Starting and Stopping SQL Server

SQL Server can be started automatically each time the Windows operating system starts or by using SQL Server Management Studio. In the **Object Explorer** window, right-click the selected server and click **Start** in the pull-down menu. The pull-down menu also contains the corresponding **Stop** function that stops the activated server. The additional **Pause** function pauses the whole system, which means that new users are not allowed to log into the system.

Managing Databases

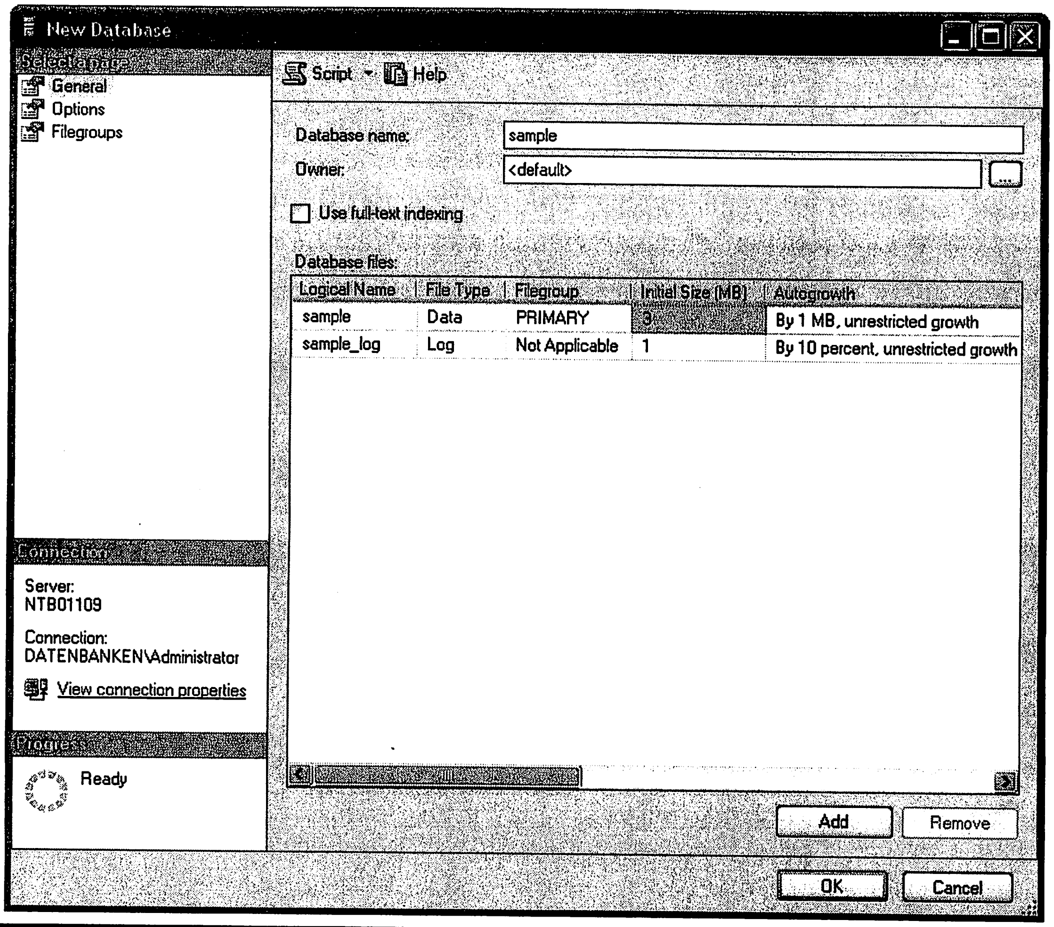
You can create a new database using SQL Server Management Studio and its component Object Explorer or Transact-SQL. (The next section discusses database creation and modification using Object Explorer. Database creation using the Transact-SQL language will be discussed in Chapter 4.)

Managing Databases Using Object Explorer

You use Object Explorer to explore the objects within a server. This component of SQL Server Management Studio can be used after a connection to the server is established. From Object Explorer you can inspect all the objects within a server and manage your server and databases. The Object Explorer tree has the same form as in the previous versions of SQL Server, with one exception: The **Database** folder contains several subfolders, one for the system databases and one for each new database that is created by a user. (System and user databases are discussed in detail in Chapter 3.)

To create a database using Object Explorer, right-click **Databases** and select **New Database**. In the **New Database** dialog box (Figure 2-5), type the name of the new database and click **OK**. (As you can see in Figure 2-5, we use the **New Database** dialog box to create the sample database.) Each database has several different properties, such as file type, initial size, and so on. Database properties can be selected from the left pane of the **New Database** dialog box. There are several different property groups, including the following:

- General
- Files (appears only for an existing database)
- Options
- Filegroups
- Permissions (appears only for an existing database)
- Mirroring (appears only for an existing database)
- Extended Properties



Database name: sample

Owner: <default>

☐ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
sample	Data	PRIMARY	3	By 1 MB, unrestricted growth
sample_log	Log	Not Applicable	1	By 10 percent, unrestricted growth

Server:
NTB01109

Connection:
DATENBANKEN\Administrator

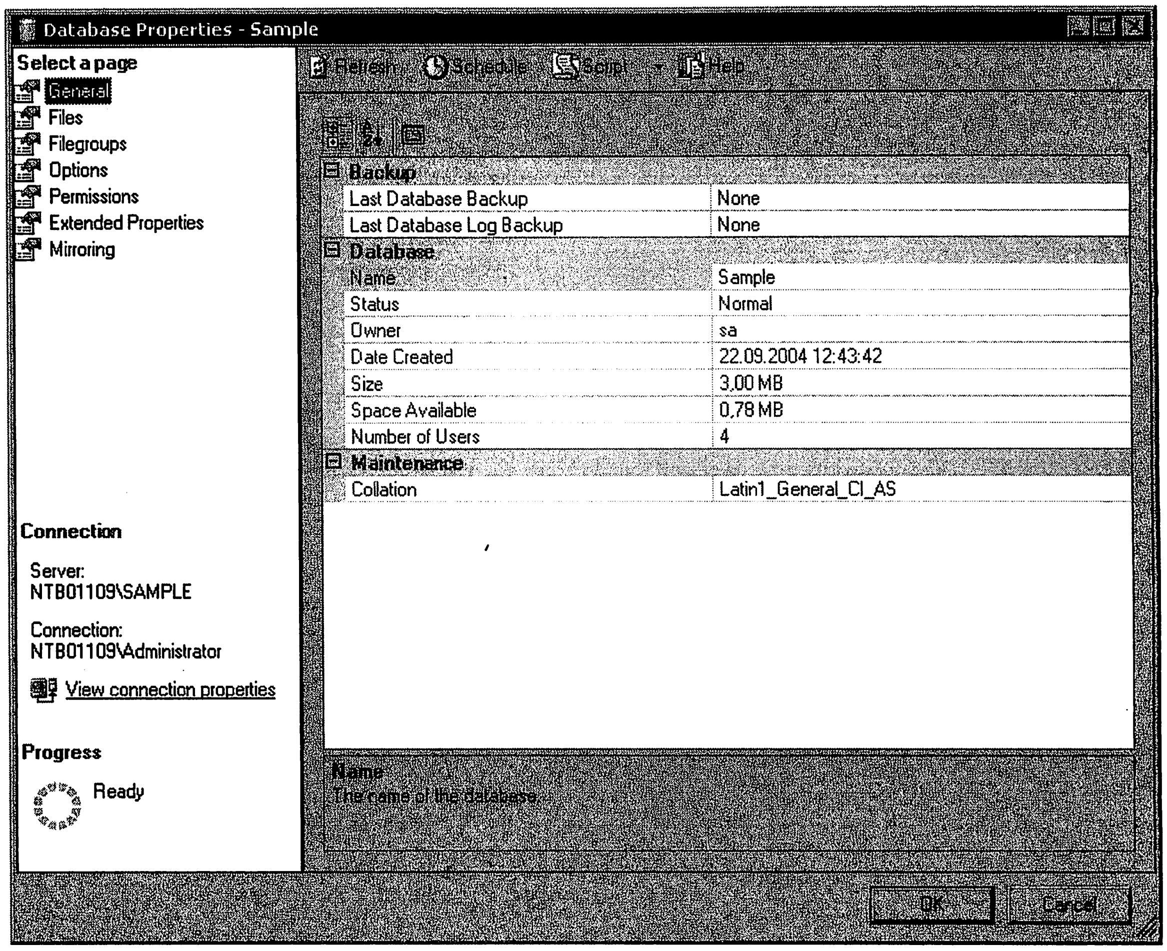
Ready

Figure 2-5 *The New Database dialog box*

General properties of the database (Figure 2-6) include, among others, the database name, the owner of the database, its collation, and size. The properties of the data files that belong to a particular database comprise the name and initial size of the file, where the database will be stored, and the type of the file (PRIMARY, for instance). A database can be stored in multiple files.

NOTE

*SQL Server has dynamic disk space management. This means databases can be set up to automatically expand and shrink as needed. If you want to change the **Autogrowth** property of the **Files** option, click . . . in the **Autogrowth** column and make your changes in the **Change Autogrowth** dialog box. The **Enable Autogrowth** check box should be checked to allow the database to autogrow. Each time there is insufficient space within the file when data is added to the database, the server will request the additional space from the operating system. The amount (in megabytes) of the additional space is set by the number in the **File Growth** frame of the same dialog box. You can also decide whether the file can grow without any restrictions (the default value) or not. If you restrict the file growth, you have to specify the maximum file size (in MB).*



Backup	
Last Database Backup	None
Last Database Log Backup	None
Database	
Name	Sample
Status	Normal
Owner	sa
Date Created	22.09.2004 12:43:42
Size	3,00 MB
Space Available	0,78 MB
Number of Users	4
Maintenance	
Collation	Latin1_General_CI_AS

Figure 2-6 *The Database Properties dialog box: the General page*

The **Filegroups** properties of a database contain name(s) of the filegroup(s) to which the database file belongs, the art of the filegroup (default or nondefault), and the allowed operation on the filegroup (read/write or read only).

All database-level options can be displayed and modified by choosing the **Options** properties. There are several groups of options: **Automatic**, **Cursor**, **Miscellaneous**, **Recovery**, and **State**. Three options concern the state of a database:

- **Database Read-Only** Allows read-only access to the database. This prohibits users from modifying any data. (The default value is **False**.)
- **Database State** Describes the state of the database. (The default value is **Normal**.)
- **Restrict Access** Restricts the use of the database to one user at a time. (The default value is **Multiple**.)

If you choose the **Permissions** properties, SQL Server will display the corresponding dialog box with all users and roles along with their permissions. (For the discussion of permissions, see Chapter 12.)

Object Explorer can also be used to modify an existing database. Using Object Explorer, you can modify files and filegroups that belong to the database. To add new data files, right-click the database name, choose **Properties**, and select **Files**. In the **Database Properties** dialog box, click **Add** and type the name of the new file. (In this dialog box, you can also change the autogrowth properties and the location of each existing file.) You can also add a (secondary) filegroup for the database by selecting **Filegroups** and clicking **Add**. (A list of all properties that can be modified is given with the definition of the CREATE DATABASE statement in Chapter 4.)

NOTE

Only the system administrator or the database owner can modify the database properties mentioned above.

To delete a database using Object Explorer, right-click the database name and choose **Delete**.

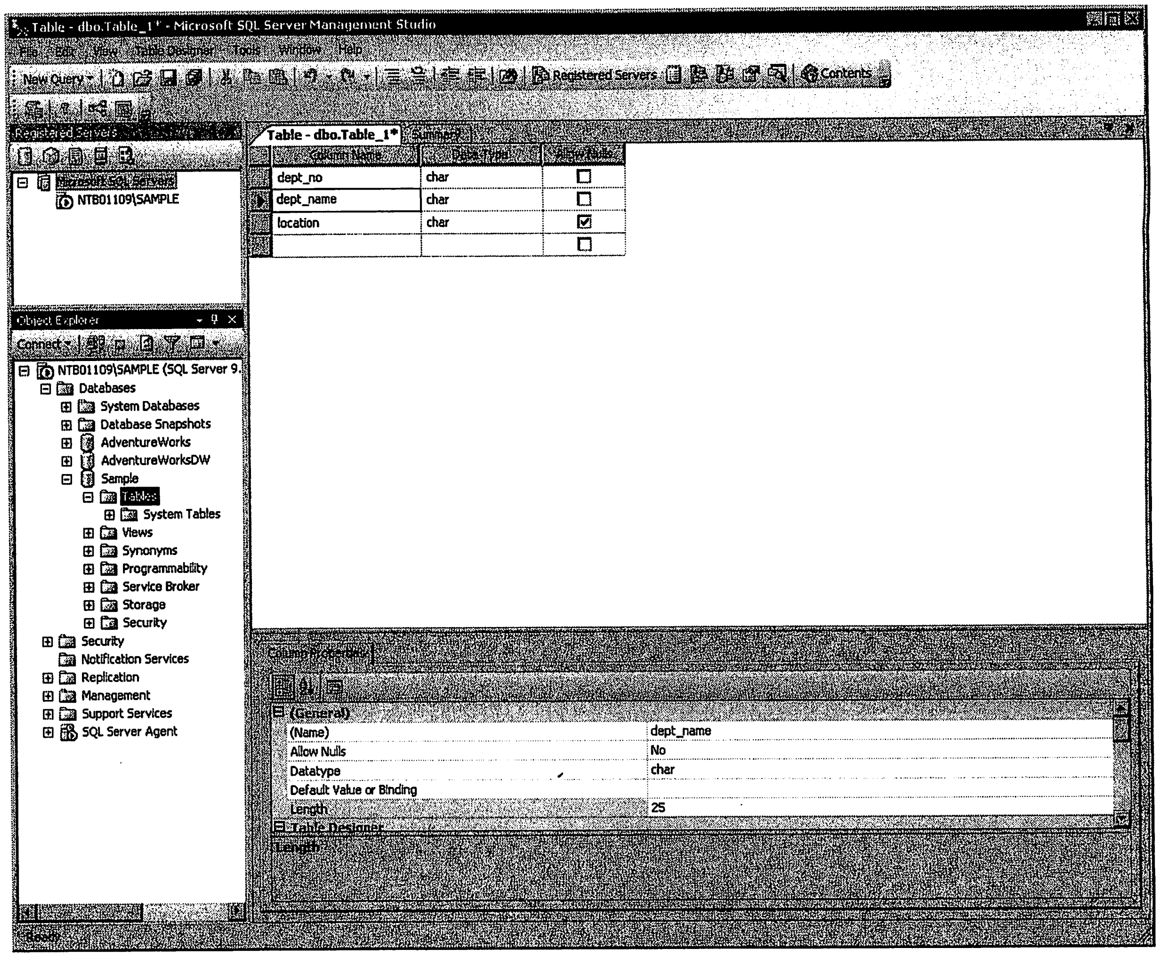
Managing Tables Using Object Explorer

The next task after the creation of a database is the creation of all tables belonging to it. Again, you can create tables by using either Object Explorer or Transact-SQL.

To create a table using Object Explorer, right-click the subfolder **Tables** of the database, and then click **New Table**. The creation of a table and all other database objects using the Transact-SQL language will be discussed in detail in Chapter 4.

To demonstrate the creation of a table using Object Explorer, the **department** table of the sample database will be used as an example. Enter the names of all columns with their properties in the **New Table** dialog box. Column names, their data types, as well as the NULL property of the column, must be entered in the two-dimensional matrix, as shown in Figure 2-7.

All data types supported by SQL Server can be displayed (and one of them selected) by clicking the arrow sign in the **Data Type** column (the arrow appears after the cell has been selected). Subsequently, you can type entries in the **Length**, **Precision**, and **Scale** rows for the chosen data type in the **Column Properties** window (see Figure 2-7). Some data types, such as CHARACTER, require a value for the **Length** row, and some, such as DECIMAL, require a value in the **Precision** and **Scale** rows. On the other hand, data types such as INTEGER do not need any of these entries to be specified. (The valid entries for a specified data type are highlighted in the list of all possible column properties.)



Column Name	Data Type	Allow Nulls
dept_no	char	☐
dept_name	char	☐
location	char	✓
		☐

(General)	
(Name)	dept_name
Allow Nulls	No
Datatype	char
Default Value or Binding	
Length	25

Figure 2-7 *Creating the department table using the SQL Server Management Studio*

The **Allow Nulls** column must be checked if you want a table column to permit null values to be inserted into that column. Similarly, if there is a default value, it should be entered in the **Default Value or Binding** row of the **Column Properties** window. (A default value is a value that will be inserted in a table column when there is no explicit value entered for it.)

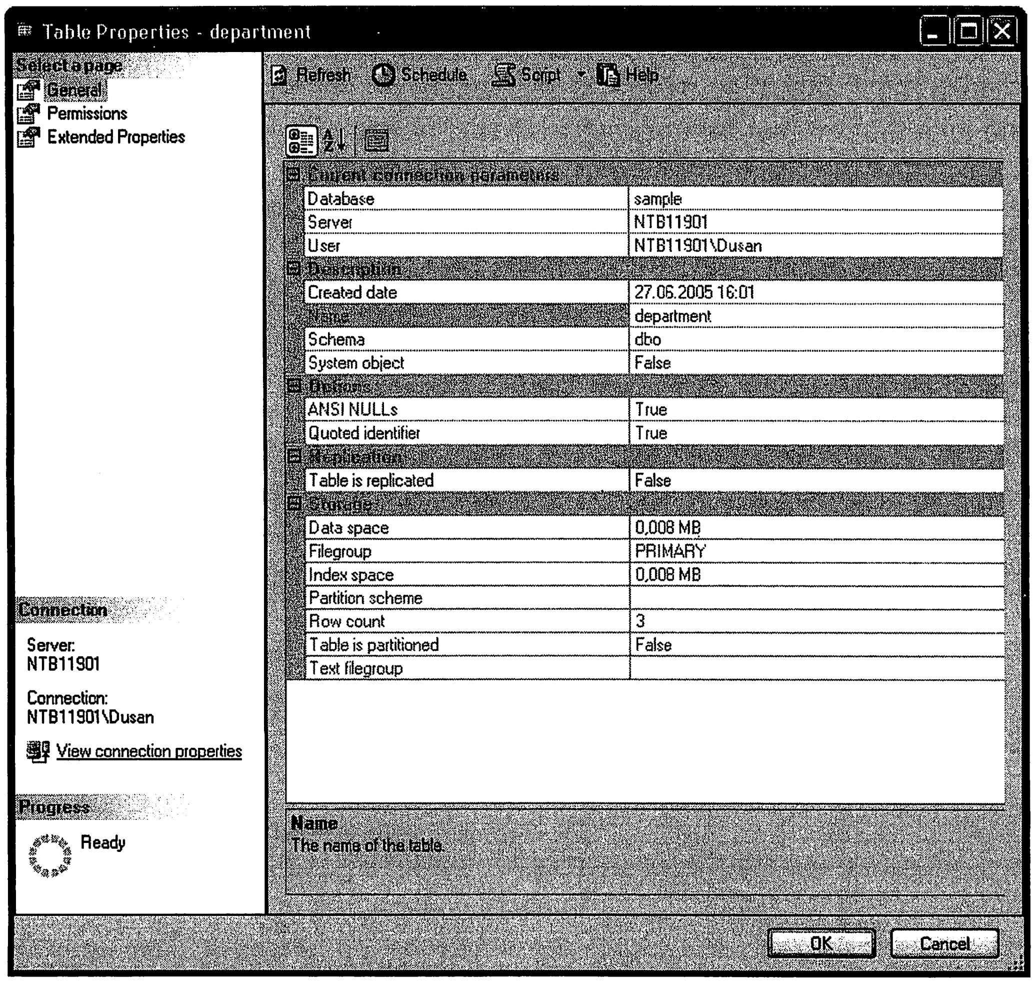
The column **dept_no** is the primary key of the **department** table. (For the discussion of primary keys of the sample database, see Chapter 1.) To specify a column as the primary key of a table, you must first right-click the column and then choose **Set Primary Key**. Finally, close the component window with the information concerning the new table. After that, the system will display the **Choose Name** dialog box, where you can type the table name.

To view the properties of an existing table, first double-click the folder of the database to which the table belongs. Subsequently, double-click **Tables**, and then right-click the name of the table and choose **Properties**. Figure 2-8 shows the **Table Properties** dialog box for the **department** table.

To rename a table, double-click the Tables folder and choose **Rename**. Also, to remove a table, double-click the **Tables** folder in the database to which the table belongs and select **Delete**.

If you create all four tables of the sample database (**employee**, **department**, **project**, and **works_on**), you can use another existing feature of SQL Server Management Studio to display the corresponding E/R- diagram of the sample database. (The process of converting the existing tables of a database in the corresponding E/R- diagram is called *reverse engineering*.)

To create the E/R- model of the sample database, right-click the **Database Diagrams** subfolder of the sample database folder and then select **New Database Diagram**. The first (and only) step is to select tables that will be added to the diagram.



Current connection parameters	
Database	sample
Server	NTB11901
User	NTB11901\Dusan
Description	
Created date	27.06.2005 16:01
Name	department
Schema	dbo
System object	False
Options	
ANSI NULLs	True
Quoted identifier	True
Replication	
Table is replicated	False
Storage	
Data space	0,008 MB
Filegroup	PRIMARY
Index space	0,008 MB
Partition scheme	
Row count	3
Table is partitioned	False
Text filegroup	

Name
The name of the table.

Figure 2-8 *The Table Properties dialog box for the department table*

After adding all four tables of the sample database, the wizard completes the work and creates the diagram (see Figure 2-9).

The diagram in Figure 2-9 is not the final diagram of the sample database, because it shows all four tables with their columns (and the corresponding primary keys), but it does not show any relationship between the tables. A relationship between two tables is based on the primary key of one table and the (possible) corresponding column(s) of the other. (For a detailed discussion of these relationships and referential integrity, see Chapter 4.)

There are exactly three relationships between the existing tables of the sample database: First, the tables **department** and **employee** have a 1:N relationship, because for each value in the primary key column of the **department** table (**dept_no**), there is one or more corresponding values in the column **dept_no** of the **employee** table **employee**. Analogously, there is a relationship between the tables **employee** and **works_on**, because only those values that exist in the primary key of the **employee** table (**emp_no**) appear also in the column **emp_no** of the **works_on** table. (The third relationship is between the tables **project** and **works_on**.)

To create each of the relationships described above, you have to redesign the diagram with the column that corresponds to the primary key column of the other

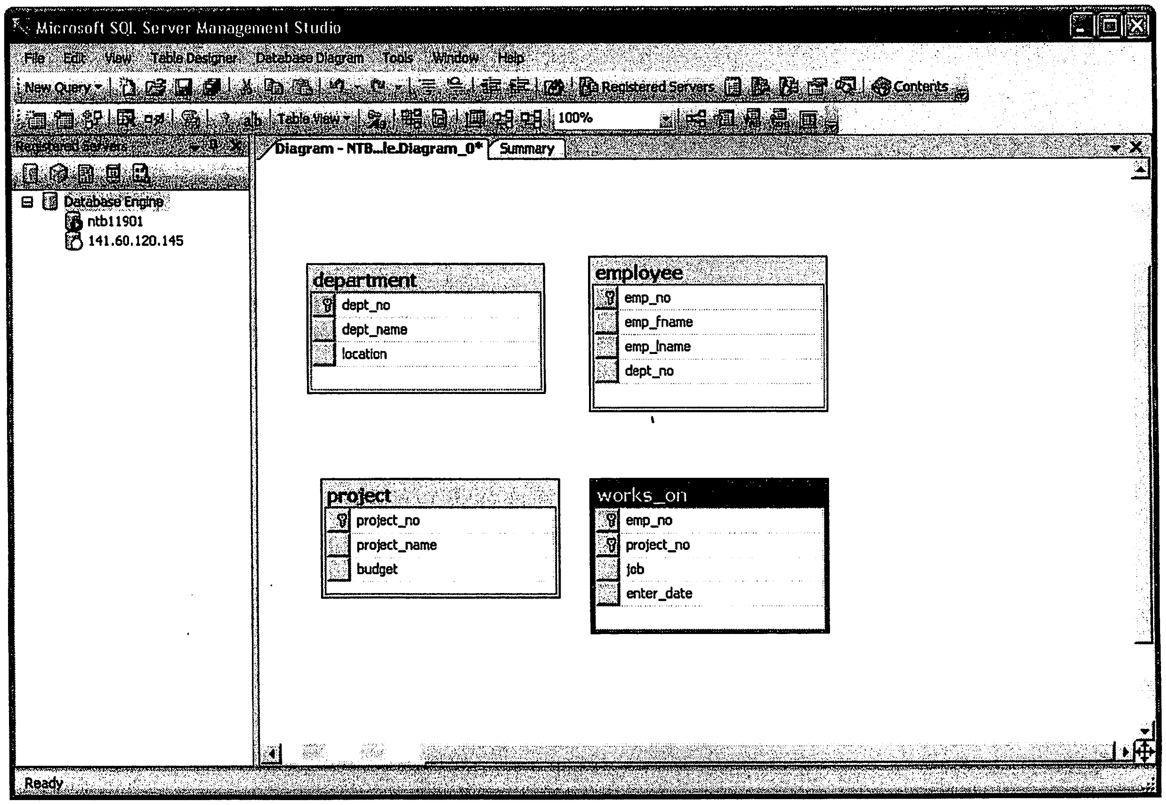


Figure 2-9 *First diagram of the sample database*

table. (Such a column is called a *foreign key.*) To show this, you can use the **employee** table and define its column **dept_no** as the foreign key of the **department** table.

Click the created diagram (it is called **sample_diagram** in our example), right-click the graphical form of the **employee** table in the detail pane, and select **Relationships**. In the **Foreign Key Relationships** dialog box, select **Add**.

Expand **Tables and Columns Specification** and click.... In the **Tables and Columns** dialog box, select the table with the corresponding primary key (the **department** table). Choose the **dept_name** column of this table as the primary key and the column with the same name in the **employee** table as the foreign key and click **OK**.

Figure 2-10 shows the modified **sample_diagram** diagram after all three relationships in the **sample** database have been created.

Authoring Activities Using SQL Server Management Studio

In the previous section we described the capabilities of SQL Server Management Studio, which concern management tasks. Beside these, SQL Server Management Studio gives you a complete authoring environment for all types of queries in SQL Server. You can create, save, load, and edit queries that execute SQL Server and other queries.

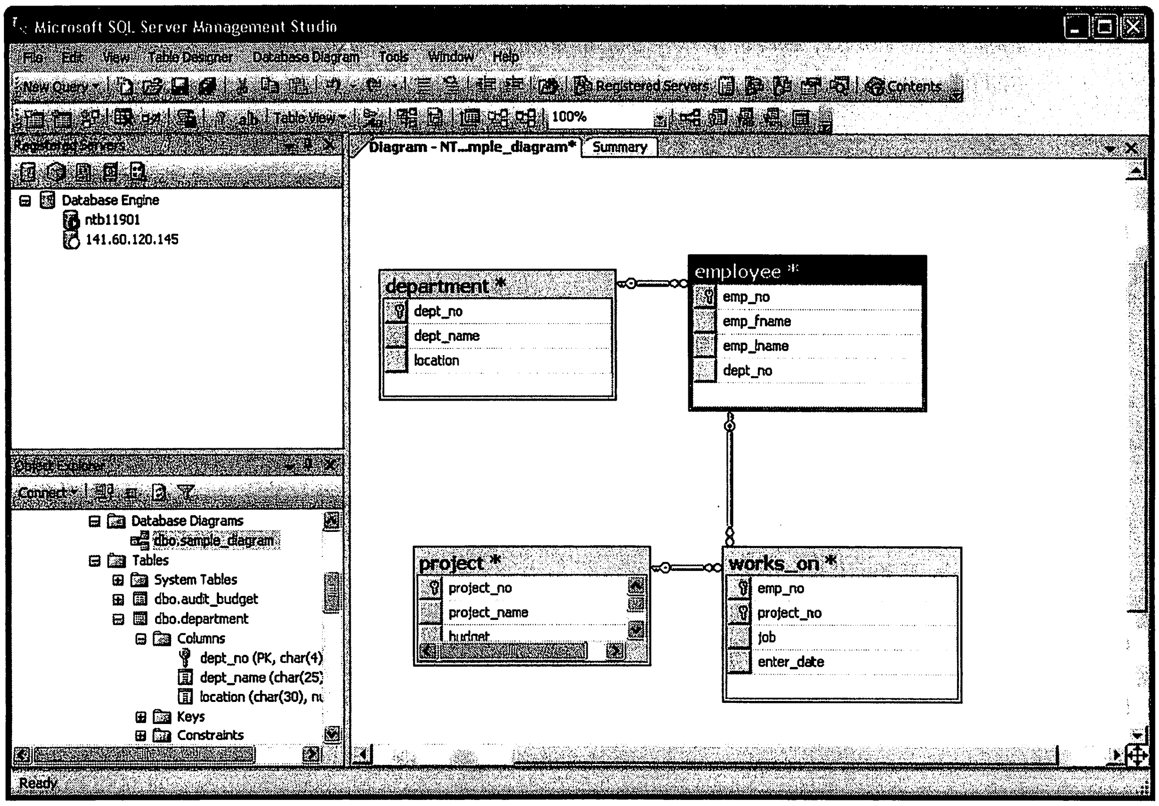


Figure 2-10 *The final diagram of the sample database*