CHAPTER 6

Accessing Databases from Visual Basic

In this chapter

- ODBC and OLE DB
- DAO object model
- RDO object model
- ADO object model
- Visual Basic database tools

In this chapter, I'm going to discuss how Visual Basic talks to a database server. ODBC and OLE DB are the low-level technologies that actually perform the work, while the DAO, RDO, and ADO object models encapsulate these technologies to make them easy for the Visual Basic programmer to use.

Microsoft Database Programming APIs

Just because your database server is based on SQL doesn't mean that you can easily access it from your favorite programming language. Many database vendors provide a special pre-compiler that translates embedded database statements into database subroutine calls that in turn communicate with the database server. The problem with this approach is that you need a different pre-compiler for each programming language the database vendor supports. Of course, since the pre-compilers are specific for each database server, you'll need a pre-compiler for each database server for which you develop programs. Rather than developing a large number of pre-compilers for each combination of database server and programming language, Microsoft developed a standard called *Open Database Connectivity* (ODBC), which later evolved into OLE DB.

ODBC

The ODBC standard defines an *Application Programming Interface* (API) for database programming. This allows you to write a program using standard subroutine calls for any database server that supports ODBC, making it possible for the same object code to access any ODBC-compatible database server.

ODBC architecture

ODBC is based on the idea that the calls to the API routines made by the application program are translated to lower calls that are passed onto a database driver (see Figure 6-1). The database driver in turn performs the necessary work to talk to the database server. Thus, the database vendor need only provide an ODBC-compatible driver for each client computer system and not for each compiler on each client operating system.

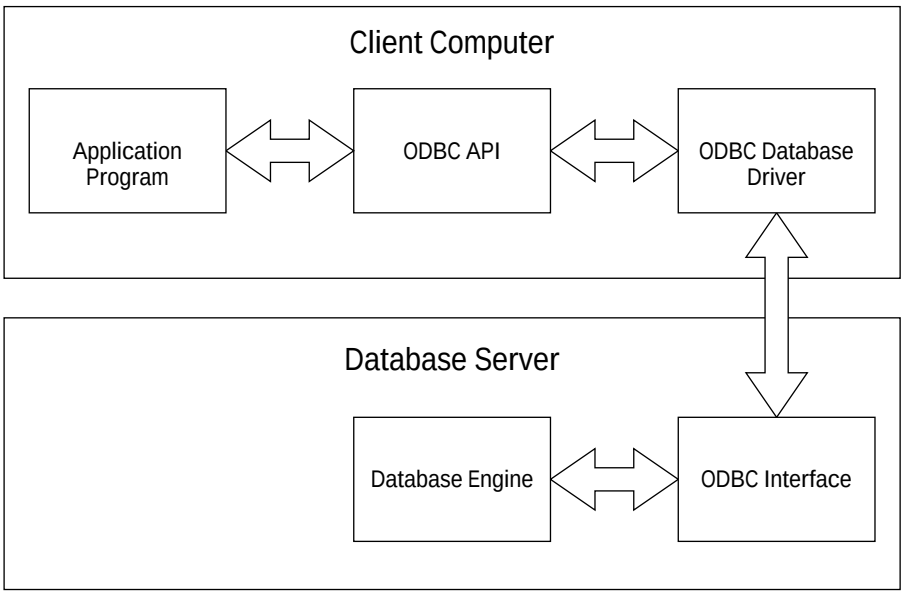

Figure 6-1: The client side of the ODBC architecture is based on the driver model.

On the database server side, all the database vendor needs to do is provide a single ODBC interface, which will be shared by all ODBC clients. This means that database vendors can preserve their native interface, plus any other specialized interfaces for other applications.

A utility program called the ODBC Administrator is included as part of the operating system to manage the set of ODBC drivers and database servers that are available to the ODBC API. This program allows you to add and remove ODBC drivers, specify how to connect to the database server, and include security information that will be used when the connection between the client computer and the database server is opened.

Drawbacks to ODBC

While the architecture of ODBC allows a great deal of flexibility on the part of the database vendor, there are several drawbacks to writing ODBC applications. First, the ODBC APIs are difficult to use, especially if you aren't programming in C. Second, the ODBC APIs are often slower than the native database interface. Third, the ODBC API often imposes restrictions on the SQL statements that can be used.

While the first drawback applies to all database servers, the second two drawbacks apply mainly to non-Microsoft database servers. Microsoft uses ODBC as the native interface to both the Jet database and SQL Server. They spent a lot of time tuning the interface for optimal results. While other database vendors support ODBC, their native interfaces may offer improved performance and functionality, especially for non-Microsoft compilers.

Database Access Objects (DAO)

Because the ODBC API is difficult to use, Microsoft chose to build an object-oriented interface to ODBC called *Data Access Objects* (DAO). DAO was originally developed for Access and Visual Basic programmers who needed to access the Jet database. Most of the functionality available in DAO mirrors features found in the Jet database.

While DAO can be used to access other databases, the process is often difficult to use and clumsy in its implementation. To access an SQL Server database, you need to create a Jet database and then go through the Jet database to access the remote database. While DAO allows you to access the ODBC database directly, each time you open the database, you must download a lot of information about the database structures that you might access. By using the Jet database, this information is saved locally and need not be downloaded each time you open the database.

Visual Basic 6 and Access 2000: While Jet 3.51 is shipped with Visual Basic 6, Jet 4.0 is shipped with Access 2000. While it's possible to use Visual Basic 6 and DAO with Jet 4.0, many of the new features found in Jet 4.0 can only be used if you're using ADO.

Remote Database Objects (RDO)

The *Remote Database Objects* (RDO) object model was built to address the problems associated with accessing SQL Server from Visual Basic. First available with the Visual Basic 5 Enterprise Edition of Visual Basic, RDO offered a streamlined object model which in turn offered much better performance than DAO when used with a database such as SQL Server or Oracle. However, RDO doesn't work with the Jet database and other single-user database systems.

One feature lacking in RDO that is found in DAO is the ability to create a database by defining objects. Given that RDO is really targeted at large installations, this isn't nearly as limiting as it seems. These installations generally have a group of people known as *database administrators* (DBAs), whose sole responsibility is the creation and maintenance of database structures. DBAs often use specialized tools to help them design and document their databases.

OLE DB

After years of working with ODBC, Microsoft recognized that the database server to database client model used by ODBC could be generalized as a *data provider* to *data consumer* model (see Figure 6-2).

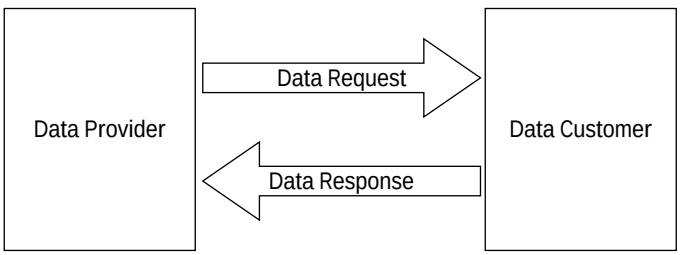

Figure 6-2: A data provider handles data requests from a data consumer.

Data providers

A *data provider* is a program that supplies data to another program. In the ODBC model, the data provider is the database server. However, in the OLE DB, nearly anything that can produce data could be considered a data provider. This allows programmers to treat such things as an Excel workbook, a flat file, or another custom-built program like a database server.

Data consumers

A *data consumer* is simply a client program that requests data from a data provider. Two types of information may be requested from the data producer: *data* and *metadata*. Data is the information used by an application, while metadata is information about the structures used to hold the data. Metadata includes such information as the name of each column returned, its size and data type, and other descriptive information.

ActiveX Data Objects (ADO)

The *ActiveX Data Objects* (ADO) object model was created to replace both DAO and RDO object models and was shipped for the first time in Visual Basic 6. It uses the newer OLE DB API rather than the older ODBC API. In addition, ADO has an even simpler object model than found in the DAO and RDO, which makes it easier to use.

ADO is modeled after RDO and includes many of the familiar objects found in DAO and RDO, such as the **RecordSet** object. However, where RDO maintains a relatively strict object hierarchy, ADO's object structure is much looser. This means that you generally need fewer objects in your program, and the resulting code is even simpler.

Like RDO, ADO doesn't include objects that you can use to define your database. However, with version 2.5 of ADO, Microsoft included a new set of objects known as ADOX, which can be used to define database structures and security. These objects would be useful if you want to write a general-purpose tool that allows you to create and manipulate database structures from multiple database vendors. They can help smooth over differences in how the various vendors implement their own SQL extensions.

ADO MD is another set of extensions to the base ADO objects. These objects allow you to work with a multidimensional database such as the OLAP Services bundled with SQL Server. You can use these objects to build your own programs for manipulating information from your own data warehouse.

OLE DB providers

Table 6-1 lists the OLE DB providers that are available with Visual Basic 6. Other OLE DB providers may be available from your specific database vendor.

Table 6-1
Common OLE DB Providers

Provider Name	*Description*
Microsoft Jet 3.51 OLE DB Provider	Supports Access 97/Jet 3.51 databases
Microsoft Jet 4.0 OLE DB Provider	Supports Access 2000/Jet 4.0 databases
Microsoft OLE DB Provider for SQL Server	Supports SQL Server 6.5 and 7.0
Microsoft OLE DB Provider for Oracle	Supports Oracle 7 and 8
Microsoft OLE DB Provider for Microsoft Active Directory Service	Supports Microsoft Active Directory Service found in Windows 2000 Server
Microsoft OLE DB Provider for Microsoft Index Service	Supports Microsoft Index Server 2.0 and newer
Microsoft OLE DB Provider for OLAP Services	Supports Microsoft OLAP Server
Microsoft OLE DB Provider for ODBC Drivers	Supports generic ODBC access

Even if your database server doesn't support OLE DB, that doesn't mean you can't access the database from ADO. There is a special provider known OLE DB Provider for ODBC. This allows you to connect your ADO-based application to any ODBC database. Of course, you should try to use a native OLE DB provider whenever possible.

Custom OLE DB providers

The architecture of OLE DB allows you to write your own OLE DB provider. This is an advanced technique that most people won't ever need to use. After all, how many people implement their own database management system? However, if you have a custom data storage application, you can create your own OLE DB provider for it. This will allow programmers to access it using tools with which they are already familiar.

Visual Basic Database Tools

To support database programming, Visual Basic includes a nice assortment of tools that will assist you in creating your application, including the following:

- ✦ Data Environment Designer
- ✦ Data View Window

- ✦ Database Designer
- ✦ SQL Editor
- ✦ T-SQL Debugger
- ✦ Query Designer
- ✦ Data Report Designer
- ✦ UserConnection Designer

With the exception of the UserConnection Designer, these tools will work only in an ADO programming environment. The UserConnection Designer works only with RDO objects.

Data Environment Designer

The Data Environment Designer (see Figure 6-3) is an interactive design-time tool that helps you create database objects for use at runtime. It is based on the ADO object model and will not work with RDO and DAO objects.

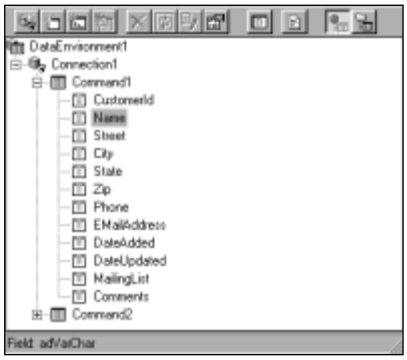

Figure 6-3: Using the Data Environment Designer

You can use the Data Environment Designer to perform the following tasks:

- ✦ Define **Connection** objects to your database using either OLE DB data sources or ODBC data sources.
- ✦ Define **Command** objects using tables, queries, and stored procedures from your database.
- ✦ Create hierarchies of **Command** objects that can be used with hierarchical tools like the **HFlexGrid** control and the **Data Reporter Designer**.

- ✦ Drag and drop fields and tables from a **Command** object in the designer onto a **Form** object or the **Data Reporter** designer that are automatically bound to the **Command** object.
- ✦ Specify default the type of control to be used as part of drag and drop operations.
- ✦ Bind data-aware controls to **Field** objects within a **Command** object.
- ✦ Attach code for **Connection** and **Recordset** objects in the Data Environment Designer.
- ✦ Trap all ADO events for the **Connection** and **Command** objects.

Data View Window

The Data View Window allows you to access your database system through Visual Basic instead of a particular database vendor's utility (see Figure 6-4). Through the Data View Window, you can access the Microsoft Visual Database Tools.

The Visual Database Tools are present only in the Enterprise Edition of Visual Basic. This not only includes the Data View Window, but the Database Designer, the Query Designer, the SQL Editor, and the T-SQL Debugger.

Figure 6-4: Using the Data View Window to browse and edit the database structures in your database

You can use the Data View Window to perform the following tasks:

- ✦ Create a **Connection** object that can be used to access your database while in design mode or at runtime.
- ✦ Design your database graphically using the Database Designer.

- ✦ View the contents of a table or view using a worksheet-like display.
- ✦ Create and edit stored procedures using the SQL Editor.
- ✦ Debug stored procedures using the T-SQL Debugger.
- ✦ Create views using the Query Designer.
- ✦ Generate reports using the Data Reporter Designer.

Database Designer

The Database Designer is a graphical tool that presents a database using a graphical diagram (see Figure 6-5). The Database Designer works with both Microsoft SQL Server and Oracle database systems.

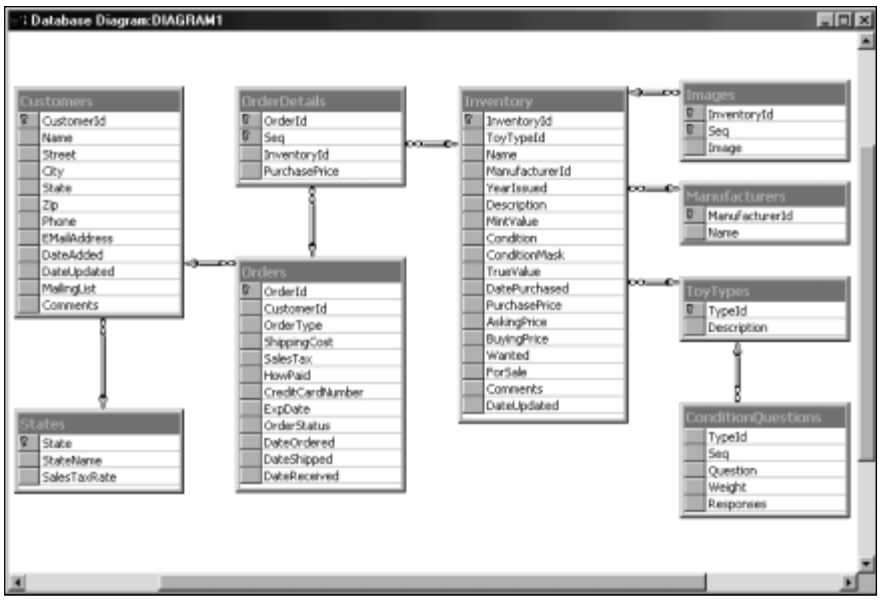

Figure 6-5: Designing a database graphically using the Database Designer

You can use the Database Designer to perform the following tasks:

- ✦ Create and modify tables.
- ✦ Add and delete indexes.
- ✦ Define relationships between tables.
- ✦ Save SQL scripts to create or update your database structures.

SQL Editor

The SQL Editor provides a simple editor to help you write stored procedures (see Figure 6-6).

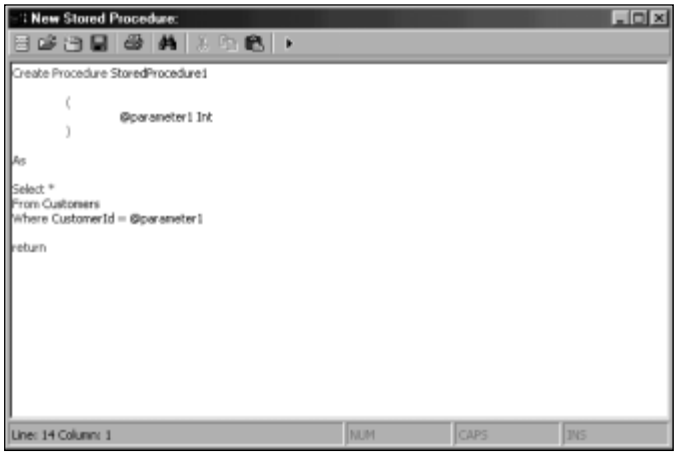

Figure 6-6: Editing a simple stored procedure using the SQL Editor

You can use the SQL Editor to perform the following tasks:

- ✦ Create and edit SQL stored procedures.
- ✦ Execute stored procedures.
- ✦ Use the T-SQL Debugger to debug your stored procedures.

T-SQL Debugger

The T-SQL Debugger helps you test and debug your stored procedures (see Figure 6-7).

You can use the T-SQL Debugger to perform the following tasks:

- ✦ Display the contents of local variables and parameters.
- ✦ Modify local variables and parameters while executing the stored procedure.
- ✦ Control execution by using breakpoints.
- ✦ Step through the stored procedure.
- ✦ View global variables.
- ✦ View the call stack.

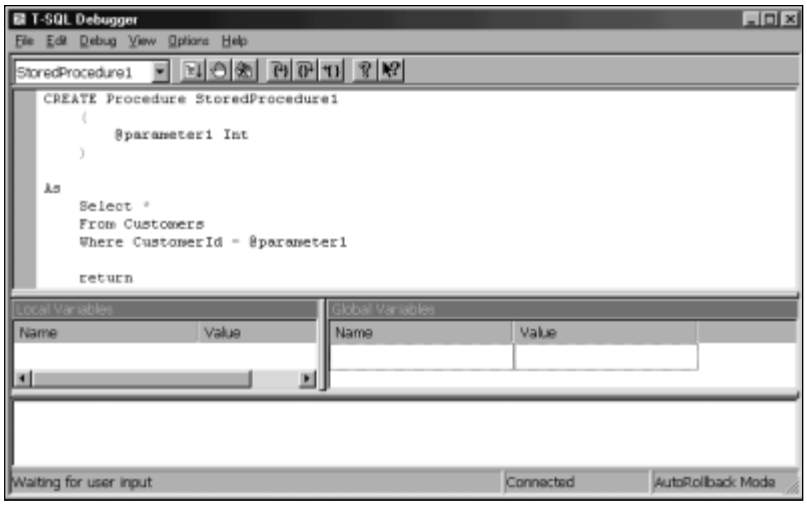

Figure 6-7: Testing a simple stored procedure using the T-SQL Debugger

Query Designer

The Query Designer is a tool that works with the Data View Window to drag and drop tables and columns to create a view or query (see Figure 6-8).

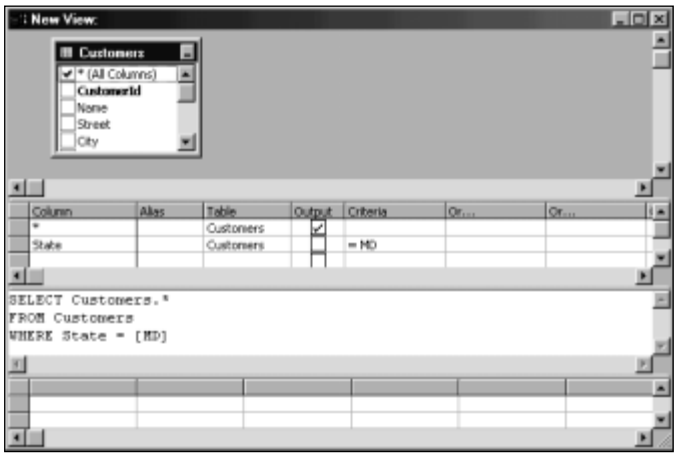

Figure 6-8: Creating a view using the Query Designer

You can use the Query Designer to perform the following tasks:

- ✦ Drag and drop tables from the Data View Window onto your query.
- ✦ Use a criteria grid to specify search criteria and sort order.

- ✦ Select the columns to be output as a result of the query.
- ✦ Browse and edit live views of your data in your view.

Data Reporter Designer

The Data Reporter Designer helps you build reports interactively (see Figure 6-9).

Figure 6-9: Building a report using the Data Reporter Designer

You can use the Data Reporter Designer to perform the following tasks:

- ✦ Create reports by dragging and dropping fields from the Data View Window.
- ✦ Add controls to your report, just like you add controls to a Form in a Visual Basic program.
- ✦ Implement a Print Preview function in your program to allow users to preview the report before sending it to the printer.
- ✦ Allow the user to resume working while the report is running (i.e., asynchronous execution of the report).

UserConnection Designer

The UserConnection Designer is a tool that assists the RDO programmer in building connection (**rdoConnection**) and query (**rdoQuery**) objects. These objects are made available at the project level and can be used anywhere in your program. A key part of these objects is that they have a simplified method for responding to database events. These objects also make it easier to call stored procedures at runtime.

RDO not ADO: The UserConnection Designer supports only the RDO object model. Do not try to use it with DAO or ADO object models. The Data Environment Designer is a much improved version of the UserConnection Designer and is available for use with ADO programs.

Thoughts on Visual Basic

Visual Basic programmers can choose from three different object models to access their databases: DAO, RDO, and ADO. Deciding which model is best for you may seem difficult, but it really shouldn't be. ADO is the model that Microsoft would prefer you to use. It is where they are placing the majority of their efforts and it is where you will see the most enhancements in the future.

Microsoft isn't going to abandon the DAO object model anytime soon. There are too many Visual Basic programs that still use it. Dropping support for DAO in a future release of Visual Basic would simply mean that many Visual Basic programmers wouldn't upgrade. (Consider the number of people still using Visual Basic 3 and 4 because Microsoft stopped supporting 16-bit versions with Visual Basic 5.) So if you're currently using DAO, you can continue to use it in the foreseeable future. However, if you're not currently using DAO, you probably shouldn't start.

Directly accessing SQL Server databases using DAO isn't really practical for large applications. As more and more people tried to do this, Microsoft developed RDO as a solution. Its low overhead offered much better performance in large-scale applications. However, RDO's time is also past, and if you're not using it now, don't start.

ADO is the wave of the future. You should be using it for new applications wherever practical. While you don't have to drop everything you're currently doing to convert your existing applications to ADO now, you should have a plan in place that ensures that they will all be converted over time. This is especially true in applications with high transaction rates or applications that are offering services over the Internet.

In the former case, many of the things that Microsoft will do to improve performance will be tied to ADO. This includes facilities such as the COM+ Transaction Server and future versions of SQL Server. In the latter case, Microsoft is already adding features, such as XML support to ADO, which will not be added to DAO and RDO. Also, all of the tools designed to make you more productive, such as the Data View Window and the Data Environment Designer, also require an ADO-based program.

Summary

In this chapter you learned that:

- ODBC was originally developed as an alternative to developing pre-compilers for each programming language to translate SQL statements into executable code.
- DAO was created to provide an object-oriented API to access Microsoft Jet as well as any ODBC database.
- RDO is a low-overhead, general-purpose ODBC interface.
- OLE DB is a general-purpose interface that allows data consumers to talk to data producers.

- ADO is the recommended way for Visual Basic programmers to access OLE DB.
- Visual Basic includes a number of specialized tools for the database programmer, which include the Data Environment Designer, Data View Window, Database Designer, SQL Editor, T-SQL Debugger, Query Designer, Data Reporter Designer, and the UserConnection Designer.

✦ ✦ ✦