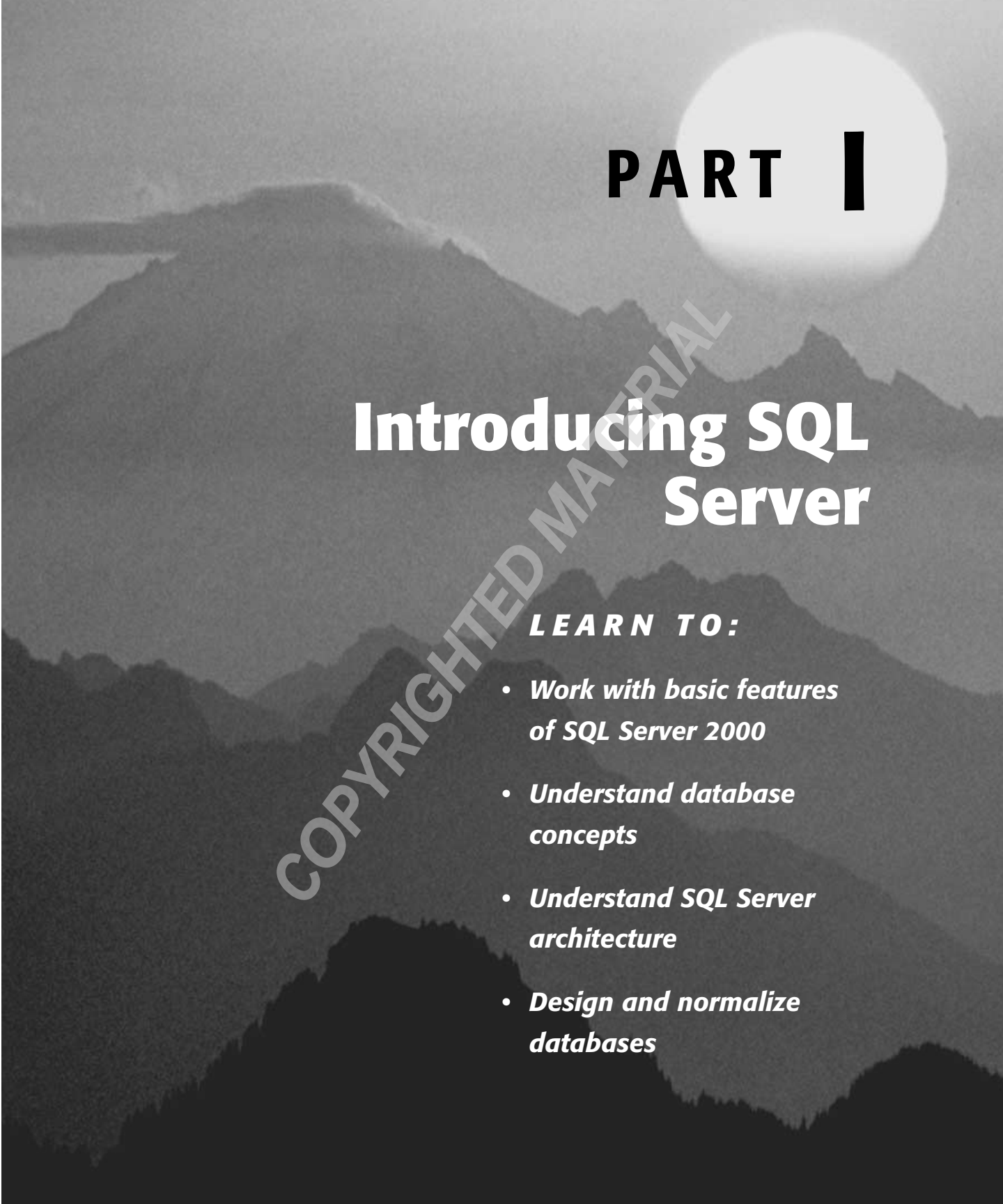


PART I

Introducing SQL Server

LEARN TO:

- ***Work with basic features of SQL Server 2000***
- ***Understand database concepts***
- ***Understand SQL Server architecture***
- ***Design and normalize databases***

CHAPTER 1

Introduction to SQL Server 2000

FEATURING:

Welcome to SQL Server 2000. In this book, we'll help you learn the basics of SQL Server and advance to more complex skills. You won't learn *everything* about Microsoft's flagship database here: It's a huge set of programs that can take years to learn fully. However, we will show you how to get up and running quickly, and how to handle the everyday tasks of keeping your data safe, secure, and available to your users.

Before we dig into the details of SQL Server, we want to introduce you to the product. You might be a budding database administrator (DBA), anxious to manage a database for others to use; you might be a developer, ready to write code that will extract information from a server that someone else is maintaining; or you might be a regular user who just needs to see some data and doesn't have time to wait for the IS department to build an application.

Whoever you are, *Mastering SQL Server 2000* has something for you. In this chapter, we'll give you three quick tours, one each for DBAs, developers, and users. We can't highlight all the features of SQL Server 2000 in these quick tours, but we can show you enough to make you as excited about this database management system as we are.

Tour for DBAs

For DBAs, the main tool will be the SQL Server Enterprise Manager. So we'll start the tour by showing you how to use this interface to manage both users and data, as well as to keep track of what's happening on your server.

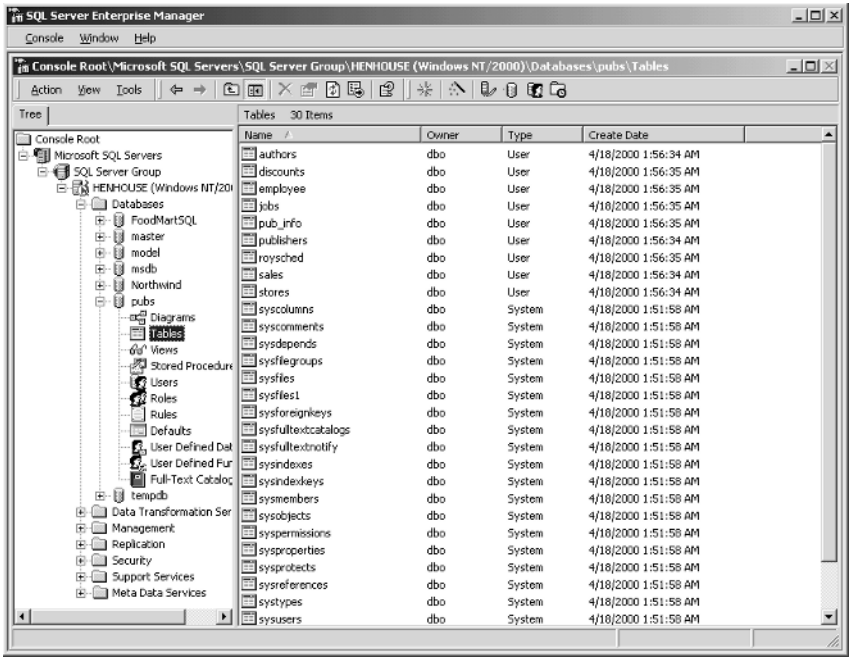
Opening Enterprise Manager

To launch SQL Server Enterprise Manager, choose Programs ➢ Microsoft SQL Server ➢ Enterprise Manager from the Windows Start menu. This will open an instance of the Microsoft Management Console (MMC), with the SQL Server Enterprise Manager loaded as the console root. From here, you can expand a treeview to drill down from servers to databases to objects, and inspect individual objects in a listview. Figure 1.1 shows how Enterprise Manager might look after you drill down a few levels. In this particular case, you'll examine the tables in the pubs database on a server named HENHOUSE in the default server group.

NOTE The pubs database comes with SQL Server 2000. In many cases throughout the book, we'll use pubs as a good generic example of a database. We'll also use the Northwind sample database, or create examples that you can emulate for your own database needs.

FIGURE 1.1

SQL Server Enterprise Manager



TIP Microsoft Management Console is the single interface that will be used to manage most Microsoft BackOffice software in the future. Windows 2000 makes extensive use of MMC for administrative tasks of all sorts. You'll learn a lot more about MMC and Enterprise Manager in Chapter 9.

Even if you don't know anything about SQL Server Enterprise Manager, you'll appreciate the wide list of objects that can be manipulated using this interface:

- Databases
- Database Diagrams
- Tables
- Views
- Stored procedures
- Users
- Roles
- Rules
- Defaults
- User-defined datatypes
- User-defined functions
- Full-text catalogs
- Data Transformation Services Packages
- Meta Data Services Packages
- Data Transformation Services Meta Data
- Alerts
- Operators
- Jobs
- Backups
- Process information
- Database maintenance plans
- SQL Server logs
- Replication
- Logins
- Server roles
- Performance Analyzer
- Web publishing
- Linked servers
- Remote servers

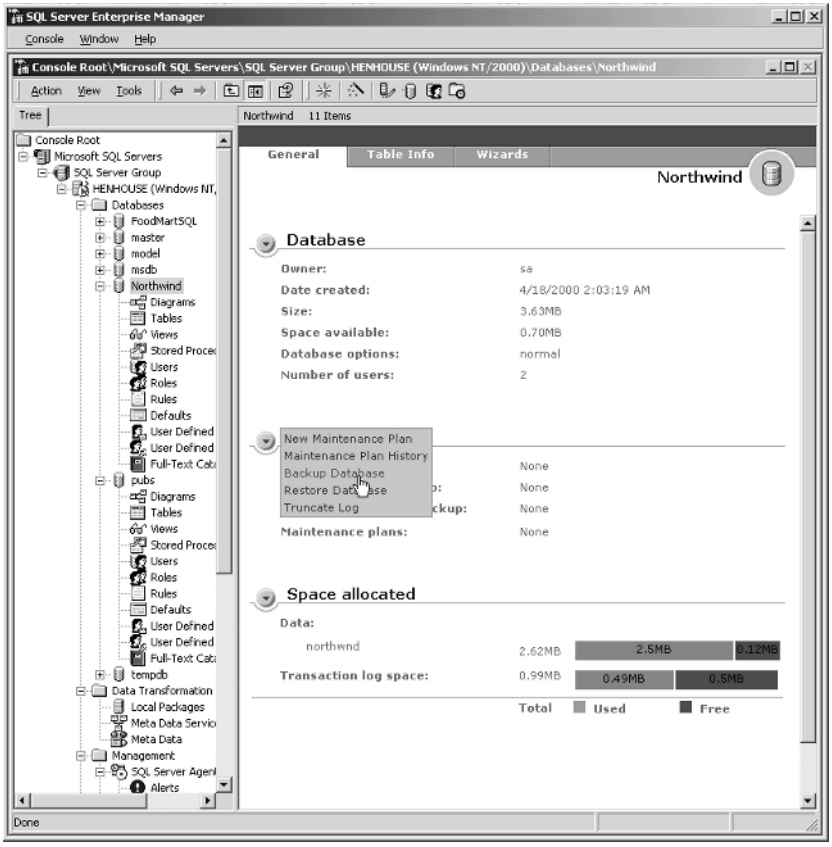
And that's just a sample! You'll learn about most of these objects in coming chapters.

MMC (and therefore Enterprise Manager) can also display fully functional HTML pages. These are sometimes called taskpads in MMC lingo. For example, Figure 1.2 shows the General page that Enterprise Manager automatically generates for the Northwind database, in this case on the server HENHOUSE. This page both shows information about the database and allows the DBA to start common tasks (the tasks listed, such as Backup Database, are hyperlinks that start the listed task when they're clicked).

NOTE Northwind is another database that comes with SQL Server 2000 (like the pubs database). You can use either of these to work through the book, or you can make your own.

FIGURE 1.2

A database taskpad in Enterprise Manager



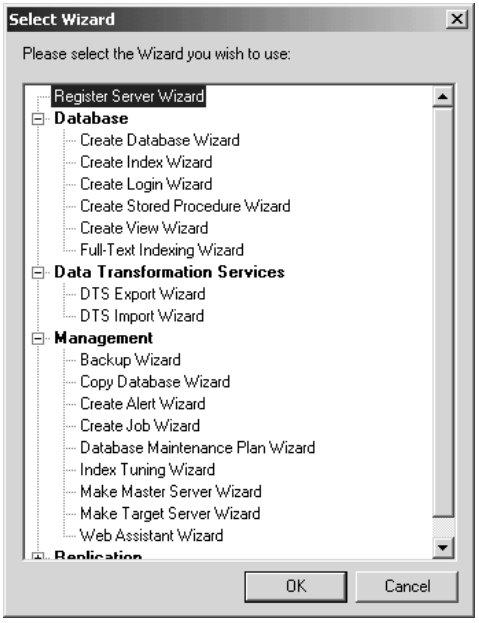
Creating a Login

One of your main tasks as a DBA will be to manage security on your SQL Server. We'll discuss security in much more detail in Chapter 16, but for now, let's look at one part of the picture: creating a login. A SQL Server login is a necessary part of making your SQL Server data available to a Windows NT user on your network.

TIP This is a good place to mention that SQL Server is, by default, not secure, because every installation contains a login named sa (system administrator) that can do anything with the server—and, by default, this user has no password. If you want your data to be secure, you should quickly set a password for the sa login. There are also many other items to manage as you secure your data. For a critical installation, you should consider using a professional tool such as ISS's Database Scanner (`http://www.iss.net`) to audit your server.

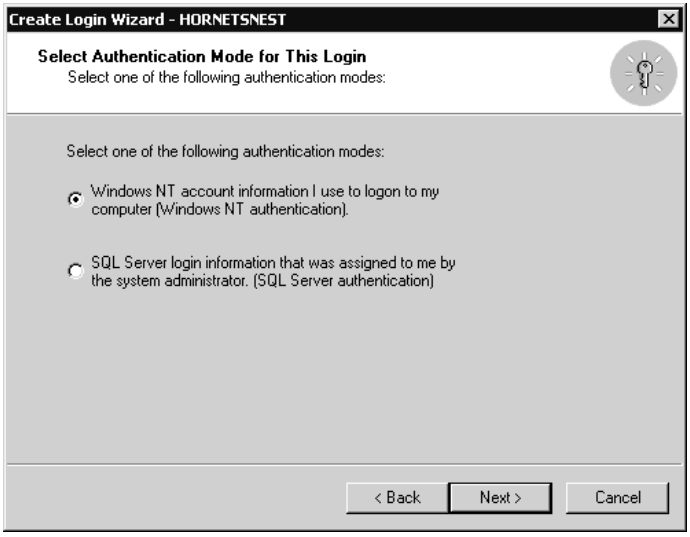
There are several ways to create a new login. The easiest way is to use the Create Login Wizard. Choose Tools ➢ Wizards from the Enterprise Manager menu to open the Select Wizard dialog box, as shown in Figure 1.3. As you can see, Enterprise Manager supplies a variety of Wizards to help you through common management tasks. This is one of the most useful features Enterprise Manager offers the new DBA.

FIGURE 1.3
Choosing a Wizard



Select Create Login Wizard from the list and click OK, or double-click the entry in the list, to launch the Create Login Wizard. After an introductory panel, the Wizard will ask you to choose an authentication mode, as shown in Figure 1.4.

FIGURE 1.4
Choosing an authentication mode



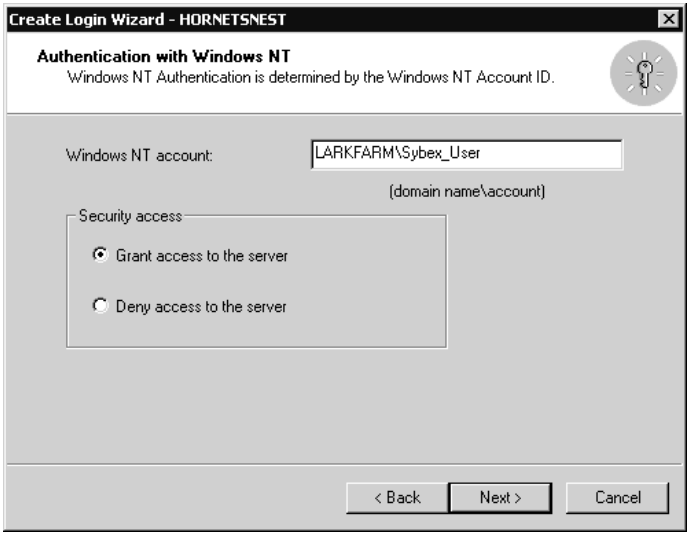
SQL Server can use two different methods to verify that a user is who they claim to be:

- Windows NT Authentication compares the user with their credentials in the Windows NT user database.
- SQL Server Authentication prompts the user for a password that's evaluated by SQL Server itself.

In most cases, you should choose Windows NT Authentication—your users won't have to supply a separate password for SQL Server, and you won't have two sets of passwords to audit and coordinate. You might want SQL Server accounts, though, for operations such as accessing a database over the Internet. Also, you should be aware that Windows NT Authentication is available only if this copy of SQL Server is running on Windows NT or Windows 2000. The option will be unavailable if SQL Server is running on Windows 98.

In the next panel of the Wizard, you'll specify the Windows NT user that you want to create a login for (assuming that you chose Windows NT Authentication mode). Figure 1.5 shows this panel. As you can see, you still need to type in the domain and username manually, rather than browsing through the Windows NT user list.

FIGURE 1.5
Choosing a user

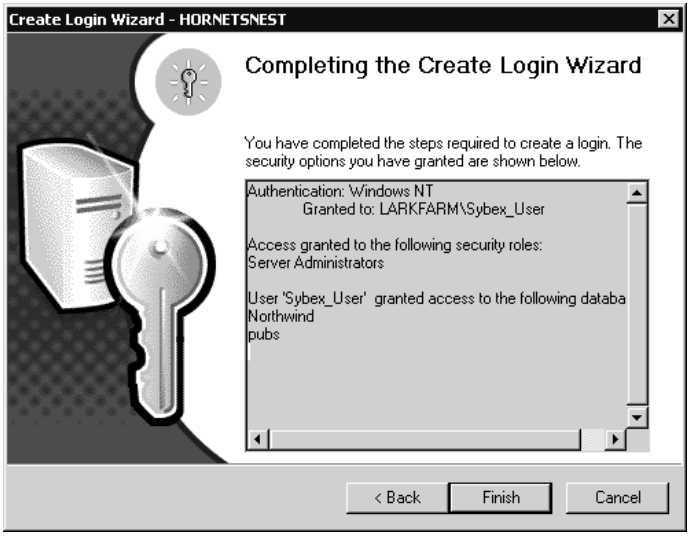


In this panel, you can either grant a user access to your server or deny a user all access to your server. As a general rule, you should deny access to everyone who doesn't explicitly need to get to the data on your server. There's no point in having idle hands riffling through your database.

The next panel of the Wizard allows you to select security roles for this user. A *security role* is a set of permissions designed for particular tasks. For example, SQL Server comes with security roles for System Administrators and Database Creators. After you choose security roles, if any, the Wizard prompts you to choose databases to which the login should have access. If you don't choose any databases here, the user can log in, but can't do anything.

The final panel of the Wizard, shown in Figure 1.6, confirms the choices that you made on the previous panels. If all is well, you just click Finish to create the login. That's all there is to it!

FIGURE 1.6
The final screen of the Create Login Wizard



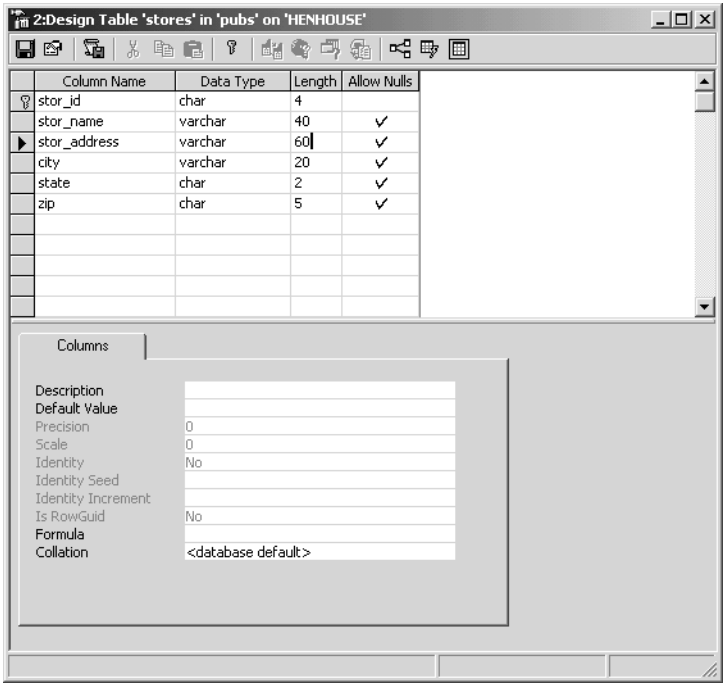
Making a Change to a Table

Another task you may be called on to perform as a DBA is the redesign of a table. Suppose, for example, you're discovering that one of your table's fields won't hold enough information, and you need to increase its size from 40 to 60 characters. In older versions of SQL Server, this would have been quite a chore. But now, a group of utilities known as the Visual Database Tools is integrated with Enterprise Manager. These tools make such tasks simple.

To make this change, first locate the table in question (we'll use the Stores table in the pubs database as an example) in the Enterprise Manager window. Then right-click the table and choose Design Table. This will open a separate table design window with the Enterprise Manager workspace. To change the size of a field, just highlight the old size and type the new size, as shown in Figure 1.7.

FIGURE 1.7

Changing the maximum size for a field



The table designer lets you make a variety of changes to existing tables. For example, you can make the following changes to a table:

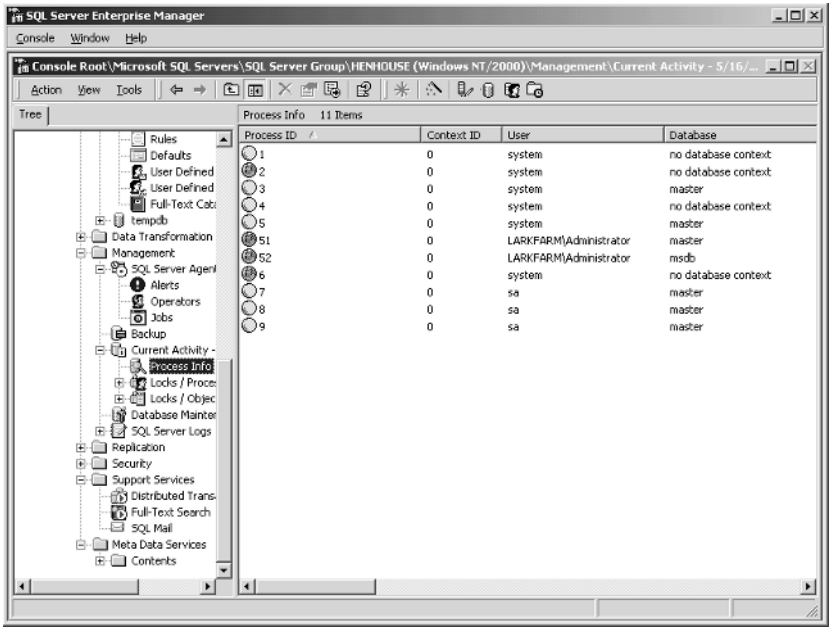
- Adding and deleting fields
- Renaming fields
- Changing datatypes or data sizes
- Making a field nullable
- Adding a default value to a field
- Establishing the primary key

In all of these cases, substantial work is needed to make such a change, usually involving creating a temporary table, copying data to that table, deleting the original table, and renaming the temporary table. However, the table designer takes care of all these details. All you have to do is make the change you want on the table design interface and click the Save button on its toolbar.

Viewing Current Activity

At times, you may want to know what's going on in your database. You can get a quick overview through Enterprise Manager by selecting the Process Info node under Current Activity in the Management section of the treeview. Figure 1.8 shows typical activity on a lightly loaded server.

FIGURE 1.8
Viewing Current Activity



You might find a process running here that you don't recognize. If so, double-clicking the process will let you see the last set of T-SQL commands that were submitted by that particular process. If you're still in the dark, you can send a message from Enterprise Manager directly to the user or computer from where the process originated.

Other nodes within Enterprise Manager allow you to easily view current locks and detect deadlock situations that may be harming performance.

Tracing Activity

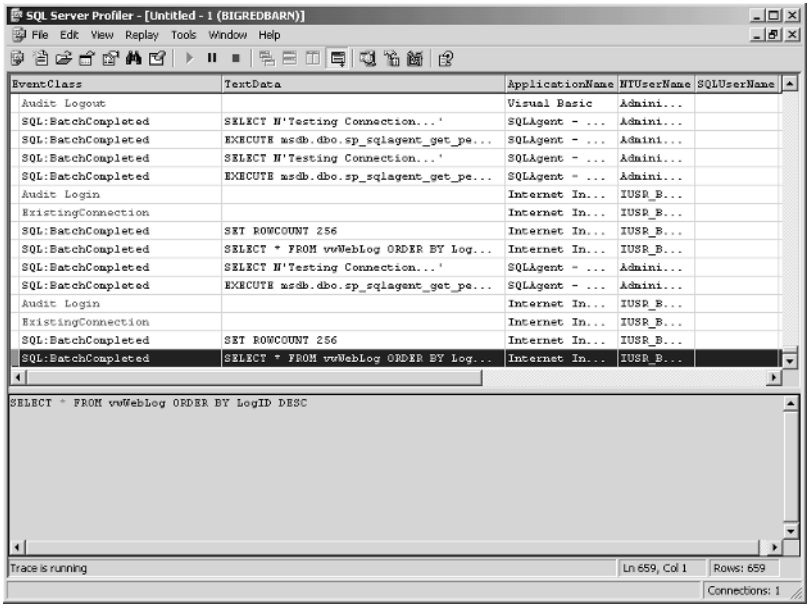
Alternatively, you might require more detail about a particular activity. SQL Server comes with a flexible utility for monitoring the activity on your server—SQL Server

Profiler, which you can launch by choosing Programs ➢ Microsoft SQL Server ➢ Profiler from the Start menu.

SQL Server Profiler acts by intercepting traffic to and from your server and recording it. You can display the results on-screen or save them to a file. You define what you'd like to see by creating a trace. Figure 1.9 shows a trace in action.

FIGURE 1.9

Tracing SQL Server activity



As you can see, a trace shows you not only what SQL statements are being executed on your server, but which applications are sending those statements and how much load they're placing on the server.

Obviously, on a busy server, a trace of all activity would quickly become overwhelming. That's why SQL Server Profiler lets you define traces with very specific purposes. You can, for example:

- Trace only table scans
- Trace only DTC transactions
- Trace only statements taking over 1 second of CPU time
- Trace only statements from a particular user

You'll learn more about SQL Server Profiler in Chapter 24.

Optimizing an Index

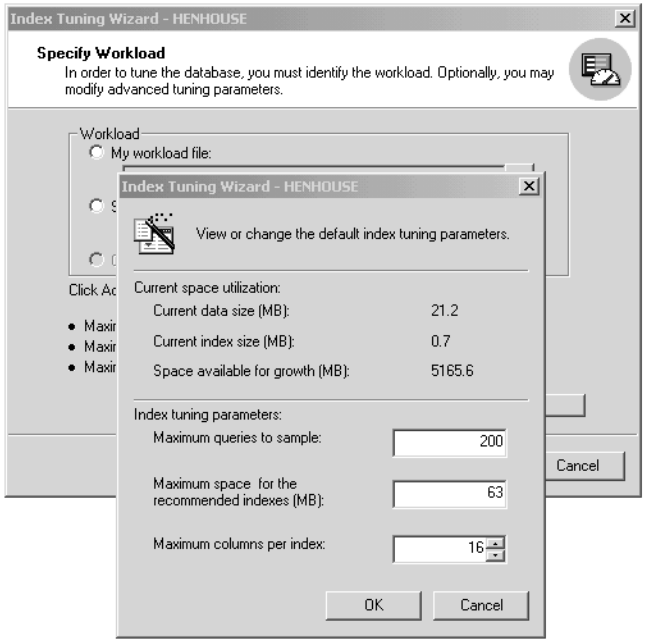
Another task that was once much harder to perform than it is now is optimizing the indexes in a database. When users query the database for specific data, SQL Server develops a query execution plan. Part of the plan specifies which indexes on the data should be used to find the data. If you define too few indexes, it can take longer than it should to find data. If you define too many indexes, it can take longer than it should to insert or update data.

At one time, optimizing indexes was a highly specialized trade that required a good deal of experience. That's changed, now that SQL Server 2000 has distilled that knowledge into the Index Tuning Wizard. Launched from the Select Wizard dialog box, this Wizard will walk you through the process of optimizing the indexes on a particular server and database.

Figure 1.10 shows the Index Tuning Wizard in action. The key advance in this Wizard is the dependency on a workload. A *workload* is a file or table saved by SQL Server Profiler during the course of normal use of your database. This is important because choosing the proper indexes depends on knowing the pattern of queries executed in the database.

FIGURE 1.10

The Index Tuning Wizard



This Wizard gives you a sense of how the different tools supplied by SQL Server all fit together. You can use SQL Server Profiler to track the activity in a database, then use that tracked activity in the Index Tuning Wizard to optimize the database. Afterward, you should be able to see the improvement in a new SQL Server Profiler trace.

Tour for Developers

As a SQL Server developer, you'll be less interested in the design and maintenance of your database than in what you can do with it. SQL Server 2000 ships with a variety of tools for developers, including ActiveX Data Objects (ADO), SQL-DMO, SQL-NS, Analysis Services, and BCP (Bulk Copy Program). You'll learn about many of these in Part 5 of this book. In this tour section, we'll give you a taste of how easy it is to use data stored in a SQL Server database by employing ADO from a Visual Basic program.

A Few Words about ADO

After a few false starts, Microsoft has started using a strategy called Microsoft Universal Data Access across all of their products. Two major components are involved:

- OLE DB is a set of COM interfaces for data access. OLE DB providers can connect to data from a variety of sources.
- ADO is an object library that exposes OLE DB data to consumers.

SQL Server 2000 ships with ADO 2.6. This is a new release that was first shipped as a part of Windows 2000. Fortunately, all versions of ADO have been backward-compatible (that is, what worked in a previous version still works in the new version). So the great mass of ADO code already out there should work just fine after installing SQL Server 2000.

The basic idea of ADO is to make access to all sources of data work the same from the developer's point of view. Whether you're dealing with SQL Server, Jet databases, or even information stored in an Exchange mailbox, your code is pretty much the same. In this section, we'll take a quick look at some of that code.

TIP For current information on ADO and other parts of the Universal Data Access strategy, it pays to monitor `http://www.microsoft.com/data`.

Creating an ADO Connection

Before you can do anything using ADO, you need to get hooked up to a data source. You do this by creating a Connection object and then using this object's Open method to specify the data source with which you'd like to work.

TIP To use any of the ADO objects from Visual Basic, you need to first use the Project ➢ References dialog box to set a reference to the Microsoft ActiveX Data Objects 2.6 Library.

Here's an example of creating a connection:

```
Private Sub cmdConnect_Click()
  ' Connect to a SQL Server database
  Dim cnn As New ADODB.Connection

  cnn.Open "Provider=SQLOLEDB.1; " & _
   "Data Source = (local);" & _
   "User ID = sa;" & _
   "Initial Catalog = Northwind"

  MsgBox cnn.ConnectionString
End Sub
```

Note that the Connection object is declared using the library name (ADODB) as well as the object name. This is good programming practice because it helps protect you from the possibility of getting the wrong object if you happen to reference two different libraries, each of which supplies a Connection object.

The single argument of the Open method is an OLE DB connection string. A *connection string* is simply a list of arguments and values that tells OLE DB where to find data. In this case, there are four arguments:

Provider: Specifies the OLE DB provider to use, and therefore the type of data to retrieve. SQLOLEDB.1 is the name of the Microsoft OLE DB Provider for SQL Server and is the fastest way to fetch SQL Server data via ADO.

Data Source: Specifies the SQL Server to connect to. In this case, we've used the special string "(local)" to indicate that the server is running on the same computer as the Visual Basic application. Alternatively, you could specify the name of a server here.

User ID: Specifies the username to be used when logging in to the SQL Server. In this case, we're using the default sa user with the default blank password. If you need to send a password, you can use the Password argument, but that wasn't needed in this case.

Initial Catalog: Specifies the name of the database to connect to.

If you type in and run this code, you'll find that the connection string returned is somewhat longer than the one you supplied to the Open method. It will be something like:

```
Provider=SQLOLEDB.1;User ID=sa; ➥
Initial Catalog=Northwind;Data Source=(local); ➥
Use Procedure for Prepare=1;Auto Translate=True; ➥
Packet Size=4096;Workstation ID=MOOCOW
```

The SQL Server OLE DB Provider has inserted default values for other arguments that it understands. For example, the Connect Timeout argument specifies how many seconds to wait for a response from the server.

NOTE You'll find more information on the Open method, as well as the rest of ADO, in Chapter 17.

Retrieving Data

To retrieve data using ADO, you use a Recordset object. A recordset represents, sensibly enough, a set of records (rows) out of a table or a group of tables joined together. Each recordset is associated with a connection and a record source that defines the desired set of records. A record source can be one of the following:

- A table name
- The name of a stored procedure that returns records
- A SQL statement
- The name of a file containing saved records

These are just some of the possibilities; ADO 2.6 even includes the ability to fetch a recordset from a URL.

Here's a bit of code that fetches a set of records and prints it out to the Immediate Window in Visual Basic:

```
Private Sub cmdRecordset_Click()
  ' Print a recordset to the Immediate Window
```

```
  Dim cnn As New ADODB.Connection
  Dim rst As New ADODB.Recordset
  Dim fld As ADODB.Field

  ' Open a connection
  cnn.Open "Provider=SQLOLEDB.1; " & _
   "Data Source = (local);" & _
   "User ID = sa;" & _
   "Initial Catalog = Northwind"

  ' Open a recordset on a table
  rst.CursorLocation = adUseClient
  rst.Open "Shippers", cnn, adOpenStatic, _
   adLockOptimistic

  ' Print the names of the fields
  For Each fld In rst.Fields
    Debug.Print fld.Name & " ";
  Next fld
  Debug.Print

  ' Print the contents of each field
  Do Until rst.EOF
    For Each fld In rst.Fields
      Debug.Print fld.Value & " ";
    Next fld
    Debug.Print
    rst.MoveNext
  Loop

  ' Tidy up
  rst.Close
  cnn.Close
  Set rst = Nothing
  Set cnn = Nothing
  Set fld = Nothing
End Sub
```

There are a few things to note about this code that will give you a sense of the flexibility of the ADO Recordset object:

- Setting the CursorLocation property of the recordset to adUseClient tells ADO to cache the records locally after they're retrieved. This allows more efficient processing and enables some advanced ADO methods.
- You can specify that you want a static recordset (one that doesn't reflect changes from other users) as well as the type of locking (in this case, optimistic locking) to use when you open the recordset.
- A recordset is a collection of fields, each with a name and value.
- The recordset supports a number of properties, including an EOF property that's true at the end of the recordset, and a number of methods, including a MoveNext method that moves the cursor to the next record.
- To be neat, you can close your ADO objects and set them equal to Nothing to explicitly free the memory they're consuming.

Figure 1.11 shows the results of running this procedure.

FIGURE 1.11

Data from a Recordset object

Editing Data

ADO also makes it simple to edit data: You can add new records, delete existing records, or modify the data stored in existing records by calling appropriate methods of a recordset. For example, here's some code to modify the recordset we just created:

```
Private Sub cmdModify_Click()
  ' Demonstrate recordset modification
  Dim cnn As New ADODB.Connection
  Dim rst As New ADODB.Recordset

  ' Open a connection
  cnn.Open "Provider=SQLOLEDB.1; " & _
   "Data Source = (local);" & _
   "User ID = sa;" & _
```

```
    "Initial Catalog = Northwind"

  ' Open a recordset on a table
  rst.CursorLocation = adUseClient
  rst.Open "Shippers", cnn, adOpenStatic, _
   adLockOptimistic

  ' Add a record
  rst.AddNew
  rst.Fields("CompanyName") = "New Shipper"
  rst.Fields("Phone") = "(509)-555-1212"
  rst.Update

  ' Modify the record just added
  rst.MoveLast
  rst("Phone") = "509-666-1212"
  rst.Update

  ' Delete the record we've been playing with
  rst.MoveLast
  rst.Delete

  ' Tidy up
  rst.Close
  cnn.Close
  Set rst = Nothing
  Set cnn = Nothing

End Sub
```

You can see the recordset methods in action here:

- The AddNew method prepares a new row to be added to the recordset.
- The Update method saves a new row or changes to data in an existing row.
- The Delete method deletes the current row.

NOTE In this case, you're assured that the new row will be the last row in the recordset because the recordset is based on a table that includes an Identity field. The server automatically assigns ID numbers in increasing order to this field. You'll learn about Identity fields in Chapter 11.

Displaying Data on a Web Page

These days, the Internet is everywhere—and where there's no Internet, there are corporate intranets. It's probably inevitable that any developer today will be asked to make data available via a Web page.

There are many ways to do this, of course. You can run client-side VBScript code that connects to a remote server. You can create ASP pages that use ADO objects directly on a server to create raw HTML to send to clients. You can also write queries that directly return XML, which some browsers can display. For now, let's look at the simplest possible case: using the tools that SQL Server supplies to publish data directly to a Web page.

From Enterprise Manager, you can choose Tools ➢ Wizards and launch the Web Assistant Wizard. (How can something be both an assistant and a Wizard? We don't know; we didn't name it.) This Wizard creates a set of SQL Server tasks that create and update a Web page based on the data you choose.

Using the Wizard is a simple process:

1. Choose the database that holds the data that you wish to publish.
2. Assign a name to the Web page and choose a table, SQL statement, or stored procedure to supply the data.
3. Select the columns from your data that you wish to publish.
4. Decide which rows to publish.
5. Select an update frequency. As Figure 1.12 shows, this step is very flexible.
6. Choose a filename for the Web page.
7. Supply information on formatting your Web page.
8. Select a list of hyperlinks for the page to include.
9. Decide whether to return all the data or chunks of data.

When you're finished making choices, click Finish—the Wizard does the rest. Figure 1.13 shows a Web page generated by the Web Assistant Wizard.

FIGURE 1.12

The Web Assistant Wizard at work

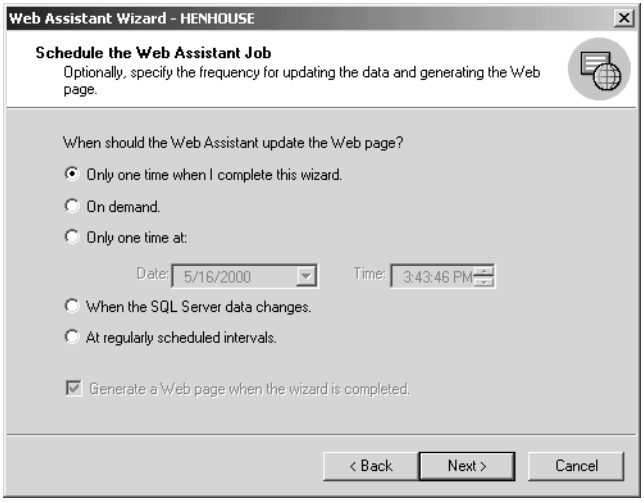
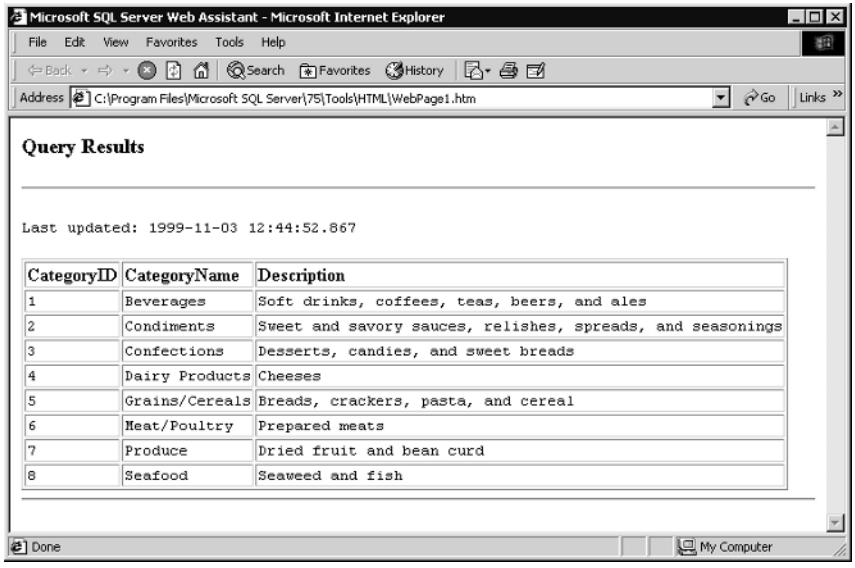


FIGURE 1.13

The finished Web page



NOTE You'll learn more about using SQL Server data with the Internet in Chapters 21 and 22.

Tour for Users

Some of you aren't DBAs or developers, just users of data stored in SQL Server databases. That's OK—there's plenty in the product (and in this book) for you too. In fact, we suspect that, increasingly, more people are going to be combination DBA/developer/users in the future, now that Microsoft has released a desktop version of SQL Server that runs under Windows 95 or Windows 98. In addition to the desktop version, which includes the management tools, there's also the Microsoft Database Engine (MSDE), which is SQL Server without any of the user interface. MSDE is shipped with other Microsoft products such as Microsoft Office or Microsoft Visual Studio.

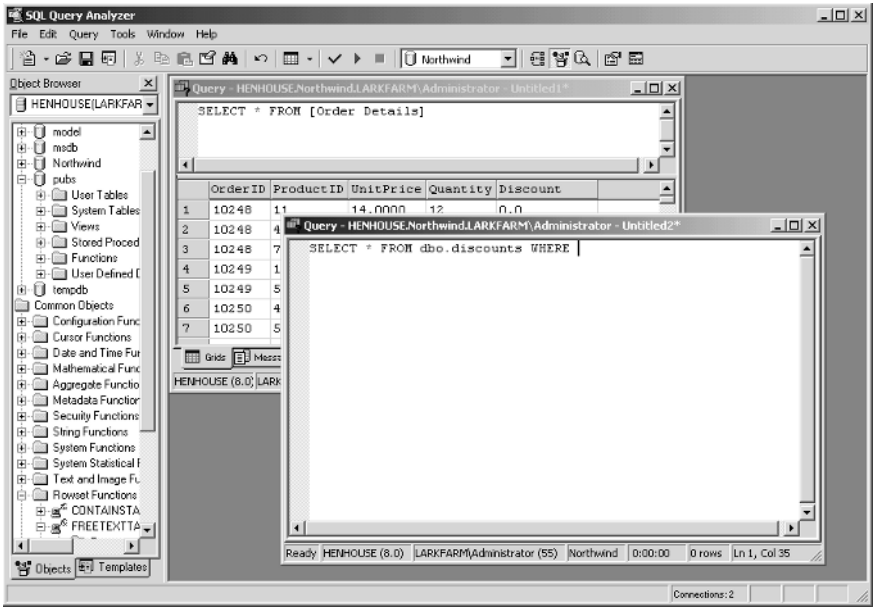
So, in this section, we'll examine the available tools to use when you just want to get to your data. First on the list is Query Analyzer, a tool that ships with SQL Server. However, we also want to highlight Microsoft Access 2000, part of the Office 2000 suite of products, for its easy connectivity to SQL Server data.

Opening Query Analyzer

For ad hoc queries (that is, queries that haven't been saved to a database), the tool that ships with SQL Server 2000 is Query Analyzer. You can launch this tool by choosing Programs ➢ Microsoft SQL Server ➢ Query Analyzer from the Start menu. You can use SQL Server setup to install Query Analyzer on a computer that doesn't have SQL Server itself installed, so that Query Analyzer can be used from anywhere on the network.

When you launch Query Analyzer, you'll be prompted to enter the name of a SQL Server and your authentication information. After that, the program will open with a blank query window. Figure 1.14 shows the basic Query Analyzer interface. In this case, one query was executed, and a new window was opened to execute a second query. The Object Browser, to the right of the Query Analyzer workspace, provides easy access to the names of all your SQL Server objects. As you can see, Query Analyzer can display multiple results at any time.

FIGURE 1.14
Query Analyzer



Query Analyzer can show you the results of any Transact-SQL statement (Transact-SQL, or T-SQL, is the language of SQL Server). For example, you might try executing this statement in the Northwind sample database:

```
SELECT CompanyName, Country
FROM Customers
WHERE CustomerID > 'MMMMM'
ORDER BY Country
```

Even if you don't know SQL, you can probably guess what this statement does. It returns the CompanyName and Country fields from the Customers table for all customers whose CustomerID is greater than (that is, later in the alphabet than) "MMMMM". The results are sorted by the customer's country. Although SQL is a specialized language, by and large, you can read SQL as plain English and get the idea.

NOTE You'll learn a lot more about SQL in Chapters 5 through 8. Appendix A contains a summary of important Transact-SQL statements.

Other Query Analyzer Features

Query Analyzer is a pretty flexible tool. Some of the other actions you can do from this interface include:

- Saving queries to text files and reloading them later
- Viewing results in either a grid or plain text
- Checking the syntax of a query without executing it
- Analyzing the indexes in a database to determine whether a particular query would be helped by additional indexes
- Showing the execution plan for a query

The last point—showing the execution plan for a query—is worth discussing. The *execution plan* for a query is the set of steps that SQL Server will follow to get you the information for which you've asked. For example, in the SELECT query in the previous section, SQL Server will first find all the rows desired using the index on the CustomerID field and then sort them in the desired order. For more complex queries, an execution plan might have dozens of steps.

If you get really familiar with SQL, you can sometimes use optimizer hints in your queries to change the execution plan and make it faster for your particular data.

WARNING Don't change the execution plan if you don't know what you're doing. SQL Server 2000 does a good job of optimizing queries all by itself.

Connecting Access 2000 to SQL Server

Although Query Analyzer is a useful tool, it's not all that user-friendly. You need to understand SQL to do much of anything with Query Analyzer. Wouldn't it be nice to just view your SQL Server data through a more friendly interface? Well, if you're familiar with Microsoft Access and you have Access 2000, you can do just that.

Access 2000 includes a new type of database called an Access project. An Access project includes all of the familiar Access user-interface tools such as forms and reports. However, instead of storing its data in a Jet database, it stores its data in a Microsoft SQL Server database. In fact, Access 2000 even comes with a desktop version of SQL Server, the Microsoft Database Engine (MSDE).

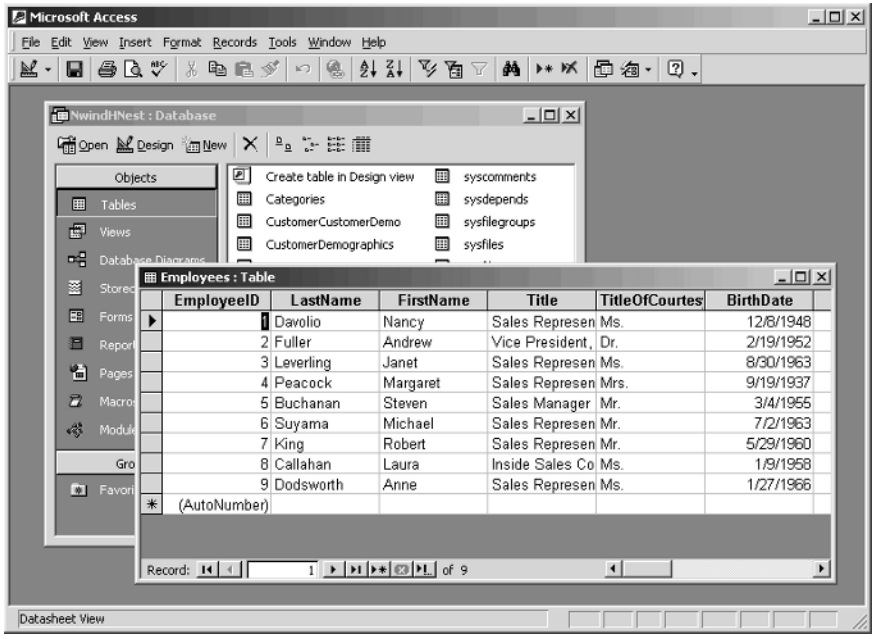
You can also create an Access project that shows data from an existing SQL Server database. To do so, follow these steps:

1. Launch Access 2000.
2. Choose Create a New Database Using Access Database Wizards, Pages and Projects from the opening dialog box.
3. Choose the General tab in the New dialog box.
4. Choose the icon for Project (Existing Database) and click OK.
5. Assign a name to your project and click Create.
6. Enter your SQL Server name, authentication information, and database name in the Data Link Properties dialog box, and click OK.

That's all there is to it. As Figure 1.15 shows, the result of following these steps with the sample Northwind database is the creation of an Access project showing your SQL Server data. In the figure, we've opened up one of the SQL Server tables to show the data.

FIGURE 1.15

An Access 2000 project



Editing Data in Access 2000

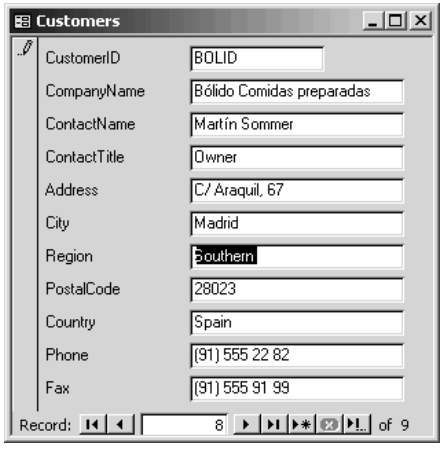
Once you've created an Access project tied to your SQL Server database, all of the Access 2000 tools are available to use. For example, suppose you want to view and edit your Customer data in a friendly format. It's easy to do using the Access Form Wizard:

1. Select a table in the Database Window (for example, Customers).
2. Select Insert ➢ Form from the Access menus.
3. Choose Autoform (Columnar) and click OK.

The result will be a form similar to the one shown in Figure 1.16.

FIGURE 1.16

SQL Server data in an Access form



From this form, you can perform all of the basic data operations:

- Entering new customers
- Editing existing customers
- Deleting customers that you no longer need

WARNING You're still limited by the way your SQL Server is set up. In particular, if your DBA has used SQL Server security to prevent you from modifying data, you won't be able to do so through an Access project. However, if you're your own DBA, working on a single-user version of SQL Server, this shouldn't be a problem.

Similarly, you can use the Access report Wizards to create summaries and lists of your SQL Server data in a format that's easy to print. When you create user-interface objects such as reports in an Access project, the user-interface objects themselves are stored in the .ADP file created by Access. All of the data objects, such as views or tables, remain on the server.

NOTE For much more information about Access projects, see *Access 2000 Developer's Handbook, Volume 2: Enterprise Edition* (by Paul Litwin, Ken Getz, and Mike Gilbert, ISBN 0-7821-2372-4, Sybex 2000).

Summary

SQL Server isn't everything to everybody, but in the current release, it certainly has something for almost every computer user. The range of SQL Server goes from simple customer databases intended for a single user all the way to terabytes (a *terabyte* is one trillion characters) of data in cases such as Microsoft's TerraServer (`http://www.terraserver.microsoft.com`). In the rest of this book, you'll learn about various aspects of SQL Server:

- Part 1 will teach you basic SQL Server and database concepts.
- Part 2 will teach you Transact-SQL.
- Part 3 examines the basic SQL Server objects in more detail.
- Part 4 covers administrative tasks.
- Part 5 reviews the developer tools that ship with SQL Server.
- Part 6 deals with SQL Server data on the Web.
- Part 7 introduces some advanced concepts.

Ready to start? Good! The next chapter will teach you basic database concepts.