APPENDIX A

Installation Instructions

In this appendix, you'll find instructions for installing the following applications:

- .NET Framework 2.0
- Microsoft Jet Engine
- Visual Web Developer 2005 Express Edition
- SQL Server 2005 Express Edition
- SQL Server 2005 Management Studio Express
- MySQL 5.0
- MySQL Query Browser 1.1
- MySQL Connector/ODBC 3.51
- MySQL Connector/NET 1.0

.NET Framework 2.0 Installation

Before you can install any of the other applications, you'll need to install .NET Framework 2.0. Follow these steps:

1. Download the .NET 2.0 Redistributable Package installer from the Microsoft site at `http://msdn.microsoft.com/netframework/downloads/updates/default.aspx`. Follow the quick link for .NET Framework 2.0 and download the correct version of the redistributable package. (`dotnetfx.exe` is 22.4MB.)
2. After the package has downloaded, double-click it and let the installer run.

Microsoft Jet Engine Installation

Since the release of Microsoft Data Access Components (MDAC) 2.6, the Jet engine isn't installed by default. Therefore, if you don't have Microsoft Access installed, you may not have the Jet engine. To install the latest version, follow these instructions:

1. Download the latest version of the Jet engine from the Microsoft Jet Security Bulletin MS04-014 page of Microsoft TechNet, `http://www.microsoft.com/technet/security/bulletin/ms04-014.mspx`.
2. After the package has downloaded, double-click the installer and let it run.

Visual Web Developer 2005 Express Edition Installation

All of the examples in the book are built using Visual Web Developer 2005 Express Edition. At the time of writing, you can download it for free.

■ **Note** Visual Web Developer Express Edition is available free for only a *limited period*, which as yet doesn't have a specified end date. Microsoft has been saying "for the next year," so you should be able to get the free version until at least the end of December 2006.

Follow these instructions to download and install Visual Web Developer Express Edition.

1. Download the latest version of Visual Web Developer from `http://msdn.microsoft.com/vstudio/express/vwd`. (`vwdsetup.exe` is a little under 3MB.)
2. After the package has downloaded, double-click the installer to run it.
3. The first step of the wizard offers the Help Improve Setup option. This doesn't affect the installation process at all. You can check what information is sent to Microsoft by visiting `http://msdn.microsoft.com/vstudio/products/privacy`. Once you've made your choice, click the Next button.
4. Accept the terms of the license agreement and click the Next button.
5. You don't need to install any of the extra packages, but you can choose to if you want them. The documentation may come in quite handy, but it is a hefty download at 248MB. Do not install SQL Server 2005 Express Edition, as some of the options that you'll need to set are not available if you install via this route. Click the Next button.
6. Click the Install button to start the installation. The installer will connect to the Internet to download the necessary components (40MB for Visual Web Developer on its own). If you want to get a cup of coffee, now is your chance.
7. Once setup is complete, you can choose to register your copy of Visual Web Developer for a few extras. To decide if you want to register, see `http://msdn.microsoft.com/vstudio/express/register/default.aspx`.

SQL Server 2005 Express Edition Installation

To run the SQL Server 2005 versions of the examples in the book, you'll need to install SQL Server 2005 Express Edition and SQL Server 2005 Management Studio Express.

SQL Server 2005 Express Edition is a freely available, cut-down version of Microsoft's SQL Server 2005 Enterprise Database Server. To install it, follow these steps:

1. Download SQL Server 2005 Express Edition from `http://msdn.microsoft.com/vstudio/express/sql/default.aspx`. (`SQLEXPR.EXE` is 53.5MB, so make sure there is enough space on your hard drive, and have another cup of coffee handy.)
2. After the package has downloaded, double-click it. The necessary files will be extracted, and the installer will be run automatically.
3. Accept the terms of the license agreement and click the Next button.
4. The SQL Server 2005 Express Edition prerequisites will now be installed. This will take a couple of minutes. Once the prerequisites have been installed, click the Next button.
5. The installer will scan your computer's configuration and then run the "real" installer. Click the Next button to proceed with the installation.
6. The installer will now perform a system configuration check to ensure that the minimum installation requirements are met. Click the Next button to continue with the installation. It is quite common at this point to have a Pending Reboot Requirement specified, halting the installation. In this case, reboot your computer and run the installer again.
7. On the next step, enter your registration information and uncheck the Hide Advanced Configuration Options check box. Click the Next button.
8. Accept the selected features by clicking the Next button.
9. You're going to install SQL Server 2005 Express as a named instance, so select the Named Instance option and enter `BAND`, as shown in Figure A-1. (You can have multiple instances of SQL Server 2005 running on the same machine, and you've used an instance named `BAND` to keep the databases for this book separated from others.) Click the Next button.
10. On the Service Account step, select the default option of using the Network Service account by clicking the Next button.
11. On the Authentication Mode step, switch the authentication mode of the database to Mixed Mode by clicking the correct radio button and entering the password that you want for the `sa` account. For the purposes of this book, enter **bandpass** as both the password and its confirmation, as shown in Figure A-2. (You can use a different password if you prefer, but then you must make sure that you use the password you have chosen instead of the one in the instructions throughout the examples in this book.) Click the Next button.

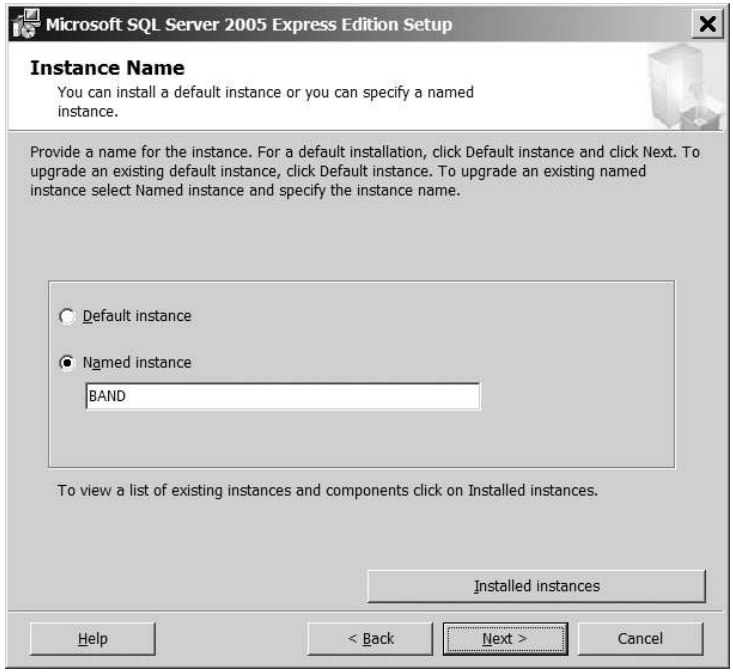

Figure A-1. *Specifing the name of the SQL Server instance to create*

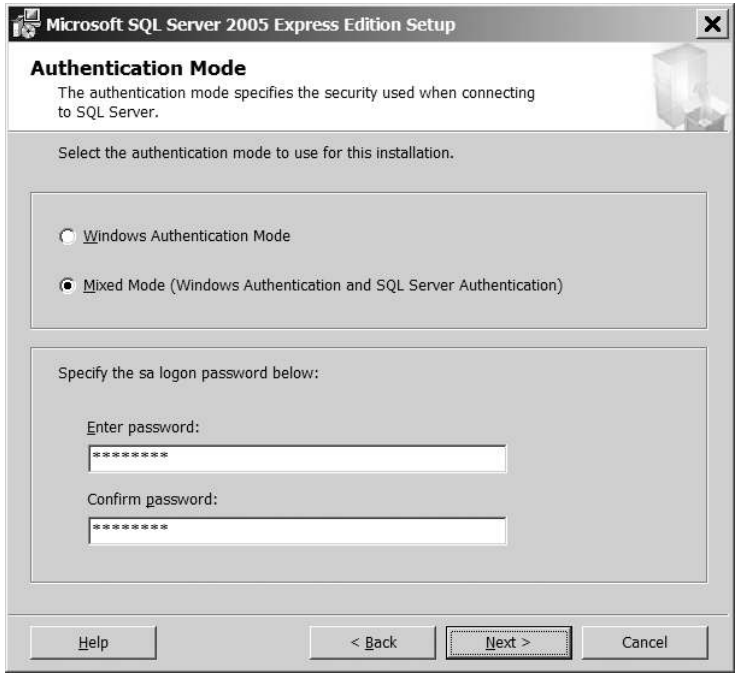

Figure A-2. *Entering the login details for the sa account*

12. On the Collation Settings step, select the default options by clicking the Next button.
13. Click Next to allow user instances of SQL Server.
14. On the Error and Usage Reporting Settings step, you can choose to turn on reporting if you wish. Once you've made your choice, click the Next button.
15. Click the Install button to begin installation of SQL Server 2005 Express Edition.
16. The installation may take a little time. Once it is finished, click the Next button to continue to the Summary step. Click Finish to close the installer.
17. You may need to restart your machine once the installer has finished.

You now need to check that SQL Server 2005 Express Edition has installed correctly. The way to do this differs across platforms, but for Windows XP, you can find it by selecting Start Menu ➤ Settings ➤ Control Panel ➤ Administrative Tools ➤ Services. This launches the Services tool, as shown in Figure A-3.

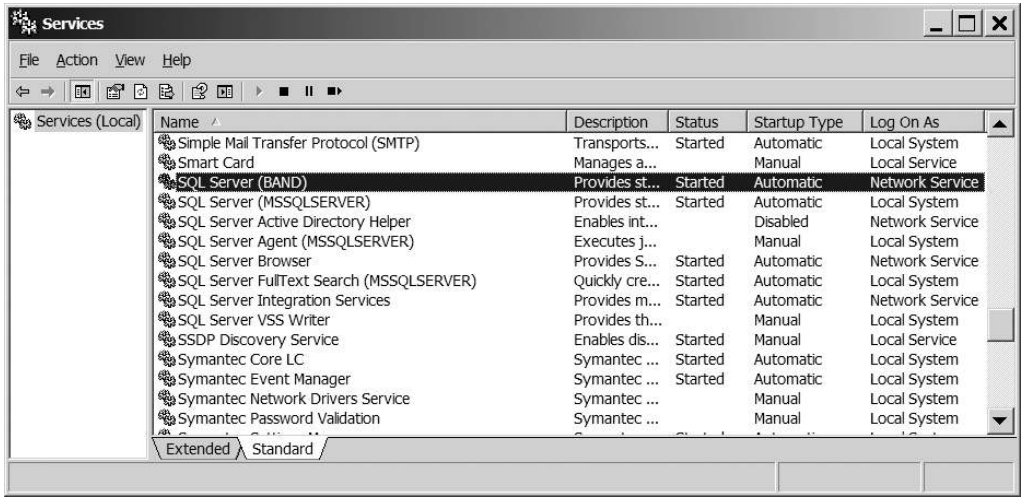

Figure A-3. *The Services tool shows the new instance of SQL Server 2005 Express Edition.*

Look for a service called SQL Server (BAND) in the list. This is your installation of SQL Server 2005 Express Edition. As you can see in Figure A-3, there are two instances of SQL Server 2005 installed on the machine: the instance that you've just installed (`BAND`) and the default instance on this machine (`MSSQLSERVER`). Make sure the Status and Startup Type options are set to Started and Automatic, respectively. The Startup Type setting will ensure that SQL Server 2005 Express Edition starts automatically whenever your computer is rebooted.

■ **Note** You can install several instances of SQL Server 2005 side by side on one machine. In fact, you can even install different versions of SQL Server on the same machine. The one for this book is named BAND (for Beginning ASP.NET Databases), and as it's a SQL Server 2005 database, it's called SQL Server (BAND).

SQL Server 2005 Management Studio Express Installation

SQL Server 2005 Management Studio Express is a graphical front-end tool for administering SQL Server 2005. Although the current free version is a Community Technical Preview, it is very stable. It's a cut-down version of the tool that is provided with the full version of SQL Server 2005. To install it, follow these steps:

1. Go to `http://msdn.microsoft.com/vstudio/express/sql/default.aspx` and click the Download SQL Server Management Studio Express link. Then download the correct installer. (`SQLServer2005_SSMSEE.msi` is slightly under 30MB, so you'll need your third cup of coffee here.)
2. After the package has downloaded, open the installer by double-clicking it.
3. Skip past the Welcome step by clicking the Next button, accept the terms and conditions on the next step, and click the Next button again.
4. Enter your registration details and click the Next button.
5. Click the Next button to accept the selected features, and then click the Install button to begin the installation.
6. Once the installation has completed, click the Finish button to close the installer.

MySQL 5.0 Installation

To build and run the MySQL 5.0 examples, you'll need to install MySQL 5.0, as well as MySQL Query Browser and MySQL Connector/ODBC. To follow the stored procedure examples in Chapter 10, you'll need to install MySQL Connector/NET.

To install the database server, follow these steps:

1. Download the Community Edition - Windows Essentials installer from `http://dev.mysql.com/downloads/mysql/5.0.html`. (`mysql-essential-5.0.18-win32.msi` is 16.8MB.)
2. After the package is downloaded, double-click it. The files needed will be extracted, and the installer will run automatically.
3. Click the Next button on the Welcome step and accept the Typical installation options by clicking the Next button on the following step. Click the Install button to start the actual installation.
4. On the Sign Up step, you can choose to log in or create an account to MySQL.com. Alternatively, you can click the Skip Sign-Up option. Once you've made your selection, click the Next button.
5. Click the Finish button to configure the newly installed database server.

6. Click the Next button on the first step of the Configuration wizard and choose the Standard Configuration on the following step. Click the Next button.
7. Make sure that both the Install As Windows Service and Include Bin Directory in Windows PATH options are checked, and then click the Next button.
8. Enter a **bandpass** as both the password and its confirmation. Then click the Next button.
9. Click the Execute button to perform the configuration.
10. Once the configuration is completed, click the Finish button to close the Configuration wizard.

To check that MySQL 5.0 has installed correctly, you need to verify that the service is installed and is running. Open the Services tool by selecting Start Menu ➤ Settings ➤ Control Panel ➤ Administrative Tools ➤ Services. In the list of services, look for the MySQL service, as shown in Figure A-4. Make sure the Status and Startup Type options are set to Started and Automatic, respectively. The Startup Type setting will ensure that MySQL 5.0 is started automatically whenever your computer is rebooted.

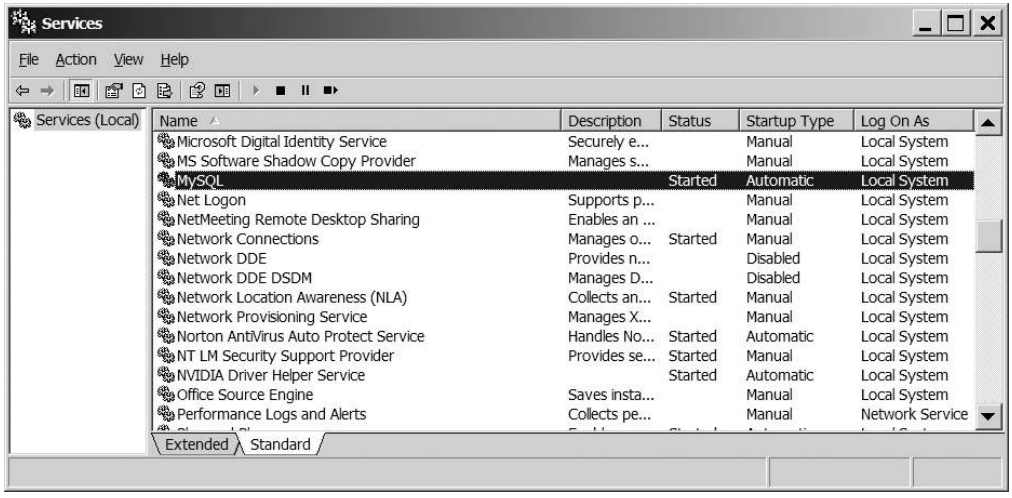

Figure A-4. *The Services tool shows the MySQL service.*

MySQL Query Browser 1.1 Installation

MySQL Query Browser 1.1 is a graphical front-end tool for administering MySQL 5.0. To install it, follow these instructions:

1. Download the Windows installer from `http://dev.mysql.com/downloads/query-browser/1.1.html`. (`mysql-query-browser-1.1.19-win.msi` is 5.1MB.)
2. After the package has downloaded, double-click it. The necessary files will be extracted, and the installer will run automatically.

3. Accept the terms of the license agreement.
4. On the Setup Type step of the installation wizard, choose to perform a Complete installation.

MySQL Connector/ODBC 3.51 Installation

In order to use MySQL 5.0 with the `Odbc` data provider, you need to install the MySQL ODBC driver. To install Connector/ODBC 3.51, follow these steps:

1. Download the Windows MSI package from `http://dev.mysql.com/downloads/connector/odbc/3.51.html`. (`mysql-connector-odbc-3.51.12-win32.msi` is 2.3MB.)
2. After the package has downloaded, double-click it. The necessary files will be extracted, and the installer will run automatically.
3. Accept the terms of the license agreement.
4. On the Setup Type step of the installation wizard, choose to perform a Typical install.

MySQL Connector/NET 1.0 Installation

In order to execute stored procedures in MySQL 5.0, you need to install the `MySqlClient` data provider. (The MySQL ODBC driver doesn't support all of the necessary features.) To install Connector/NET 1.0, follow these steps:

1. Download the Source and Binaries ZIP file from `http://dev.mysql.com/downloads/connector/net/1.0.html`. (`mysql-connector-net-1.0.7.zip` is 545KB.)
2. Open the downloaded ZIP file, and then double-click the `MySql.Data.msi` installer to run the installer.
3. On the Setup Type step of the installation wizard, choose to perform a Typical installation.

APPENDIX B

SQL Data Types

Databases support the same core set of data types as defined in the SQL standard, but annoyingly, they call these data type different things. This appendix provides an easy reference to the data types defined in SQL Server 2005, MySQL 5.0, and Microsoft Access. In this appendix, the data types are grouped as follows:

- Text types
- Numeric types
- Date and time types
- Binary types
- Miscellaneous types

Text Types

Several text-based types are defined in SQL and are distinguished by the following two main characteristics:

- Does it support Unicode?
- Does it have a fixed length or a variable length?

If you choose a fixed-length type for a field, and the string it contains is smaller than that length, the string is padded with spaces to make it the correct length. This padding won't occur if you use a variable-length data type, but you must specify a maximum length for the string using the `Length/Size` property for the field.

char

The `char` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In SQL Server, `char` represents a fixed-length string of up to 8,000 non-Unicode characters. You should use the `SqlDbType.Char` type identifier for parameters of this type.
- In MySQL, `char` represents a fixed-length string of up to 255 non-Unicode characters. Depending on the data provider, you should use the `OdbcType.Char` or `MySqlClient.String` type identifiers for parameters of this type.
- In Microsoft Access, the `char` type is called `Text` and represents a variable-length string of up to 255 Unicode characters. You should use the `OleDbType.Char` type identifier for parameters of this type.

longtext

The `longtext` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In MySQL, `longtext` represents a variable-length string of up to 2^{32}-1 (4,294,967,295) non-Unicode characters. Depending on the data provider, you should use the `OdbcType.Text` or `MySqlClient.String` type identifiers for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

mediumtext

The `mediumtext` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In MySQL, `mediumtext` represents a variable-length string of up to 2^{24}-1 (16,777,215) non-Unicode characters. Depending on the data provider, you should use the `OdbcType.Text` or `MySqlClient.String` type identifiers for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

nchar/national char

The `nchar/national char` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In SQL Server, `nchar` represents a fixed-length string of up to 4,000 Unicode characters. You should use the `SqlDbType.NChar` type identifier for parameters of this type.
- In MySQL, `national char` represents a fixed-length string of up to 255 Unicode characters. Depending on the data provider, you should use the `OdbcType.NChar` or `MySqlClient.String` type identifiers for parameters of this type.
- There is no equivalent in Microsoft Access.

nvarchar/national varchar

The `nvarchar/national varchar` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In SQL Server, `nvarchar` represents a variable-length string of up to 4,000 Unicode characters. You should use the `SqlDbType.NVarChar` type identifier for parameters of this type.
- In MySQL, `national varchar` represents a variable-length string of up to 65,535 Unicode characters. Depending on the data provider, you should use the `OdbcType.NVarChar` or `MySqlClient.String` type identifiers for parameters of this type.
- There is no equivalent in Microsoft Access.

SQL Server 2005 defines a new type `nvarchar(max)` that allows 2^{30}-1 (1,073,741,823) Unicode characters to be stored. This is still a `string` in C#, but you should use the `SqlDbType.NText` type identifier for parameters of this type.

ntext

The `ntext` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In SQL Server, `ntext` represents a variable-length string of up to 2^{30}-1 (1,073,741,823) Unicode characters. You should use the `SqlDbType.NText` type identifier for parameters of this type.
- There is no equivalent in MySQL or Microsoft Access.

text

The `text` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In SQL Server, `text` represents a variable-length string of up to 2^{31}-1 (2,147,483,647) non-Unicode characters. You should use the `SqlDbType.Text` type identifier for parameters of this type.
- In MySQL, `text` represents a variable-length string of up to 65,535 non-Unicode characters. Depending on the data provider, you should use the `OdbcType.Text` or `MySqlClient.String` type identifiers for parameters of this type.
- In Microsoft Access, the `text` type is called `Memo` and represents a variable-length string of up to 2^{31}-1 (2,147,483,647) Unicode characters. You should use the `OleDbType.Char` type identifier for parameters of this type.

tinytext

The `tinytext` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In MySQL, `tinytext` represents a variable-length string of up to 255 non-Unicode characters. Depending on the data provider, you should use the `OdbcType.Text` or `MySqlClient.String` type identifiers for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

varchar

The `varchar` type maps to a `string` (`System.String`) in C# and is defined as follows:

- In SQL Server, `varchar` represents a variable-length string of up to 8,000 non-Unicode characters. You should use the `SqlDbType.VarChar` type identifier for parameters of this type.
- In MySQL, `varchar` represents a variable-length string of up to 65,535 non-Unicode characters. Depending on the data provider, you should use the `OdbcType.VarChar` or `MySqlClient.String` type identifiers for parameters of this type.
- There is no equivalent in Microsoft Access.

SQL Server 2005 defines a new type `varchar(max)` that allows 2^{31}-1 (2,147,483,647) non-Unicode characters to be stored. This is still a `string` in C#, but you should use the `SqlDbType.Text` type identifier for parameters of this type.

Numeric Types

Like .NET types, data types are available in SQL Server and MySQL for both integer and floating-point values. However, there's a difference in implementation. Both databases assume by default that these types are signed—that is, they have negative values—but only MySQL gives you the option to make a type unsigned (by clicking the UNSIGNED check box in MySQL Query Browser).

In addition, all numeric types are fixed-length data types. If a field contains a value that doesn't use all of its allocated value, it's padded with spaces. In MySQL Query Browser, if you click the ZEROFILL check box, numeric values are padded with zeros instead of spaces.

Autonumbers

Supporting the notion that primary key fields must contain unique values, all three database types can auto-generate unique integer values for ID (primary key) fields. They can also generate globally unique ID fields (GUIDs)—128-bit hexadecimal numbers that are both random and unique. The following are the various ways to do this:

- To auto-generate integers in SQL Server, set the field's data type to one of the integer values and set `IsIdentity` to `true`. The properties `IdentitySeed` and `IdentityIncrement` let you set the first value to be generated for the field and the difference between each subsequent value, respectively.

- To auto-generate GUIDs in SQL Server, set the field's data type to `uniqueidentifier` and its default value to `newid()`.
- To auto-generate integers in the Jet engine, set the field's data type to `AutoNumber`.
- To auto-generate GUIDs in the Jet engine, set the field's data type to `ReplicationID` and then set `Autogenerate` to `true`.
- MySQL supports only auto-generated integers. Set the field's data type to one of the integer data types and set `AUTO_INCREMENT` to `true`.

Integer Types

Integer types are those types that can hold only nondecimal numbers.

bigint

You should use `bigint` only if you're absolutely sure that the integers you need to store cannot fit in an `int` data field. The `bigint` type represents an 8-byte integer and maps to a `long` (`System.Int64`) in C#. It is defined as follows:

- In SQL Server, `bigint` can take values between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807). You should use the `SqlDbType.BigInt` type identifier for parameters of this type.
- For a signed `bigint`, MySQL allows the same range of values as SQL Server. An unsigned `bigint` can represent a range of integers between 0 and 2^{64} (18,446,744,073,709,551,616). Depending on the data provider, you should use the `OdbcType.BigInt` or `MySqlClient.Int64` type identifiers for parameters of this type.
- There is no equivalent in Microsoft Access.

bit

The `bit` type is typically used as the data type to store Boolean values. Columns of type `bit` can't have indexes on them. The `bit` type maps to a `bool` (`System.Boolean`) in C# and is defined as follows:

- In SQL Server, `bit` is an integer type that can take two values: 0 or 1. You should use the `SqlDbType.Bit` type identifier for parameters of this type.
- In MySQL, `bit` is an integer type that can take two values: 0 or 1. It's also known as `bool` or `boolean`. Depending on the data provider, you should use the `OdbcType.Bit` or `MySqlClient.Bit` type identifiers for parameters of this type.
- In Microsoft Access, `bit` is called `Yes/No` and can take the values of Yes (1) or No (0). You should use the `OleDbType.Boolean` type identifier for parameters of this type.

int

The `int` type represents a 4-byte integer and maps to an `int` (`System.Int32`) in C#. It is defined as follows:

- In SQL Server, an `int` represents a range of integers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647). You should use the `SqlDbType.Int` type identifier for parameters of this type.
- For a signed `int`, MySQL uses the same definition as SQL Server. An unsigned `int` can represent a range of integers between 0 and 2^{32} (4,294,967,296). Depending on the data provider, you should use the `OdbcType.Int` or `MySqlClient.Int32` type identifiers for parameters of this type.
- In Microsoft Access, an `int` type is represented as a `Number` data type and a `Long Integer` field size. It supports the same range of values as SQL Server, and you should use the `OleDbType.Integer` type identifier for parameters of this type.

mediumint

The `mediumint` type represents a 3-byte integer and has no direct mapping in .NET. The closest is an `int` in C#. It is defined as follows:

- A signed `mediumint` in MySQL represents a range of integers between -2^{23} (-8,388,608) and $2^{23}-1$ (8,388,607). An unsigned `mediumint` can represent a range of integers between 0 and 2^{24} (16,777,216). Depending on the data provider, you should use the `OdbcType.Int` or `MySqlClient.Int24` type identifiers for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

smallint

The `smallint` type represents a 2-byte integer and maps to a `short` (`System.Int16`) in C#. It is defined as follows:

- In SQL Server, a `smallint` represents a range of integers between -2^{15} (-32,768) and $2^{15}-1$ (32,767). You should use the `SqlDbType.SmallInt` type identifier for parameters of this type.
- For a signed `smallint`, MySQL uses the same definition as SQL Server. An unsigned `smallint` can represent a range of integers between 0 and 2^{16} (65,536). Depending on the data provider, you should use the `OdbcType.SmallInt` or `MySqlClient.Int16` type identifiers for parameters of this type.
- In Microsoft Access, a `smallint` type is represented as a `Number` data type and an `Integer` field size. It supports the same range of values as SQL Server, and you should use the `OleDbType.SmallInt` type identifier for parameters of this type.

tinyint

The `tinyint` type represents a 1-byte integer and maps to a `sbyte` (`System.SByte`) in C#. It is defined as follows:

- In SQL Server, a `tinyint` represents a range of integers between 0 and 255. You should use the `SqlDbType.TinyInt` type identifier for parameters of this type.
- For an unsigned `tinyint`, MySQL uses the same definition as SQL Server. A signed `tinyint` can represent a range of integers between -128 and 127. Depending on the data provider, you should use the `OdbcType.TinyInt` or `MySqlClient.Byte` type identifier for parameters of this type.
- In Microsoft Access, a `tinyint` type is represented as a `Number` data type and a `Byte` field size. It supports the same range of values as SQL Server, and you should use the `OleDbType.TinyInt` type identifier for parameters of this type.

decimal

The `decimal` type represents a number range defined by a maximum number of digits (its *precision*) and the maximum number of digits that can be used to the right of the decimal point (its *scale*). It maps to a `decimal` (`System.Decimal`) in C# and is defined as follows:

- In SQL Server, the `decimal` type can have a maximum precision of 38. The default precision is 18, and the default scale is 0. You should use the `SqlDbType.Decimal` type identifier for parameters of this type.
- In MySQL, the `decimal` type can have a maximum precision of 65 and a maximum scale of 30. The default precision is 10, and the default scale is 0. Depending on the data provider, you should use the `OdbcType.Decimal` or `MySqlClient.Decimal` type identifier for parameters of this type.
- In Microsoft Access, the `decimal` type is represented as a `Number` data type and a `Decimal` field size. The default precision is 18, and the default scale is 0. You should use the `OleDbType.Number` type identifier for parameters of this type.

Variable-Size Floating-Point Numbers

The `decimal` type specifies a fixed number of digits both before and after the decimal point, and will always store the number you want exactly as you enter it. With variable-size floating-point numbers, the value that is stored may be imprecise. The more bytes that are used to store the number, the more precise it will be.

4-Byte Floating-Point Numbers

A 4-byte (32-bit) floating-point number represents a fixed range of values between $3.40E^{38}$ to $-1.18E^{-38}$ for negative values, zero, and $1.18E^{-38}$ to $3.40E^{38}$ for positive values. It maps to a `float` (`System.Single`) in C# and is defined as follows:

- In SQL Server, a 4-byte floating-point number is a `real`. You should use the `SqlDbType.Real` type identifier for parameters of this type.
- In MySQL, a 4-byte floating-point number is a `float`. Depending on the data provider, you should use the `OdbcType.Real` or `MySqlClient.Float` type identifier for parameters of this type.
- In Microsoft Access, a 4-byte floating-point number is represented as a `Number` data type and a `Single` field size. You should use the `OleDbType.Single` type identifier for parameters of this type.

8-Byte Floating-Point Numbers

An 8-byte (64-bit) floating-point number represents a fixed range of values between $1.79E^{308}$ to $-2.23E^{-308}$ for negative values, zero, and $2.23E^{-308}$ to $1.79E^{308}$ for positive values. It maps to a `double` (`System.Double`) in C# and is defined as follows:

- In SQL Server, an 8-byte floating-point number is a `float`. You should use the `SqlDbType.Float` type identifier for parameters of this type.
- In MySQL, an 8-byte floating-point number is a `double`. Depending on the data provider, you should use the `OdbcType.Double` or `MySqlClient.Double` type identifier for parameters of this type.
- In Microsoft Access, an 8-byte floating-point number is represented as a `Number` data type and a `Double` field size. You should use the `OleDbType.Double` type identifier for parameters of this type.

Date and Time Types

All three databases support "instance-in-time" fields, but only MySQL supports time unit types.

date

The `date` type maps to a `System.DateTime` in C# without a time specified (00:00:00). It is defined as follows:

- In MySQL, a `date` represents a date between 1000-01-01 and 9999-12-31. Depending on the data provider, you should use the `OdbcType.Date` or `MySqlClient.Date` type identifiers for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

datetime

The `datetime` type maps to a `System.DateTime` in C# and is defined as follows:

- In SQL Server, `datetime` represents a date and time combination between 00:00:00 on Jan. 1, 1753, through to 23:59:59 on Dec. 31, 9999. You should use the `SqlDbType.DateTime` type identifier for parameters of this type.
- In MySQL, `datetime` represents a date and time combination between 00:00:00 on Jan. 1, 1000, through to 23:59:59 on Dec. 31, 9999. Depending on the data provider, you should use the `OdbcType.DateTime` or `MySqlClient.Datetime` type identifier for parameters of this type.
- In Microsoft Access, a `datetime` is represented as a `Date/Time` data type and represents a date and time combination between 00:00:00 on Jan. 1, 1000, through to 23:59:59 on Dec. 31, 9999. You should use the `OleDbType.DBDate` type identifier for parameters of this type.

smalldatetime

The `smalldatetime` type maps to a `System.DateTime` in C# and is defined as follows:

- In SQL Server, `smalldatetime` represents a date and time combination between 00:00:00 on Jan. 1, 1900, through to 23:59:59 on June 6, 2079. You should use the `SqlDbType.SmallDateTime` type identifier for parameters of this type.
- There is no equivalent in MySQL or Microsoft Access.

time

The `time` type maps to a `System.TimeSpan` in C# and is defined as follows:

- In MySQL, time represents a period of time in the format *HH*:*MM*:*SS* between -838:59:59 and 838:59:59. Depending on the data provider, you should use the `OdbcType.Time` or `MySqlClient.Time` type identifier for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

timestamp

The `timestamp` type maps to a `byte` array in C# and is defined as follows:

- In SQL Server, a `timestamp` represents an automatically generated 8-byte binary number (which is guaranteed to be unique within a database) given to the field when its row is added to the table or modified. More information can be found at `http://msdn.microsoft.com/en-us/library/ms182776.aspx`. You should use the `SqlDbType.Timestamp` type identifier for parameters of this type.

- In MySQL, a `timestamp` is a date and time value automatically generated and given to the field when its row is added to the table or modified. More information can be found at `http://dev.mysql.com/doc/refman/5.0/en/timestamp-4-1.html`. Depending on the data provider, you should use the `OdbcType.Timestamp` or `MySqlClient.Timestamp` type identifier for parameters of this type.
- There is no equivalent in Microsoft Access.

year

The `year` type maps to an `int` (`System.Int32`) in C# without a day, month, or time specified. It is defined as follows:

- In MySQL, a `year` represents a range of years from 1901 to 2155 if you're using four digits and 1970 to 2066 if you're using two digits. Depending on the data provider, you should use the `OdbcType.Int` or `MySqlClient.Year` type identifier for parameters of this type.
- There is no equivalent in SQL Server or Microsoft Access.

Binary Types

All databases define a few data types for binary data for storing items such as compiled programs, images, and audio. You'll see the rather unflattering acronym for binary data in general—BLOBs, for Binary Large OBjects.

SQL Server Binary Types

SQL Server has three binary types: `binary`, `image`, and `varbinary`.

binary

The `binary` type maps to a `byte` array in C# and represents a fixed-length binary sequence of up to 8,000 bytes. You should use the `SqlDbType.Binary` type identifier for parameters of this type.

image

The `image` type maps to a `byte` array in C# and represents a variable-length binary sequence of up to 2^{31}-1 (2,147,483,647) bytes. You should use the `SqlDbType.Image` type identifier for parameters of this type.

varbinary

The `varbinary` type maps to a `byte` array in C# and represents a variable-length binary sequence of up to 8,000 bytes. You should use the `SqlDbType.VarBinary` type identifier for parameters of this type.

MySQL Binary Types

MySQL has four binary types: `blob`, `longblob`, `mediumblob`, and `tinyblob`.

blob

The `blob` type maps to a `byte` array in C# and represents a variable-length binary sequence of up to 65,535 bytes. Depending on the data provider, you should use the `OdbcType.VarBinary` or `MySqlClient.Blob` type identifier for parameters of this type.

longblob

The `longblob` type maps to a `byte` array in C# and represents a variable-length binary sequence of up to 2^{32}-1 (4,294,967,295) bytes. Depending on the data provider, you should use the `OdbcType.VarBinary` or `MySqlClient.LongBlob` type identifier for parameters of this type.

mediumblob

The `mediumblob` type maps to a `byte` array in C# and represents a variable-length binary sequence of up to 2^{24}-1 (16,777,215) bytes. Depending on the data provider, you should use the `OdbcType.VarBinary` or `MySqlClient.MediumBlob` type identifier for parameters of this type.

tinyblob

The `tinyblob` type maps to a `byte` array in C# and represents a variable-length binary sequence of up to 255 bytes. Depending on the data provider, you should use the `OdbcType.VarBinary` or `MySqlClient.TinyBlob` type identifier for parameters of this type.

Microsoft Access Binary Data

Microsoft Access handles binary data through a single type, `OLE Object`, which allows you to store as much binary data as there is space on disk. You should use the `OleDbType.VarBinary` type identifier for parameters of this type.

Miscellaneous Types

Several types aren't easily classifiable into the previous categories. These include `enum('value1','value2',...)`, `money`, `set('value1','value2',...)`, `smallmoney`, and `uniqueidentifier`.

enum('value1','value2',...)

The `enum('value1','value2',...)` type, defined only in MySQL, allows you to define a column that contains one of the values defined for the `enum`. For more information, see `http://dev.mysql.com/doc/refman/5.0/en/enum.html`.

money

The `money` type maps to a `decimal` (`System.Decimal`) in C# and is defined as follows:

- In SQL Server, `money` represents a range of monetary values from -922,337,203,685,477.5808 to 922,337,203,685,477.5807 with an accuracy of four decimal places. You should use the `SqlDbType.Money` type identifier for parameters of this type.
- In Microsoft Access, the `money` type is specified as a `Currency` data type and supports the same range of values as SQL Server. You should use the `OleDbType.Currency` type identifier for parameters of this type.
- There is no equivalent in MySQL.

set('value1','value2',...)

The `set('value1','value2',...)` type, defined only in MySQL, allows you to define a column that contains up to 64 of the values defined for the `set`. For more information, see `http://dev.mysql.com/doc/refman/5.0/en/set.html`.

smallmoney

The `smallmoney` type, defined only in SQL Server, maps to a `decimal` (`System.Decimal`) in C# and allows you to hold a monetary value from -214,748.3648 to 214,748.3647 with an accuracy of four decimal places. You should use the `SqlDbType.SmallMoney` type identifier for parameters of this type.

uniqueidentifier

The `uniqueidentifier` type maps to a `System.Guid` in C# and is defined as follows:

- In SQL Server, `uniqueidentifier` denotes that the field will hold a GUID. You should use the `SqlDbType.Uniqueidentifier` type identifier for parameters of this type.
- In Microsoft Access, a `uniqueidentifier` is represented as a `Number` data type and a `Replication ID` field size. You should use the `OleDbType.Guid` type identifier for parameters of this type.
- There is no equivalent in MySQL.

APPENDIX C

SQL Primer

This appendix summarizes the basic syntax for `SELECT`, `INSERT`, `UPDATE`, and `DELETE` queries. This isn't intended as a full reference, but as a recap of how to use the queries in ASP.NET, as demonstrated in the examples in this book.

In this appendix, the SQL keywords are shown in all uppercase letters. Optional elements of a query are surrounded by brackets. User-defined elements of a query are in *italics*.

Note that SQL keywords don't need to be in uppercase, and the queries don't need to be separated over many lines, as they are shown in this appendix. These conventions just make it easier to read and understand them. SQL is case-insensitive, except for cases where the database server insists that table and column names *are* case-sensitive, which depends on how your database server is configured. If you've followed the installation instructions in Appendix A, then both SQL Server 2005 and MySQL 5.0 are case-insensitive. However, you should endeavor to be consistent in your letter casing, because it will reduce the chances of problems cropping up later.

You can find Microsoft's Transact-SQL reference online at `http://msdn.microsoft.com/en-us/library/ms189826.aspx`. You can find MySQL's SQL reference online at `http://dev.mysql.com/doc/refman/5.0/en/sql-syntax.html`. Each section in this appendix includes links relevant to the specific query from these references.

Tip Two good books on SQL in general are *The Programmer's Guide to SQL* by Cristian Darie and Karlie Watson (1-59059-218-2; Apress, 2003) and *Teach Yourself SQL in 10 Minutes* by Ben Forta (0-67232-567-5; Sams, 2004). Both are good introductory guides to the subtleties of SQL not covered here.

SELECT

The purpose of a `SELECT` query is to return some information from the database. This information may be any of the following:

- A single scalar value returned as an object by a call to `ExecuteScalar()`
- A table of values returned as a single result inside a DataReader object by a call to `ExecuteReader()`
- A set of tables of values returned as multiple results inside a DataReader object

- A table returned into a `DataSet` via a DataAdapter's `Fill()` method
- A set of results returned from a `SqlDataSource`

The syntax of the `SELECT` query looks like this:

```
SELECT <select column list>
FROM <table>
  [ <join expression> ]
[ WHERE <constraints> ]
[ ORDER BY <order column list> ]
```

This query has the following five pieces:

- A *`select column list`* to be retrieved from the database. Generally, this is a comma-separated list of column names from the database, the * wildcard (meaning every column in the given table should be returned), or an aggregate function on a set of columns such as `COUNT()` or `TOP()`.
- The name of the *`table`* from which the selection should originate.
- An optional *`join expression`* determining how other tables should be linked to the information in the *`table`*. There can be as many *`join expression`* statements as needed to retrieve the required data from other tables in the database.
- An optional *`constraints`* prefixed by the `WHERE` clause that allow you to filter the data to be returned. The *`constraints`* isn't a comma-separated list. Each condition is joined by one of the three Boolean conditions `OR`, `AND`, or `NOT` and is a comparison of a column to either a literal value or another column.
- An optional comma-separated *`order column list`* indicating which columns the results of the `SELECT` query should be ordered by. By default, they're organized into ascending order. Adding the keyword `DESC` to a column in the `ORDER BY` clause will sort that column in descending order.

For example, to retrieve the names of all the family members in a genealogy database, use the following query:

```
SELECT MemberName FROM familymember
```

If you want to retrieve all the details about the dogs in the family, use the following query:

```
SELECT * FROM familymember
WHERE MemberSpecies = 'dog'
```

To retrieve the Social Security number for every member of the family born before 1987, use the following query:

```
SELECT familymember.MemberName, financialdetail.SSN
FROM familymember
  INNER JOIN financialdetail
  ON financialdetail.MemberID = familymember.MemberID
WHERE MemberBirthdate < '01/01/1987'
```

Note that some of the elements of SELECT have been left out for simplicity's sake and because they aren't covered in the book. For a complete look at SELECT, check out the following links:

- SQL Server: http://msdn.microsoft.com/en-us/library/ms189499.aspx
- MySQL: http://dev.mysql.com/doc/refman/5.0/en/select.html

INSERT

The purpose of an INSERT query is to add some new information to a table in a database. This new data must conform to the rules and constraints already laid out on the table, or an error will be returned. INSERT queries are called through a DataAdapter's Update() method, executed by calling ExecuteNonQuery() on a Command object (which returns the number of rows in the table the query has added), or called through a SqlDataSource.

The syntax of the INSERT query looks like this:

```
INSERT [INTO] <table name> [ ( <column list> ) ]
VALUES ( <column value list> )
```

This query has the following five pieces:

- The optional keyword INTO to make the query more readable.
- The *table name* that determines the table to which the information will be added.
- The optional (comma-separated) *column list* that names the columns in the new row to which you're giving values. This list must be surrounded by parentheses.
- The keyword VALUES that separates the *column list* from the *column value list*.
- The (comma-separated) *column value list* that contains a value for each of the columns in the *column list* for the new row. Each value can be a literal, an expression saying how a value is to be determined from the values of other columns (firstname + ' ' + surname, for example), the keyword DEFAULT indicating that the column should take its default value as defined in the database, or NULL. This list must be surrounded by parentheses.

The number of the items in the *column list* should equal the number of items in the *column value list* and be ordered in the same way. Thus, the first column named in the *column list* will be filled with the first value in the *column value list*, the second with the second, and so on. If a *column list* isn't supplied, the *column value list* must supply a new value for every column in the table.

For example, to add a newcomer to a family database, use the following query:

```
INSERT INTO familymember
  (MemberID, MemberName, MemberBirthdate, MemberSpecies)
VALUES (25, 'Spot', '14/04/04', 'Cat')
```

For more information about the INSERT query, try these links:

- SQL Server: http://msdn.microsoft.com/en-us/library/ms174335.aspx
- MySQL: http://dev.mysql.com/doc/refman/5.0/en/insert.html

UPDATE

The purpose of an UPDATE query is to modify some already existing information in a table in the database. UPDATE queries are called through a DataAdapter's Update() method, executed by calling ExecuteNonQuery() on a Command object (which returns the number of rows in the table the query has modified), or called through a SqlDataSource.

The syntax of the UPDATE query looks like this:

```
UPDATE <table name>
SET column1 name = expression1,
    column2 name = expression2,
         .
         .
         .
    columnM name = expressionM
[ WHERE <constraints> ]
```

This query has the following four pieces:

- The *table name* that identifies the table in which data will be updated.
- The keyword SET to denote the start of the updated information.
- A comma-separated list of assignments where individual columns are set to given values.
- An optional *constraints* prefixed by the WHERE clause that allows you to filter the data to be updated.

For example, to change a female family member's name in the family database, use the following query:

```
UPDATE familymember
SET MemberName = 'Jane Maharry'
WHERE MemberName='Jane Randall'
```

For more information about the UPDATE query, try these links:

- SQL Server: http://msdn.microsoft.com/en-us/library/ms177523.aspx
- MySQL: http://dev.mysql.com/doc/refman/5.0/en/update.html

DELETE

The purpose of a DELETE query is to remove one or more rows of information from a table in a database. DELETE queries are called through a DataAdapter's Update() method, executed by calling ExecuteNonQuery() on a Command object (which returns the number of rows in the table the query has deleted), or called through a SqlDataSource.

The syntax of the DELETE query looks like this:

```
DELETE [FROM] <table name>
[ WHERE <constraints> ]
```

This query has the following three pieces:

- The optional keyword `FROM` to make the query more readable.
- The *`table name`* that determines the table from which data will be deleted.
- An optional *`constraints`* prefixed by the `WHERE` clause that allows you to filter the data to be deleted.

For example, to remove all cats from the family database, use the following query:

```
DELETE FROM familymember
WHERE MemberSpecies = 'Cat'
```

Note that calling `DELETE` without a `WHERE` clause removes all the rows from a table but does not remove the table itself. Also, you should check whether your databases will `DELETE` data regardless of whether the `DELETE` query breaks referential integrity constraints. Some do; some don't.

For more information about the `DELETE` query, try these links:

- SQL Server: `http://msdn.microsoft.com/en-us/library/ms189835.aspx`
- MySQL: `http://dev.mysql.com/doc/refman/5.0/en/delete.html`

APPENDIX D

Sample Database Tables

This appendix contains the complete structure of and data for the four tables in the sample database used in the examples in this book. Use it in conjunction with the instructions given in Chapter 2 to build the sample database. Alternatively, use the instructions at the end of this appendix for generating the databases automatically.

Note that if you're creating a MySQL database, all tables must be InnoDB-type tables.

In SQL Server 2005 and MySQL 5.0, the sample database has a user account called `BAND` attached to it. Refer to Chapter 2 to see how to add this user account to the database.

The Manufacturer Table

The Manufacturer table contains the columns listed in Table D-1 and the data listed in Table D-2. Table D-1 also lists the properties of each column that you should set (or ensure that they're set as specified).

Table D-1. *Columns in the Manufacturer Table*

Column Name	Microsoft Access	SQL Server 2005	MySQL 5.0
ManufacturerID	DataType: Autonumber PrimaryKey: true	DataType: int PrimaryKey: true Is Identity: true Allow Nulls: false	DataType: integer PrimaryKey: true AUTO_INCREMENT: true Not Null: true
ManufacturerName	DataType: Text Size: 50 Required: Yes	DataType: varchar Length: 50 Allow Nulls: false	DataType: varchar Length: 50 Not Null: true
ManufacturerCountry	DataType: Text Size: 50 Required: No	DataType: varchar Length: 50 Allow Null: true	DataType: varchar Length: 50 Not Null: false
ManufacturerEmail	DataType: Text Size: 100 Required: No	DataType: varchar Length: 100 Allow Null: true	DataType: varchar Length: 100 Not Null: false
ManufacturerWebsite	DataType: Text Size: 100 Required: No	DataType: varchar Length: 100 Allow Null: true	DataType: varchar Length: 100 Not Null: false

Table D-2. *Data in the Manufacturer Table*

ManufacturerID	ManufacturerName	ManufacturerCountry	ManufacturerEmail	ManufacturerWebsite
1	Apple	USA	lackey@apple.com	http://www.apple.com
2	Creative	Singapore	someguy@creative.com	http://www.creative.com
3	iRiver	Korea	knockknock@iriver.com	http://www.iriver.com
4	MSI	Taiwan	hello@msicomputer.co.uk	http://www.msicomputer.co.uk
5	Rio	USA	greetings@rio.com	http://www.rio.com
6	SanDisk	USA	heyhey@sandisk.com	http://www.sandisk.com
7	Sony	Japan	hi_san@sony.co.jp	http://www.sony.com
8	Cowon	Korea	moomoo@cowon.com	http://www.cowon.com
9	Frontier Labs	Hong Kong	frontdesk@frontierlabs.com	http://www.frontierlabs.com
10	Samsung	Japan	mashimashi@samsung.co.jp	http://www.samsung.com

The Player Table

The Player table contains the columns listed in Table D-3 and the data listed in Table D-4. Table D-3 also lists the properties of each column that you should set (or ensure that they're set as specified).

Table D-3. *Columns in the Player Table*

Column Name	Microsoft Access	SQL Server 2005	MySQL 5.0
PlayerID	DataType: Autonumber PrimaryKey: true	DataType: int PrimaryKey: true Is Identity: true Allow Nulls: false	DataType: integer PrimaryKey: true AUTO_INCREMENT: true Not Null: true
PlayerName	DataType: Text Size: 50 Required: Yes	DataType: varchar Length: 50 Allow Nulls: false	DataType: varchar Length: 50 Not Null: true
PlayerManufacturerID	DataType: Number Field Size: Long Integer Required: Yes	DataType: int Allow Null: false	DataType: integer Not Null: true
PlayerCost	DataType: Number Field Size: Decimal Required: Yes	DataType: decimal Size: 10, 2 Allow Null: false	DataType: decimal Size: 10, 2 Not Null: true
PlayerStorage	DataType: Text Size: 50 Required: Yes	DataType: varchar Length: 50 Allow Null: false	DataType: varchar Length: 50 Not Null: true

Table D-4. *Data in the Player Table*

PlayerID	PlayerName	PlayerManufacturerID	PlayerCost	PlayerStorage
1	iPod Shuffle	1	99.00	Solid State
2	MuVo V200	2	96.00	Solid State
3	iFP-700 Series	3	149.00	Solid State
4	iFP-900 Series	3	199.00	Solid State
5	MegaPlayer 521	4	93.00	Solid State
6	Forge	5	119.00	Solid State
7	Digital Audio Player	6	135.00	Solid State
8	Network Walkman NW-E99	7	138.00	Solid State
9	iPod	1	209.00	Hard Disk
10	iPod Mini	1	169.00	Hard Disk
11	iPod Photo	1	309.00	Hard Disk
12	iAudio M3	8	249.00	Hard Disk
13	Zen Micro	2	138.00	Hard Disk
14	Zen Touch	2	169.00	Hard Disk
15	L1	9	149.00	Hard Disk
16	H10	3	189.00	Hard Disk
17	H300 Series	3	319.00	Hard Disk
18	Carbon	5	169.00	Hard Disk
19	Napster YH-920	10	179.00	Hard Disk
20	Network Walkman NW-HD3	7	215.00	Hard Disk

The Format Table

The Format table contains the columns listed in Table D-5 and the data listed in Table D-6. Table D-5 also lists the properties of each column that you should set (or ensure that they're set as specified).

Table D-5. *Columns in the Format Table*

Column Name	Microsoft Access	SQL Server 2005	MySQL 5.0
FormatID	DataType: Autonumber PrimaryKey: true	DataType: int PrimaryKey: true Is Identity: true Allow Nulls: false	DataType: integer PrimaryKey: true AUTO_INCREMENT: true Not Null: true
FormatName	DataType: Text Size: 10 Required: Yes	DataType: varchar Length: 10 Allow Nulls: false	DataType: varchar Length: 10 Not Null: true

Table D-6. *Data in the Format Table*

FormatID	FormatName
1	wav
2	mp3
3	aac
4	wma
5	asf
6	ogg
7	atrac
8	aiff

The WhatPlaysWhatFormat Table

The WhatPlaysWhatFormat table contains the columns listed in Table D-7 and the data listed in Table D-8. Table D-7 also lists the properties of each column that you should set (or ensure that they're set as specified).

Table D-7. *Columns in the WhatPlaysWhatFormat Table*

Column Name	Microsoft Access	SQL Server 2005	MySQL 5.0
WPWFPlayerID	DataType: Number Field Size: Long Integer PrimaryKey: true Required: Yes	DataType: int PrimaryKey: true Allow Nulls: false	DataType: integer PrimaryKey: true Not Null: true
WPWFFormatID	DataType: Number Field Size: Long Integer PrimaryKey: true Required: Yes	DataType: int PrimaryKey: true Allow Nulls: false	DataType: integer PrimaryKey: true Not Null: true

Table D-8. *Data in the WhatPlaysWhatFormat Table*

WPWFPlayerID	WPWFFormatID	WPWFPlayerID	WPWFFormatID
1	1	11	1
1	2	11	2
1	3	11	8
2	2	12	1
2	4	12	2
3	2	12	4
3	4	12	5
3	5	12	6
3	6	13	2
4	2	13	4
4	4	14	1
4	5	14	2
4	6	14	4
5	1	15	1
5	2	15	2
5	4	15	4
6	2	16	2
6	4	16	4
7	2	17	1
7	4	17	2
8	2	17	4
8	7	17	5
9	1	17	6
9	2	18	2
9	3	18	4
9	8	19	2
10	1	19	4
10	2	20	2
10	3	20	4
10	8	20	6

Using the Database Scripts

In the code download for this book (available from the Apress Web site, `http://www.apress.com`), you'll find a `scripts` folder, which contains two script files: `sqlserver.sql` and `mysql.sql`. You can use these to automatically build the databases in SQL Server 2005 and MySQL 5.0, as described in the following sections.

The code download also includes a Microsoft Access database, `players.mdb`, in the root of the download. You can use this immediately as the data source for your pages.

■ **Caution** By running these scripts, you erase any changes made to the contents of the database since its creation. Therefore, save any changes you may want to keep before running these scripts.

Using a Script to Build the SQL Server 2005 Database

To refresh the SQL Server 2005 database, follow these steps:

1. Open SQL Server Management Studio and cancel any attempt to connect to an existing database.
2. Select File ➤ Open ➤ File and select the `sqlserver.sql` file from the `scripts` folder download (from wherever you've saved the files onto your machine).
3. When the Connect dialog box appears, connect to the `localhost\BAND` database server using the `sa` account (which, if you've followed the setup instructions in Appendix A, will have a password of bandpass).
4. Click the Execute button on the toolbar to run the script and refresh the database.
5. If this is a database refresh, you may receive the error shown in Figure D-1, but don't worry about it. The script is designed to build the database completely, and as you already have the user account created (either manually or from the script previously), you can't create it again.

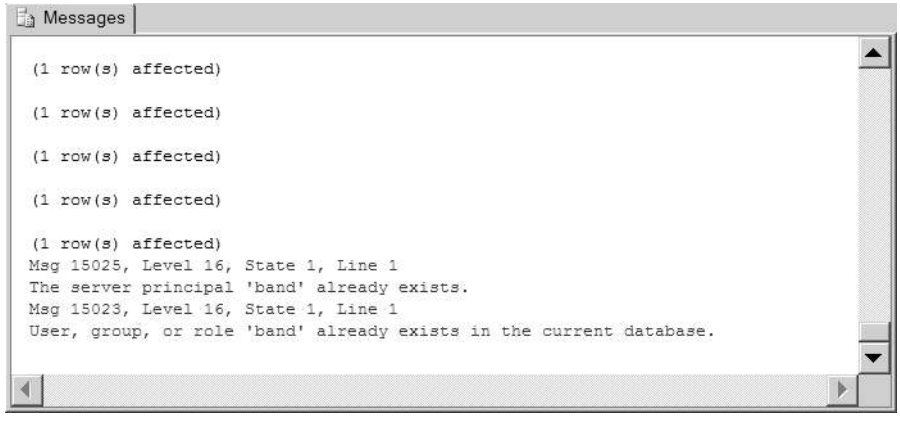

Figure D-1. *You can't re-create the user login if it already exists.*

Using a Script to Build the MySQL 5.0 Database

To refresh the MySQL 5.0 database follow these steps:

1. Open MySQL Query Browser and connect to the `localhost` database server using the `root` account (which, if you've followed the setup instructions in Appendix A, will have a password of bandpass).
2. Select File ➤ Open Script and select the `mysql.sql` file from the `scripts` folder download (from wherever you've saved the files onto your machine).
3. Click the Execute button on the toolbar to run the script and refresh the database.