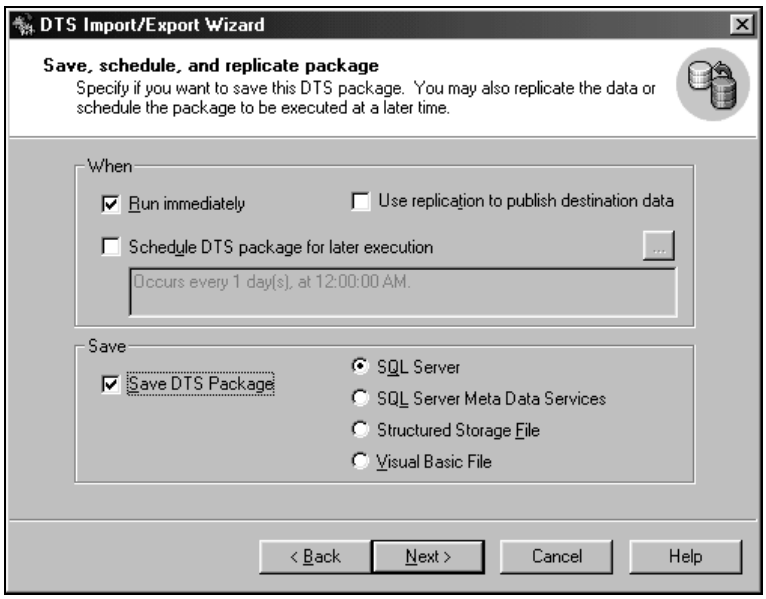
You can also use this feature to make fields upper case or to convert a zip code to the proper format. For example, the Customers table in the Northwind database has a number of postal codes from around the world, some alphanumeric, some numeric. With VBScript, you could determine if the postal code met a certain requirement, like length, and then transform it. Other fields would be copied straight through. The code below shows you an example on how to do perform such logic. Don't worry about the code quite yet. Several chapters are dedicated to how to do this in more detail.

```
If LEN(DTSSource("PostalCode")) = 9 Then
DTSDestination("PostalCode") = Left(DTSSource("PostalCode"),5) + "-" +
Right(DTSSource("PostalCode"),4)
Else
DTSDestination("PostalCode") =DTSSource("PostalCode")
```

As we mentioned earlier, the DTS Designer is a much better place to be performing this type of transformation. Quite a few pre-defined transformations have already been set up for you in the Designer. For example, you can easily make a field upper case in Designer with one click and no programming.

Saving and Scheduling the Package

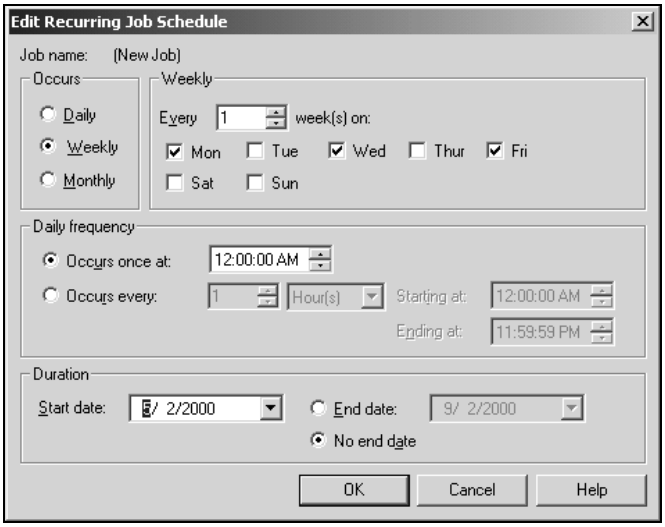
After clicking OK and Next, you will be prompted to save, schedule, and execute the package. For the purpose of this example, check the box that causes the package to Run immediately, and save the package onto the local SQL Server by selecting the following options:



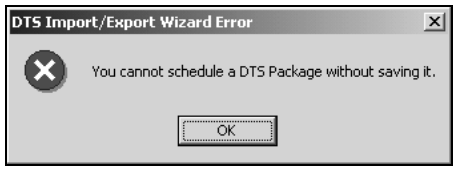
We'll go into details about the various places to save your package in Chapter 2.

Advanced Options

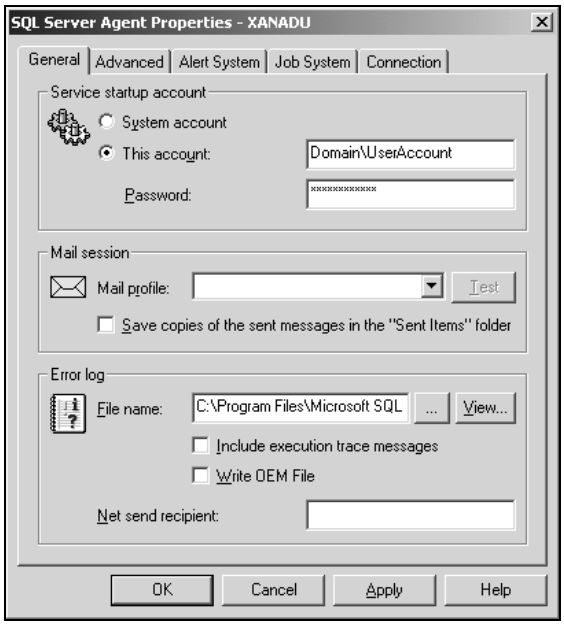
If you check Use replication to publish destination data, the Create Publication Wizard will begin after the data Import/Export Wizard finishes transforming the data. You can then set up you destination database to replicate to other sources, or schedule a package for later execution. This option will only work if SQL Server Agent is running. If it is not running, the wizard will insert the record into the MSDB database, which holds job information, and schedule the job, but the job will not be executed. You will have to start it manually from Enterprise Manager or wait until the next scheduled cycle arrives. A scheduled job can be manually executed by drilling down to the Jobs node under SQL Server Agent (which is under Management).



You will have to save the package before it can be scheduled. If you do not save the package, you will receive the below error:

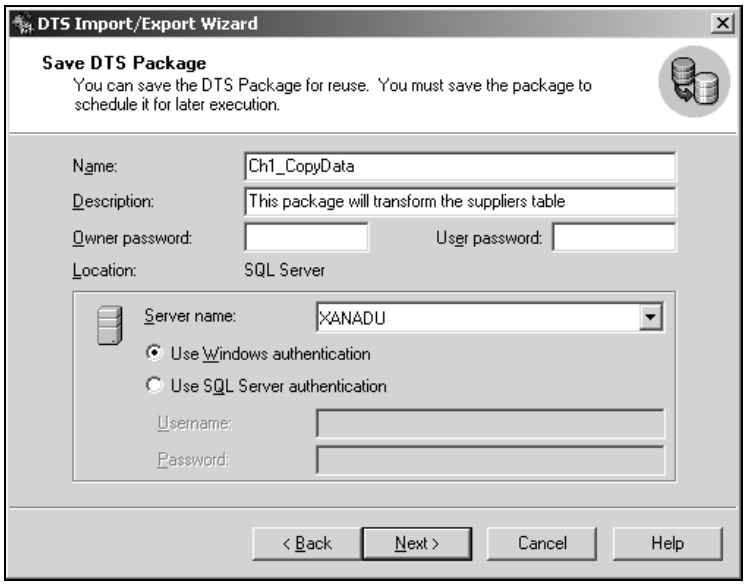


If you schedule a package for later execution, SQL Server Agent will execute the package as a `CmdExec` job using a utility called `DTSRUN`, which we will discuss later in the book. If your source and destination connections use Windows Authentication, then you will need to ensure that the account that starts the SQLServerAgent service has permissions to the destination and source database, and is started. You can change the permission for SQL Server Agent by drilling down under the Management group and clicking on SQL Server Agent with your right mouse button, then selecting Properties, to reveal the screen shown overleaf:



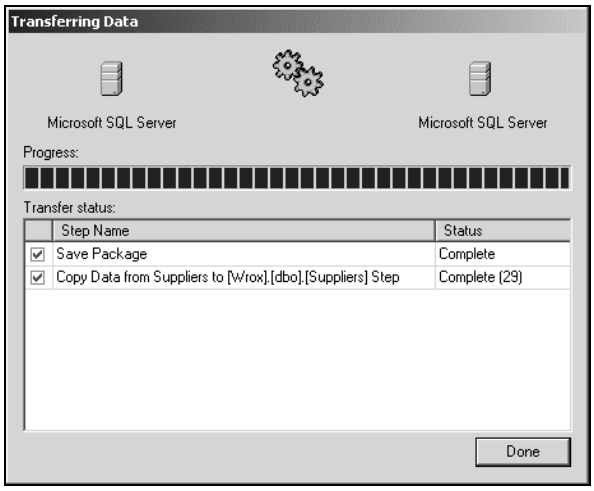
Naming and Executing the Package

When you have completed the information required on the save, schedule and replicate package dialog, and clicked Next, you will be prompted to name the DTS package that you have just created. Name this example package Ch1_CopyData. You can also specify an owner and user password here. We will discuss this in the next chapter; so in the meantime, leave this blank.



Make sure that you also add a meaningful description in the appropriate box. As you add more and more packages to your server, naming conventions and descriptions become increasingly important. The descriptions and names you type in these examples can later be loaded into the Microsoft Meta Data Services as meta data (data about your data). This meta data will allow you to view details about your package, the connections that it has and other descriptive data.

Click Next, and you will be presented with a summary screen. Clicking on Finish will cause the package to execute, first saving the package in the event that an error occurs. If an error occurs, you will be given an opportunity to go back and correct the problem. Data that has already been transformed will stay committed on the destination database.



You should find that the Suppliers table has been added to your Wrox database.

The Import/Export Wizard is really designed to transfer data rapidly from source to destination. It does not provide much room for customization. Once your data has been transferred, if you didn't alter the SQL statement used to create the table, you will have to create the primary key and foreign key relationships. Also any type of other information, like identity fields, will need to be created. You will not have to worry about this if you're transferring into a table that already exists.

If you recall, Copy table(s) and view(s) was only one option that our Wizard offered us for transferring our data. What about the other options?

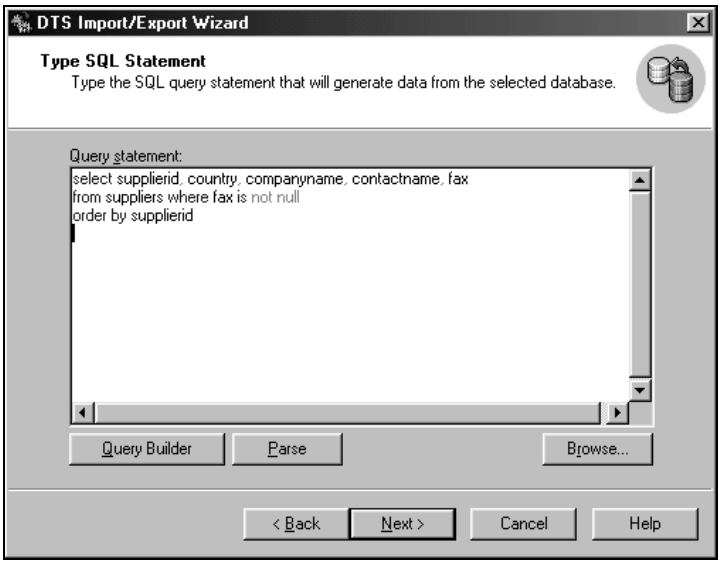
Using Queries to Transfer Data

Follow the steps above to get back to the Specify Table Copy or Query screen. This time we'll export only a select set of records into the Wrox database from Northwind.

In this example, we'll create a table that will be used for bulk faxing. Some of our data in the source table is useless for this purpose and needs to be excluded.

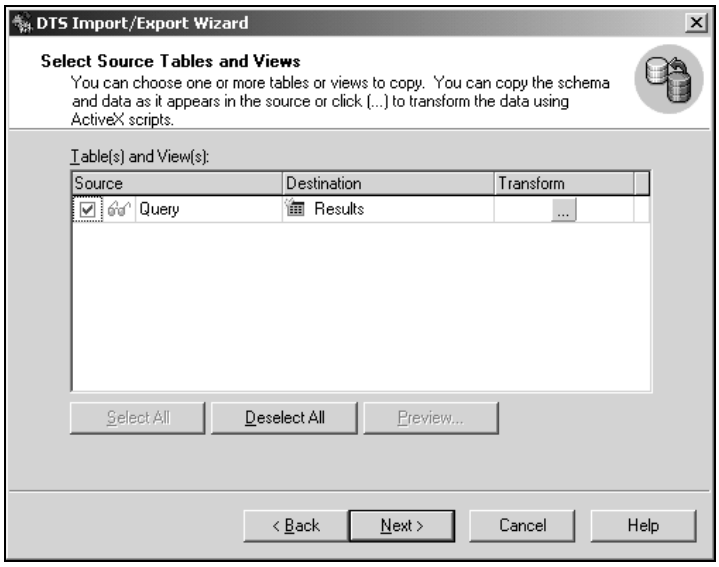
Select Use a query to specify the data to transfer. You can transfer data from one or more views or tables with this option. You are then given the option to type a query into the dialog box (Query statement) or use Query Builder to help you create the query. We will go into more detail on the Query Builder in a few moments. Since our query is simple, just type the following query in the Query statement. This query will select all the suppliers that have entered a fax number:

```
select supplierid, country, companyname, contactname, fax
from suppliers where fax is not null
order by supplierid
```



The Parse button will confirm that your query is a valid one. The Parse button checks all object names to makes sure that everything exists. The Browse button will allow you to find a prewritten script.

After you have continued on to the screen below by clicking the Next button, you may want to click on the "…" button under the Transform column to adjust the destination table schema, as shown earlier. The default table name that the data will be transferred into is Results, as shown below. You can only do one query at a time with the Import/Export Wizard.



Execute your package as you did in the previous example (chose to run it immediately and save it the local SQL Server), but this time save it as Ch1_QueryResult. Again, you will see the same screen as you did before as it executes step after step. First the package will save as previously, then transform your new table named Results.

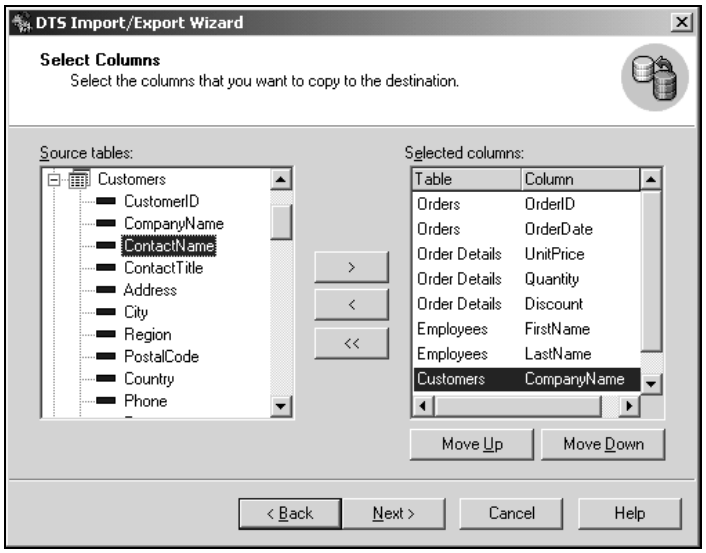
Query Builder

We briefly touched on the Query Builder in the last example. The Query Builder allows you to easily build queries that can include multiple tables from the same source database. You cannot go outside the source database or select views in the Query Builder. It is a superb way to build quick queries for those who are do not like the rigors of programming a nasty inner join in T-SQL. You will see Query Builder is available in a number of DTS tasks.

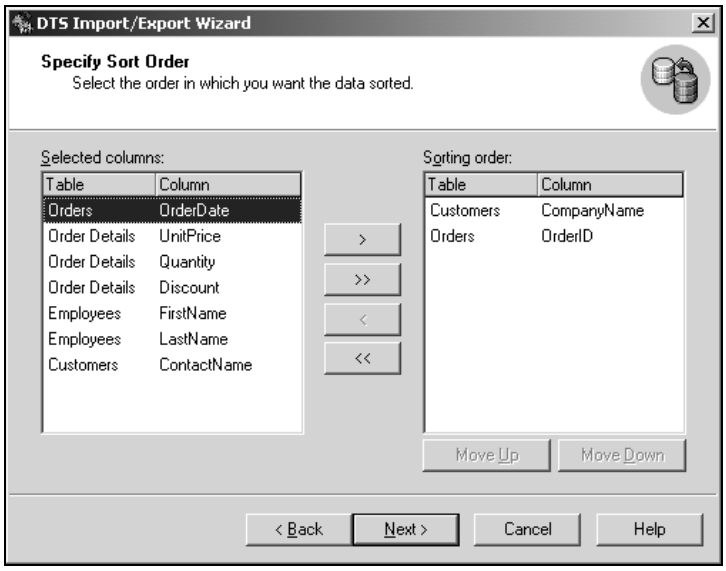
Once again, repeat the steps to get back to the Query statement screen seen previously. This time open Query Builder, chose the columns you want in your query and double-click on them. The interface will only allow you to select one column at a time.

In this example, we are de-normalizing the database slightly. This means that we're translating all the foreign keys into their real data. We've taken CustomerID in the Order table and joined it with the Customers table to find out the customer's CompanyName.

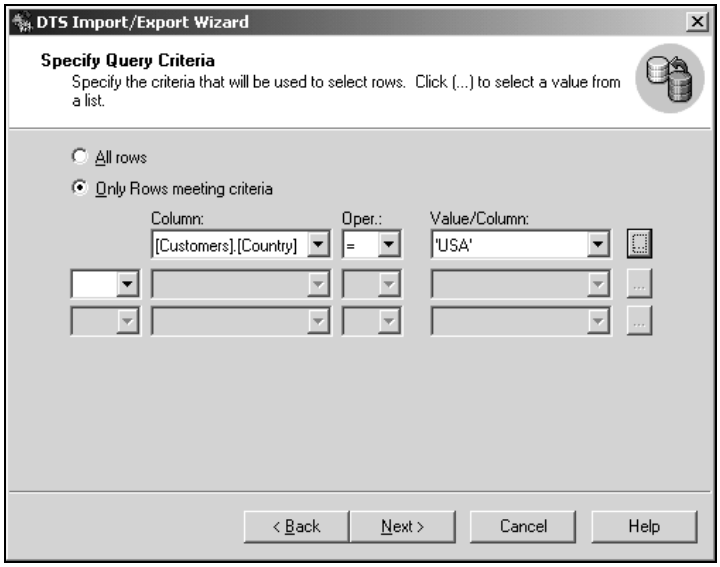
Begin by selecting the columns shown in the next screenshot:



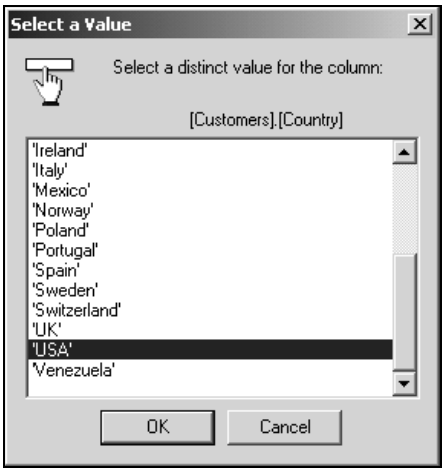
Next, select the column(s) on which you want your query to be ordered. Unless you change this later in the Query statement screen, the query will be sorted in ascending order. In our example, the CompanyName column will be ordered first, beginning with the 'A's, followed by the OrderID. This performs the same action as an `ORDER BY` clause. Query Builder does not have the ability to do `GROUP BY` clauses. `GROUP BY` clauses allow you to see the combined orders that each customer made. If you want to do a `GROUP BY`, then you will have to make the adjustments in the Query statement screen we saw earlier.



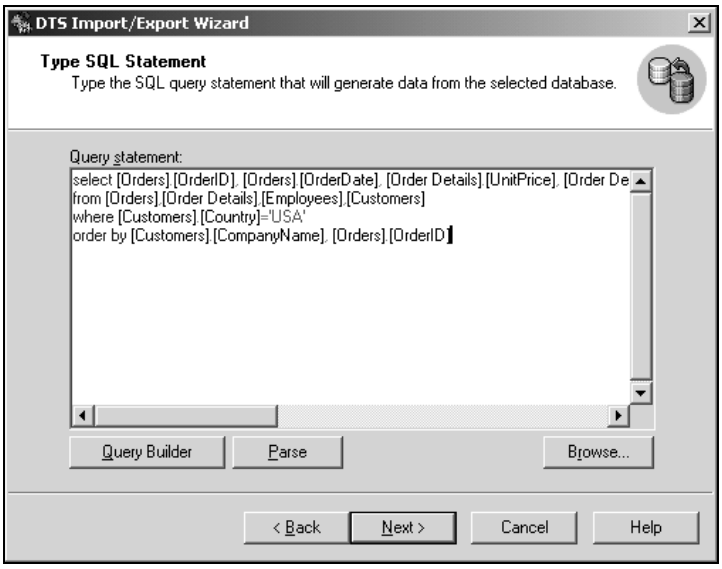
In the next screen, you can set the criteria for the query, by filtering your query horizontally based on another column, or data in the column. This screen performs the same action as the `WHERE` clause. The first Column: drop-down box will set the column you want to filter. Then, the Oper: drop-down box sets the operator that will be used (=, <, >, <>). Finally, setting the Value/Column: drop-down box makes the comparison.



To select an actual value, you can select the browse button ("...") next to the Value/Column: drop-down box. To select a value, double-click on the value and it will populate the drop-down box.



You are then returned to the Query statement screen where you can then modify any part of the query by hand, now that the hard part of your query is written:

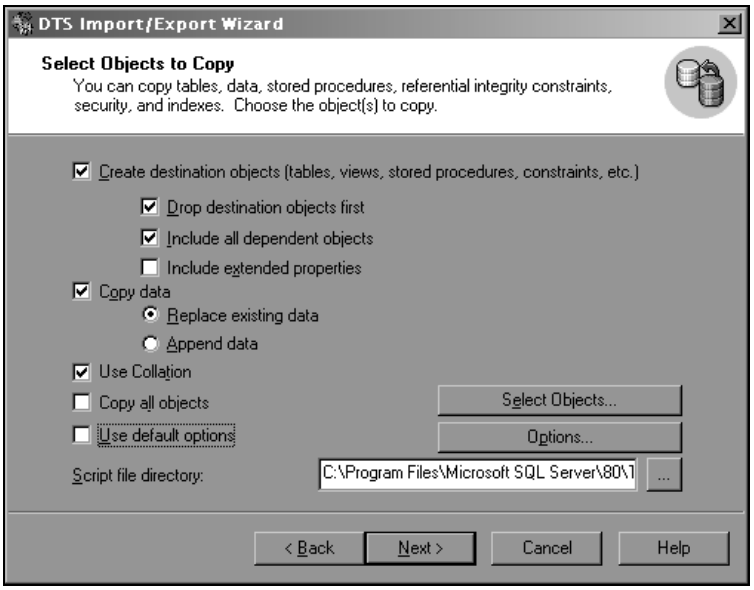


Some of the limitations of Query Builder stop most programmers from using it. The inability to perform a `GROUP BY` clause or sort in a descending manner is a big restriction. Most programmers find themselves writing their query in Enterprise Manager then copying and pasting the query into the Query statement screen. By right-clicking on any table in Enterprise Manager and selecting Query from the Open Table option, you can write a very effective query without any of the restrictions.

Transferring SQL Objects

We will now look at the final option on the Specify Table Copy or Query screen. The Copy objects and data between SQL Server databases option is a close relative to Transfer Manager in SQL Server 6.5 and gives you the ability to transfer SQL Server objects between databases. You can transfer any SQL Server object from any SQL Server 2000 instance to any SQL Server 2000 instance. You can also go from any SQL Server 7.0 instance to 7.0 instance or upgrade to 2000 with this branch of the wizard. To demonstrate how it works, we're going to copy selected objects from the Orders table into Northwind.

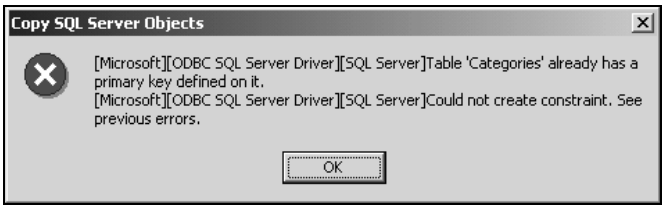
Again, follow the steps to get back to the Specify Table Copy or Query screen, and select Copy objects and data between SQL Server databases. You should see the screen below as the first screen. This screen is similar to Transfer Manager.



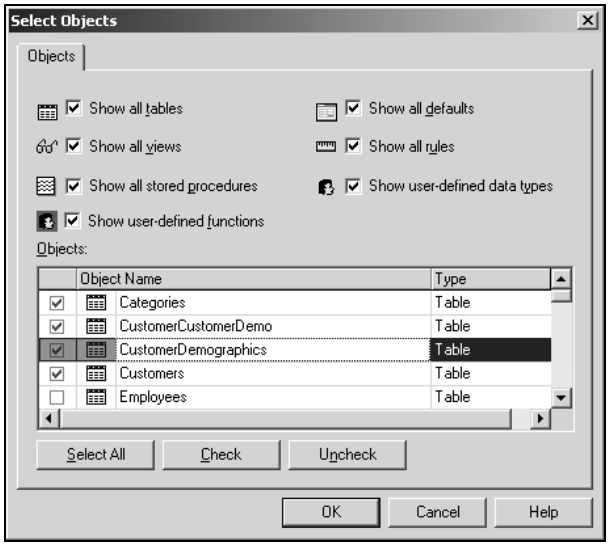
Advanced Options

You can configure this branch of the wizard to perform the following functions:

- The Create destination objects option will create the objects you wish to transfer on the destination server. Uncheck this option if you wish to only transfer data.
- The Drop destination objects first will drop all the objects on the destination SQL Server before it issues the commands to create them again. Check this option if you think that the objects you're trying to transfer may already be on the destination server, and you'd like to re-create them. If you run through the wizard and experience errors, you may want to check this option as well. Otherwise, you may see the error shown above opposite which states that the object already exists on the destination server. This is because you went part way through the transfer and there is no rollback performed.

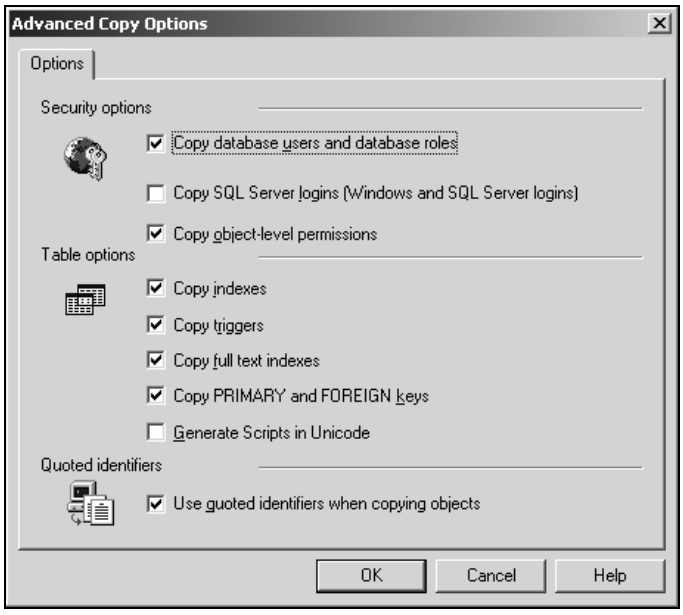


- The Include all dependent objects option will transfer objects that depend on the table you're trying to transfer. For example, any views that depend on the table will be transferred if you select this option.
- The Include extended properties option will transfer all extended properties on SQL Server 2000 databases. This option does not apply if you're transferring objects from a SQL Server 7.0 database.
- The Copy data option will enable you to transfer the data from the source to the destination server.
- The Replace existing data will purge the source tables before it transfers the new data into it.
- The Append data option will add the new data from the source at the end of the table on the destination server.
- The Use collation option will enable you to transfer data between servers of different collations.
- The Copy all objects option will transfer all objects in the database and not allow you to select certain objects to transfer. This includes all tables, views, stored procedures, functions, defaults, rules, and user-defined data types. If you uncheck this option, you are given the option to select which objects you would like to transfer. The Select Objects button allows you to check the objects you'd like to transfer from the source SQL Server, as shown below:



- The Use default options checkbox will transfer the other SQL Server objects like indexes, users, and primary and foreign keys. If you wish to specify other attributes to transfer, such as logins, you must uncheck this option and select the Options button. This will open the screen overleaf, which will allow you to specify more advanced options.

Generally, it is not a good idea to specify any other options other than the default. We recommend that if you want to transfer logins, for example, you just create those on the destination server by hand. The Copy SQL Server logins option does not give you control of what logins you transfer. One of the other options that are not checked by default is the Generate Scripts in Unicode checkbox. This option is nice if you have a number of Unicode fields on the source (`nchar`).



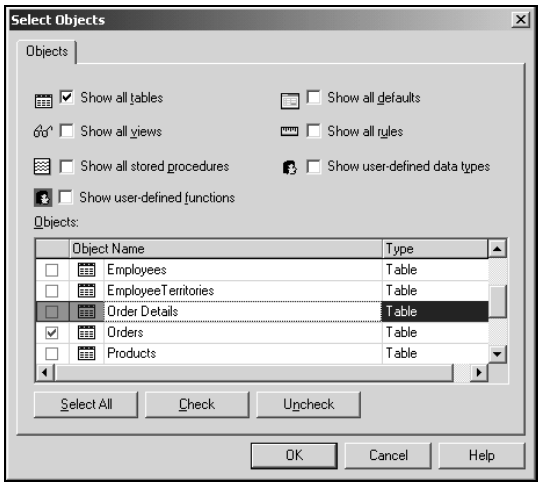
- The final option on the Select Objects to Copy screen is the Script file directory. This option denotes where the scripts are written to on the computer executing the wizard. These scripts are run on the destination server to produce and transfer the objects.

  You have the option to generate a script that will automatically create the SQL Server objects in the destination database, or you can just transfer the data. By checking Include All Dependent Objects, DTS will transfer all the tables that a view refers to.

Transferring SQL Objects Continued

For our example, we will need to ensure that referential integrity is kept by transferring all dependent objects. To do this, check the Include All Dependent Objects option.

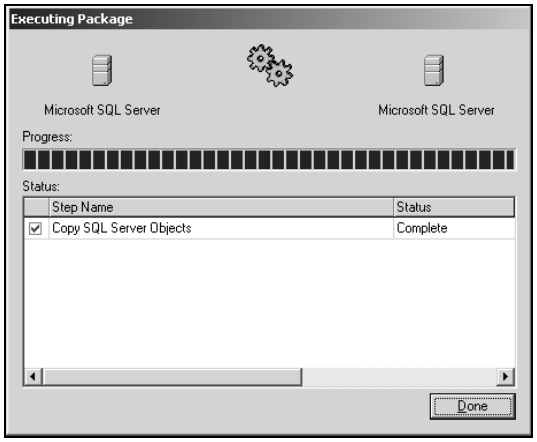
Next, uncheck Copy all objects, and click on Select Objects. You'll be presented with a list of objects in the database to transfer. In this example, we're only concerned with the Orders table and its data, so select just the Orders table:



It is generally a good idea to leave the default setting enabled for this branch of the wizard, so you can ensure that the proper indexes and keys are also transferred. Valid objects that you can transfer include:

- Tables
- Views
- Extended Properties
- Stored Procedures
- Defaults
- Rules
- User-defined data types
- Logins and their object-level permissions

Save this package as Ch1_TransferObject and execute it. After executing the package, view the Wrox database in Enterprise Manager. You will notice that the Orders table has been transferred, as well as its dependents, which are Employees, Shippers, and Customers.



We've seen how we can transfer data between two tables on the same database. What about transferring data between different databases?

The Copy Database Wizard (CDW)

The **Copy Database Wizard** (**CDW**) allows you to copy or move one or more databases, and all their related objects, to another server. CDW can be executed from the source server, destination server, or workstation. Typical applications for using CDW include:

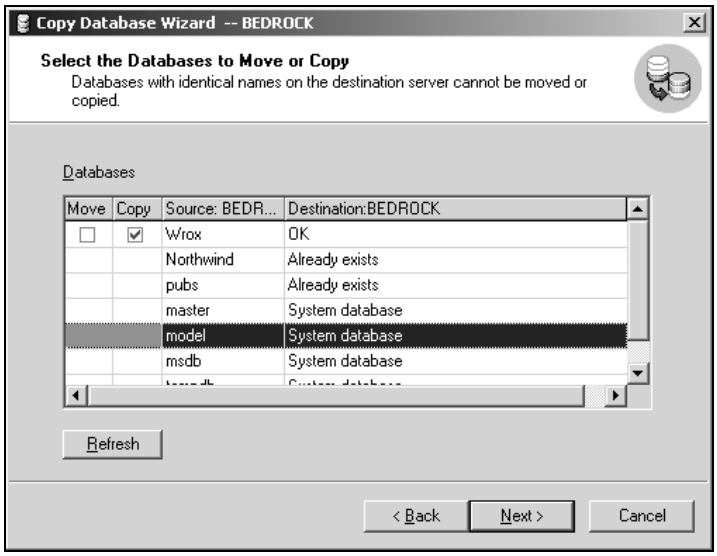
- Merging databases in another SQL Server instance to one instance, to consolidate the licenses
- Moving a database from an Alpha processor machine to an Intel-based server
- Copying a database from development to production
- Cloning a database to many servers
- Upgrade a database from SQL Server 7.0 to 2000

CDW uses DTS on the backend to perform its operations. The database is detached from the source, and re-attached to the new server. Through a number of new tasks added to DTS, CDW can transfer a database and all associated logins, jobs, messages, and history. There are some rules that apply to using CDW:

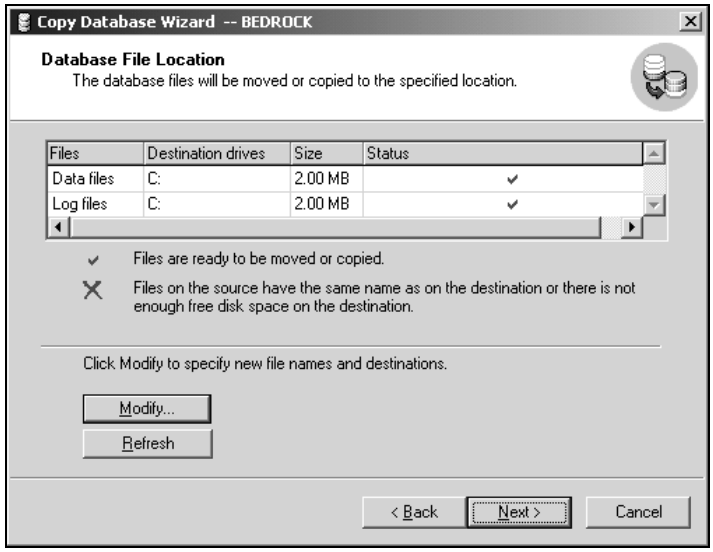
- The destination server must be a SQL Server 2000 server
- The source server can be a SQL Server 7.0 or 2000 machine
- No sessions on the source database can be active
- User using CDW must be a member of the `sysadmin` role and must have NT administrator rights to the system
- There cannot be a database on the destination server with the same name as the source you're trying to copy

Accessing the Copy Database Wizard

To begin the Wizard, go to Tools | Wizards… in Enterprise Manager, and under Management, select the Copy Database Wizard. The wizard will only exist on workstations running the SQL Server 2000 client and is a part of the base SQL Server installation. Specify both your source and destination servers as you did in the Import/Export Wizard. Note that, since the database names can't be the same on the source and the destination, you cannot use the same instance of a server as both the source and the destination. At that point SQL Server will scan the source sever to find out up front which databases can be transferred, as seen above opposite:

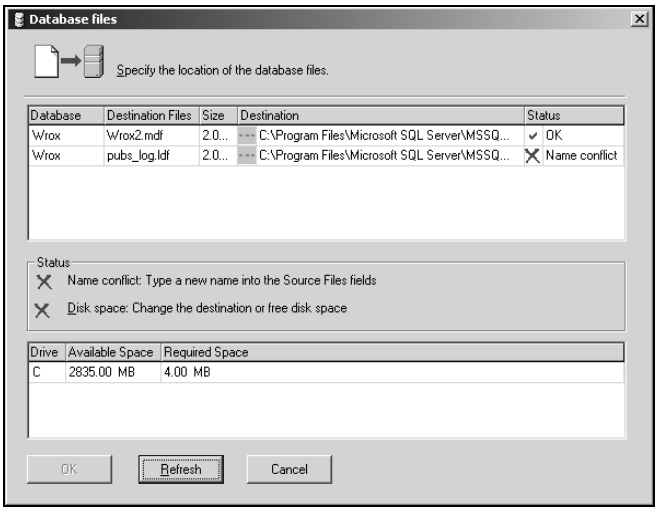


After clicking Next, CDW will scan the directory to make sure there aren't any file name conflicts. Conflicts occur when the file name on the destination is the same on the as on the source. Conflicts are represented with a red X:



Advanced Options

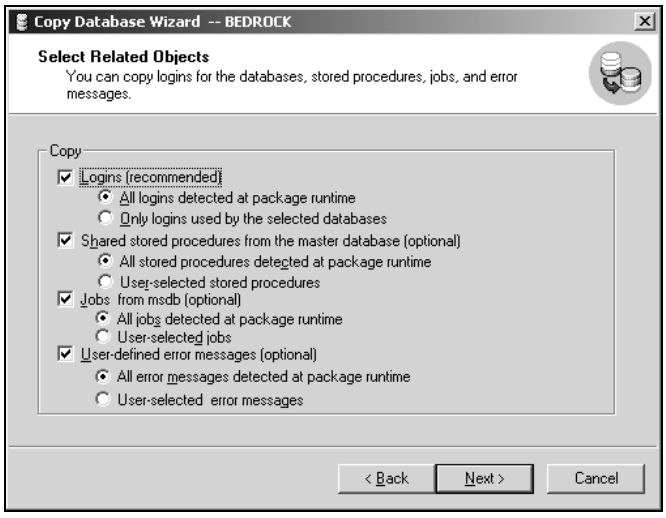
Conflicts can be corrected by clicking on the Modify button. You can also adjust where the files will be placed on the target server. If a conflict occurs, CDW will not allow you to leave the below screen:



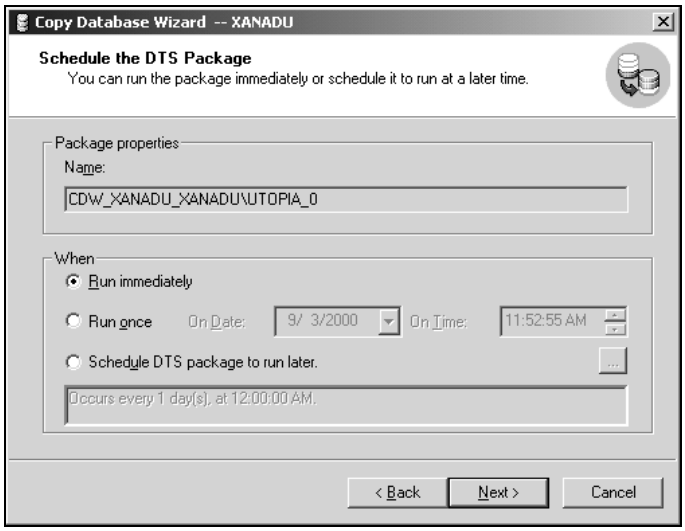
If you're moving a database to a new server, the physical files will not be deleted until you ensure that the database is fine and delete them manually – you have to delete the physical files after you finish the move. If you're moving a database from one instance to another instance on the same SQL Server machine, then the file will be physically moved to the new location, and no manual delete is necessary.

Specifying Items and Executing the Package

The next screen is one of the most important in CDW. This dialog box is where you specify which SQL Server items you'd like to move. By default, every SQL Server login will be copied. If you would like CDW to be more selective about what it moves, select the relevant option. For the purpose of this exercise, just accept the defaults:

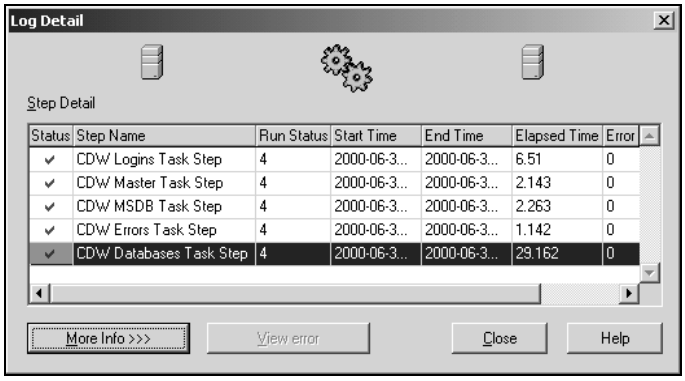


Next, select when you'd like the package to execute, as shown in the screen below. The Name property designates what name the package will be saved as on the destination server. The package is saved in case you receive an error during the copying of the database. If you receive an error, you can execute the package manually later from Enterprise Manager. The most common error is that users may be logged into the source server's database that you're trying to move or copy. You can also schedule the package as you did in the Import/Export Wizard.

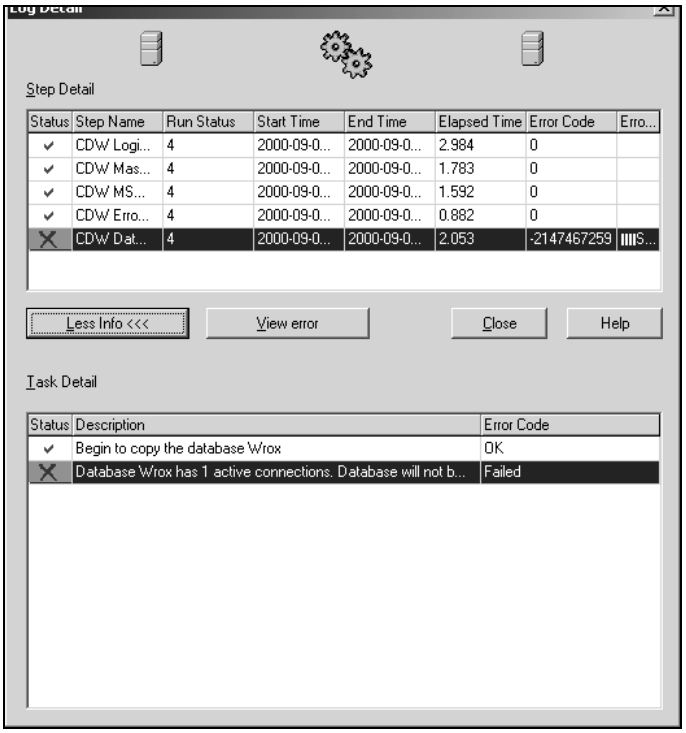


Since all the users must be disconnected from the source database being transferred, when you're doing this in the "real world", you may want to schedule the package to be executed at off-peak hours.

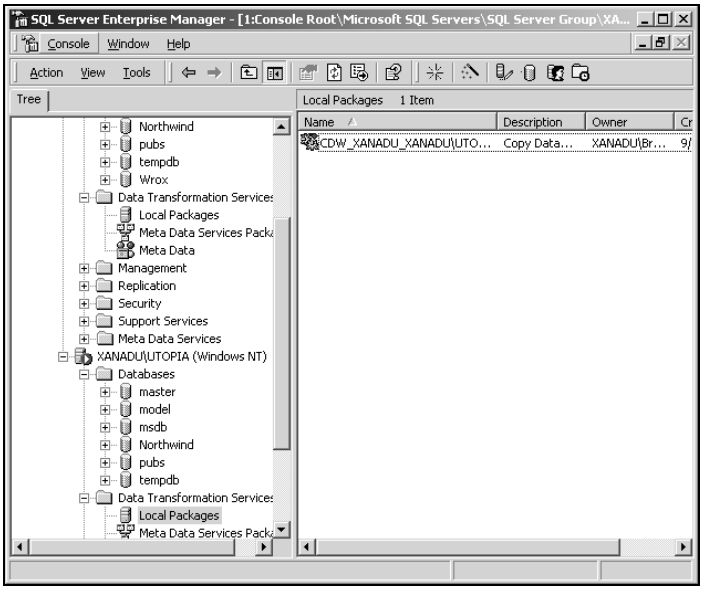
After executing the package, you should see a dialog box that will display the status of the move:



The More Info >>> button will show you the step details. For example, in the following screenshot you can see that we received an error from a user being connected to the source database:



If you select more than one database to move, only one database is moved at a time. Again, once you have validated the move, you can delete the physical files from the source, if they're no longer needed:



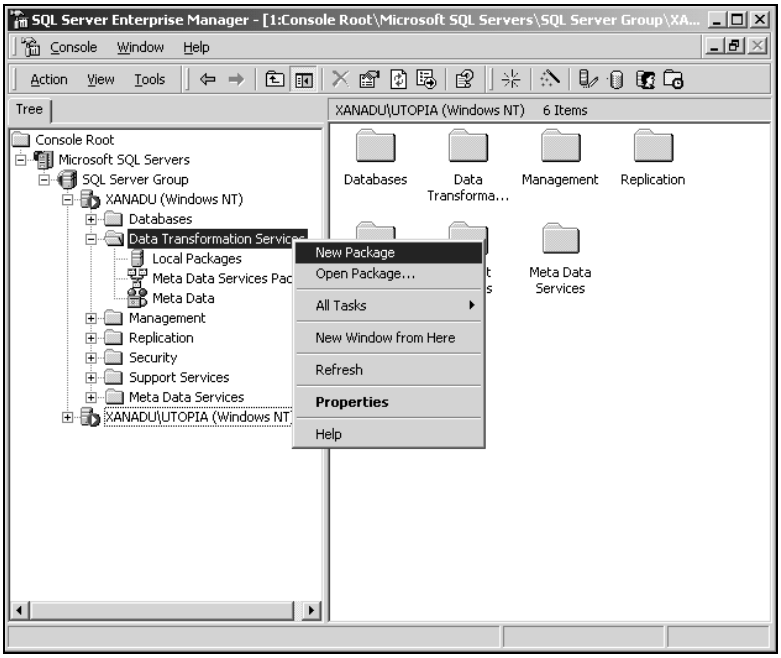
Because CDW uses detach and attach processes, it can move a database much faster than the DTS Import/Export Wizard. CDW also physically creates the database. The advantage to the Import/Export Wizard is that it can handle heterogeneous data and can import into databases that already exist.

The DTS Designer

We said earlier that the DTS Designer is a GUI that allows us to design and execute packages. We will examine it in much more detail in the next chapter, but before we take a preliminary look here, you need to realize that this is a very powerful utility. Not only can you use the DTS Designer to create new packages, but you can also edit existing packages, including those created with the wizards. Let's begin by seeing how we access this Designer.

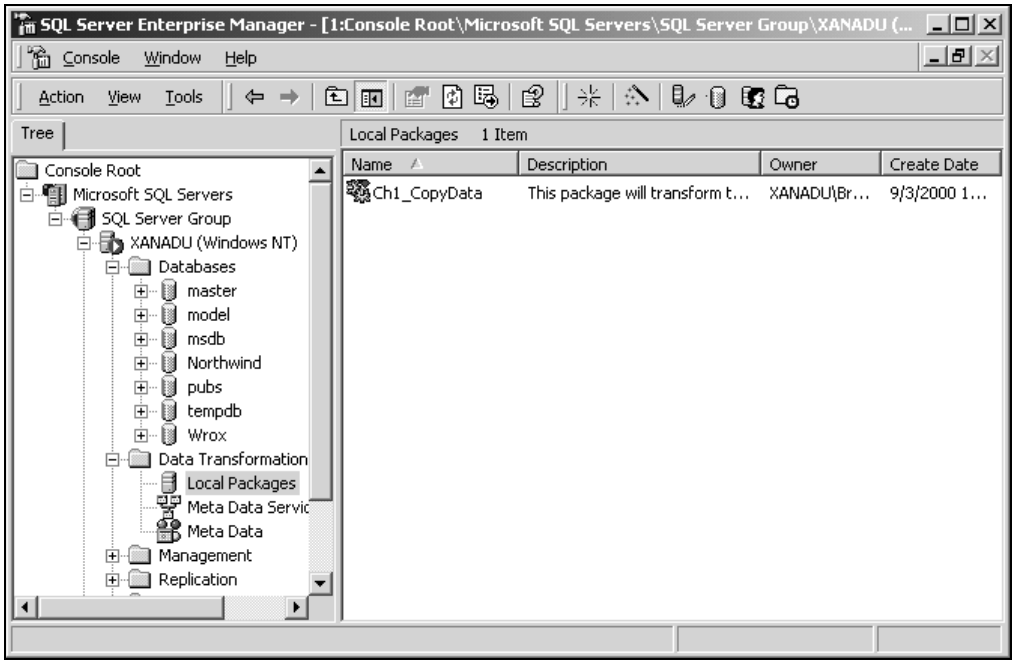
Accessing the DTS Designer

To access the DTS Designer, open SQL Server Enterprise Manager, then click your right mouse button on the Data Transformation Services group and click New Package or you can click Action from the Enterprise Manager menu and select New Package.



Viewing DTS Packages

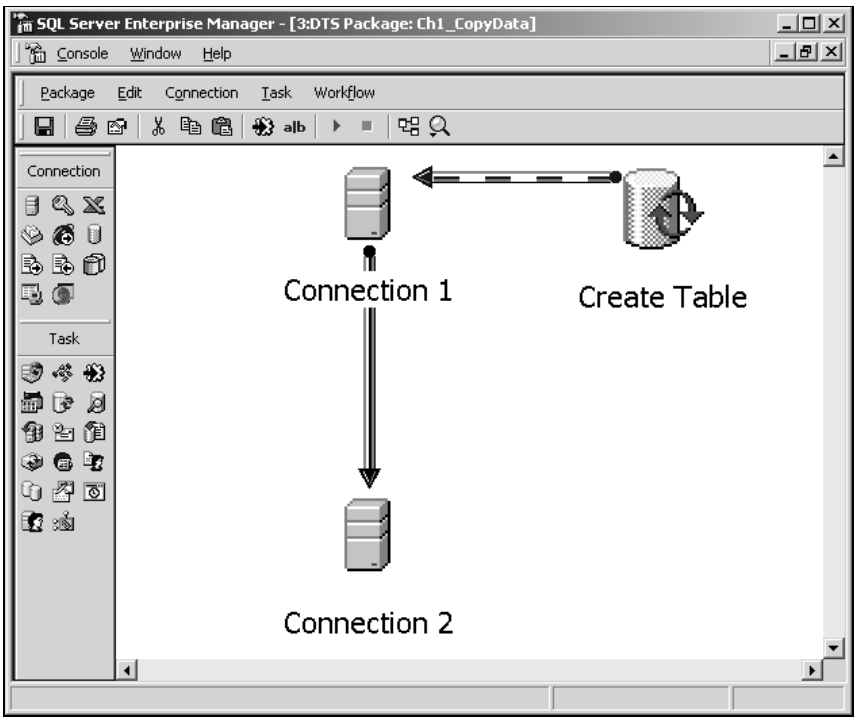
In the examples in this chapter, we've saved our packages locally onto our SQL Server. To view a package after you've saved with the Wizard, select Local Packages under the Data Transformation Services group in Enterprise Manager. In the right pane, you will the packages that you have saved:



We will cover the advantages of saving packages in the various formats in the next chapter. If you have saved a package as a Meta Data Services package, you can open it again by clicking Meta Data Services Packages under the Data Transformation Services group in Enterprise Manager. To open COM-structured file packages, click your right mouse button on the Data Transformation Services group, then select Open Package.

You can execute a package in Enterprise Manager by clicking your right mouse button on the package, then selecting Execute Package.

Double-clicking on any package opens it in the DTS Designer. For example, our Ch1_CopyData package looks like the screenshot above opposite. In the below package, you can see that the first step to the right is to create the table needed, then we transform the data from Connection1 to Connection2 in the second step.



Although the wizards do not offer much flexibility, they are a good building block if you wish to edit them in DTS Designer later. We'll go into more detail about the various options and DTS Designer in the next chapter.

A Note About NT Authentication Security

SQL Servers running Windows 98 will not be able to use Windows Authentication. However, if you're signing into a domain and try to access a SQL Server on the network running Windows NT or 2000, then this option is available. It is important to note that Windows Authentication is the preferred method to access SQL Server since no passwords are passed over the network or saved. A majority of the SQL Server security bugs that Microsoft has announced were based on standard SQL Server security. For example, many SQL Server bugs are announced where SQL Server saves the `sa` password as clear text in the registry.

Summary

DTS is a collection of objects and tools that allow you to transform data easily, and is an integral part of SQL Server. In the past, you would have had to buy expensive programs to do such a task. DTS is not just for transforming data though. You can develop a workflow system to execute programs or scripts. You can even develop your own custom COM components to plug into DTS.

We've seen that the package is the primary component of DTS and is what you create and execute. Inside the package is a set of objects called tasks, which are a set of instructions, and steps, which tell DTS which order to execute the tasks in.

After our basic introduction to DTS, we moved on to look at the Wizards that DTS provides to help simplify some of the most fundamental tasks. They also provide us with the essentials to create a package. You can create a package in the wizards, then go back into it in Enterprise Manager (after saving the package in the wizard) to add your own logic into the package.

One of the most efficient ways to move data is through the Copy Database Wizard, which will move the database and all related items. The DTS Import/Export Wizard also provides an easy way to convert data quickly from other sources such as Excel or DB2. We have an entire chapter dedicated to heterogeneous data conversion.

There are three ways of transforming our data through the wizard:

- The copying data option is the fastest way to transfer your data but lacks in selectivity.
- Using a query to select data to be transferred allows you to copy select data. In this chapter, we had an example where we filtered the Suppliers table to only return suppliers in the USA.
- The transfer of SQL Server objects allows you to transfer objects like tables, stored procedures, and views.

In the next chapter we will discuss more advanced ways of manually creating a package through the DTS Designer.