CHAPTER 29

Overview of Microsoft Jet

In this chapter, I'll discuss the Microsoft Jet database system. I'll explain its architecture, data types, and security model. I'll also cover how to use ADO with Jet, including how to build a connection string and use the special facilities built into Visual Basic to support designing Jet databases.

In This Chapter

Introducing Microsoft Jet

Understanding the Database architecture

Examining the security model

Connecting to Jet with ADO

Overview of Jet

There are perhaps more copies of Microsoft Jet installed on PCs than any other database engine, due to the fact that Jet is a critical part of Microsoft Access, which is part of the Microsoft Office family. This is the ideal choice when you want to build a single-user database system or share your database on a file server with a small group of people.

Microsoft Jet contains the following key features:

- Standard part of Visual Basic Professional Edition and Enterprise Edition
- Limited version included in the Visual Basic Learning Edition.
- Can be included with your Visual Basic application at no charge
- Requires little or no maintenance to maintain
- Doesn't require a dedicated database server
- Compatible with Microsoft Access
- Supports both ADO and DAO object models

Jet and Access or Access and Jet: Since the Jet database is so tightly coupled to the Access application, many people use the terms interchangeably. For instance, the Visual Data Manager tool refers to Access databases rather than Jet databases. I will refer to the database itself as Jet and the application development tool as Access, unless I'm referring to a menu item or command that specifically refers to Access rather than Jet.

Microsoft Jet versions

Microsoft Jet is just the database engine component from the Microsoft Access database system. Jet became available for the first time in 1992, when Access 1.0 was released. When Visual Basic Version 3.0 was released in 1993, Jet version 1.1 was included. Each new version of Access resulted in a new version of the Jet database engine. When a new version of Visual Basic was ready for release, it would include the latest version of the Jet database, with a few minor improvements.

Because the same database engine is used for both Access and Visual Basic, you can build an application that includes components in Access and in Visual Basic. Of course, you will need to use compatible versions of the software. In other words, don't expect to use an Access 1.0 database with Visual Basic 6.

Visual Basic 6 includes version 3.51 of the Jet database engine, which is basically the same as the version 3.5 that was shipped with Access 97. Access 2000 includes version 4.0 of the Jet engine, which can also be used with Visual Basic. These are the two most common versions of the Jet engine you are likely to encounter as a Visual Basic programmer.

The tools don't work: The tools included with Visual Basic 6 only work on version 3.5 and earlier Jet databases. If you plan to use a version 4.0 database, you must have Access 2000 installed on your system or be prepared to write your own code using ADOX or DAO to administrator your database.Jet version 3.5

If you develop VB6 applications with Jet, you are most likely using version 3.51. This is a very stable release, and it works well with Access 97. It is significantly faster than previous versions of the Jet engine and designed to be a little bit more friendly when accessing a database stored on a file server.

Key features of Jet 3.5 include:

- Support for the most common SQL statements, such as **Select**
- Database kept in a single file
- Ability to secure information based on user name and security group

- ✦ Ability to link to remote tables and treat them as local tables
- ✦ Database replication with other database servers
- ✦ Page-level locking

Jet version 4.0

Access 2000 introduced a newly revised Jet database system. Most of the changes made to Jet between Version 3.5 and 4.0 are transparent to the typical user. These changes were made to improve Access's integration with other Office 2000 applications. However, some improvements were made in the database engine itself:

- ✦ Support for Unicode data
- ✦ Record-level locking
- ✦ Migration utility to convert version 3.5 and earlier databases to the version 4.0 format

Bigger is not always better: If you convert a database from Jet 3.5 to Jet 4.0 and the database contains a lot of character data, expect the size to double. This is due to the fact that the actual data in the database is stored using Unicode characters. So, even if you try to store simple ASCII characters, the Jet database will store them using their Unicode equivalents.

Jet and DAO

Jet was designed around using the *DAO* (Data Access Objects) to access the contents of the database. DAO is a highly-structured object model that has evolved alongside Jet since its beginnings. In many ways, it is difficult to view the Jet database engine without thinking of the DAO object library.

See Chapter 6, "Accessing Databases from Visual Basic," for explanations of the DAO, ADO, and RDO models.

There are more programs today that use DAO, rather than ADO, to access Jet databases. However, I recommend using ADO over DAO for two reasons. First the ADO object library offers a simpler interface to your database, which translates into less code needed to do the same work. Second, Microsoft has decided that ADO is the way everyone should access databases using Microsoft development tools such as Visual Basic. While this doesn't mean that DAO is DOA, it does mean that its days are numbered. Access 2000 already supports ADO access to the Jet database, which incidentally allows you to use Access 2000 as a development tool for SQL Server databases. Also, Microsoft has developed a client computer friendly version of SQL Server called the Microsoft Database Engine (MSDE), which is 100% upwards compatible with SQL Server — something that Jet is not.

Jet 4.0, DAO 3.6: Even though the Jet database engine has been upgraded to version 4.0, the Data Access Objects (DAO) most often associated with the Jet database have been enhanced to version 3.6.

Jet utilities

For the most part, there are very few utilities associated with the Jet database, other than Access of course. Visual Basic includes a tool called the Visual Data Manager. This is an add-in tool that allows you to design Jet databases. (I'll talk more about the Visual Data Manager in Chapter 30).

Understanding the Database Architecture

Microsoft Jet is designed to be an SQL-based database management system that supports a handful of concurrent users. It works especially well with only a single user. While it can easily be used with Visual Basic applications, much of the architecture revolves around supporting Access applications.

Unlike most database systems, Microsoft Jet relies on shared access to a single file to coordinate activities among multiple users. This means that the code that manages the database must run in each user's address space and that you can't buffer data in memory, because different programs can't share their memory. All sharing must be done at the file level, using file locking and other techniques. This really limits performance and means that you probably won't even get close to Jet's 255 concurrent user limit.

.MDB files

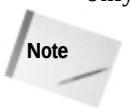
All of the information for the Jet database is stored in a single file, called the `.MDB` file. This file contains all of the data, including tables and indexes, that comprise the Jet database. In addition, facilities are included that permit Access to store information in the same file without storing it in regular database tables.

Data is stored in an `.MDB` file using different types of pages. The first page in the file is the database header page, which contains general information about the database, plus information that tracks user operations in a multi-user environment. Table pages are used to hold information stored in a table, while index pages hold index information. Long value pages keep long values such as Text and Memo fields, which can exceed the size of a normal page. Finally, system table information is stored in its own unique type of page.

There are four system tables:

- **MSysACES** holds security permissions.
- **MSysObjects** holds information about each database object, such as a table or index, plus information on Access objects, such as forms and reports.
- **MSysQueries** holds information about predefined queries in the database.
- **MSysRelationships** holds information about predefined relationships between tables in the databases.

In addition to these tables, other tables will store information about objects used only in Access applications.



You didn't hear this from me: Microsoft considers the system tables in Access to be undocumented. This basically means that Microsoft reserves the right to change the organization and number of system tables without notifying anyone. Therefore, you shouldn't attempt to use these tables as part of your application.

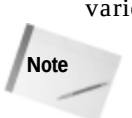
.LDB files

When someone opens an `.MDB` file for the first time, an `.LDB` file is created in the same directory as the `.MDB` file. After the last user closes the database, the `.LDB` file will automatically be deleted. This file contains locking information for the `.MDB` file. Each user that has a connection to the database has an entry in the `.LDB` file. This entry contains the user name and the name of the computer that is being used.

When the user locks a piece of data in the `.MDB` file, an extended byte range lock is placed on the `.LDB` file. With a little creativity, this type of lock provides both shared and exclusive access to data in the `.MDB` file without the overhead of using physical locks on the `.MDB` file itself.

.MDW Files

The security information for a Jet database is stored in an `.MDW` file, which is also known as a workgroup file. This file contains information about users and groups, including their login passwords. Once a user is validated, they will be assigned an internal security identifier (SID). The permissions that associate the SID with the various database objects are stored inside the `.MDB` file.



MDA or MDW: Depending on the version of Access you have installed, the `SYSTEM.MDA` file may also be known as `SYSTEM.MDW`.

Database objects

As you might expect, a Jet database can hold many of the same database objects that are found in other database management systems. However, due to the nature of the target audience, some of the database objects you wish to use may not be available. While tables and indexes are present in Jet, views and triggers are not.

Key database objects include:

- **Tables**, which store user data in fields.
- **Indexes**, which provide quick ways to locate information in a table.
- **Relationships**, which are used to define relationships between tables and to enforce referential integrity.
- **Queries**, which are used to define standard queries (similar to views) against the database.
- **Users and Groups**, which are used to associate individuals with database permissions.

Linked databases

One of the nicer facilities in Jet is the ability to create a link to a table in another database in such a way that the programmer thinks the table is local to the Jet database. Besides linking to other Jet databases, you can also link to these databases:

- dBase
- FoxPro
- Paradox
- Excel
- Exchange and Outlook data files (read access only)
- Some text file formats, including comma-separated value files
- Any ODBC-compliant database, such as SQL Server or Oracle8

Link-be-gone: In the past, the only way to access a remote database was to create a link using a Jet database. With Visual Basic 6, both DAO and ADO make this feature unnecessary. You can now access any ODBC-compatible database directly with DAO and the ODBCDirect interface, or you can use the more flexible OLE DB with ADO. (Of course, you can always use RDO, but ADO makes that tool even more obsolete than DAO).

Database capacities

Table 29-1 lists some of the key capacities and limitations of the Jet database.

Table 29-1
Database Capacities

Item	*Capacity*
Maximum database size	1 gigabyte
Maximum characters per object name	64
Maximum characters per user name	20
Maximum characters per user password	14
Maximum characters per database password	20
Maximum number of concurrent users	255
Maximum characters in a table name	64
Maximum characters in a field name	64
Maximum table size	1 gigabyte
Maximum number of fields per table	255
Maximum number of indexes per table	32
Maximum number of fields in an index	10
Maximum number of fields in an **Order By** clause	10

Jet data types

Jet supports a wide variety of data types, as shown in Table 29-2.

Connecting to Jet with ADO

In order to connect to a Jet database using ADO, you can use the following connection string:

```
Provider=MSDAORA.1; Data source=vb6db.Athena.justpc.net
```

Table 29-2
Jet Data Types

Jet Data Type	*Visual Basic Data Type*	*Comments*
`Boolean`	`Boolean`	**Null** values can't be used with a `Bit` field
`Byte`	`Byte`	Holds numbers from 0 to 255
`Currency`	`Currency`	Eight-byte scaled integer accurate to four decimal places
`Date`	`Date`	A date and time value ranging from 1 Jan 100 to 31 Dec 9999
`Double`	`Double`	A 64-bit floating-point number
`Integer`	`Integer`	Holds numbers from –32,768 to +32,767 (16-bit integer)
`Long`	`Long`	Holds numbers from –2,147,483,648 to +2,147,483,647 (32-bit integer)
`Memo`	`String`	A text field containing up to 1.2 billion characters
`OLE Object`	`Byte Array`	Holds pictures and other large, raw binary values up to 1.2 gigabytes in length
`Single`	`Single`	A 32-bit floating-point number
`Text`	`String`	Variable length string up to 255 characters in length

If you specified user-level security in your database, you can also include the `User ID=` and `Password=` keywords if you want to include that information as part of the connection string — or you can supply them as arguments to the `Open` method.

Of course, you can use the Data Link Properties window to configure the Jet Provider. Select the provider that corresponds to the version of Jet you are using (see Figure 29-1). After pressing the Next button, enter the name of your database, and the user name and password needed to access the database (see Figure 29-2). If you haven't implemented user-level security, you can leave the default values for user name (`Admin`) and password (Blank password). To test the connection, press the Test Connection button.

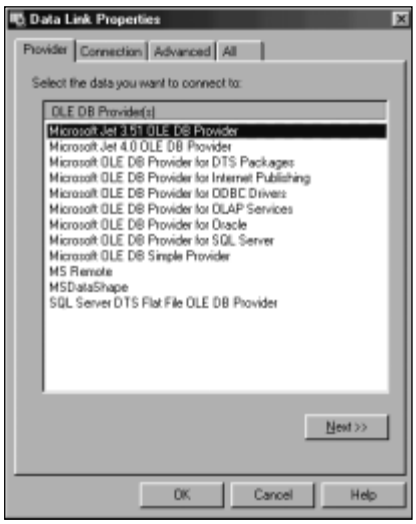


Figure 29-1: Choosing the proper OLE DB Jet provider.

Figure 29-2: Specifying the name of the database, user name, and password information.

If you specified a database password, you can specify that value on the All tab of the Data Link Properties window. Simply select `Jet OLEDB: Database Password` field, press the Edit Value button, and enter the appropriate value (see Figure 29-3).

Figure 29-3: Specifying a database password.

Jet security

Like other database systems, Jet incorporates its own security subsystem. While you can choose to use it or ignore it, don't ignore it if you plan to let more than one person access your database. In fact, you should use it even if only one person will use your database. It just might stop them from corrupting the database by preventing them from opening the database with any other application.

Share-level security

In share-level security, the database is assigned a password. Anyone that knows the password can access the database. While this isn't as secure as user-level security (discussed next), it will prevent a user from accessing your database with Excel or Access. You can set the password using Access or by using the DAO `CreateDatabase` method in the Workspace object. Simply append the string `;pwd=MyPassword` to the end of the `Locale` parameter to set the database password to `MyPassword`. You then specify the password in the `Connect` parameter of the DAO `OpenDatabase` method. In ADO, you simply specify the password as part of the `Connection` string by using the `;pwd=MyPassword` keyword.

User-level security

User-level security works like the security systems in SQL Server and Oracle8i. A userID and password are created for each person who accesses the database. Then the userID is granted permission to access the various objects in the database. Groups can be used with user-level security to simplify administration. You can create a group and assign permissions to it, just as if it was another user. Then you can assign groups to various userIDs. When users log on to the database, they will inherit all of the permissions associated with the group or groups that have been assigned to them, plus any individual permissions they may have been granted.

Finding the workgroup file

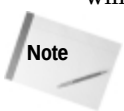
Security information is kept in a workgroup file (`.MDW`). Only one workgroup file is required per system, and the following windows registry key will be used to locate it:

```
HKEY_LOCAL_SYSTEM\Software\Microsoft\Office\8.0\Access\Jet\3.5\
Engine\SystemDB
```

A typical value for this key is:

```
C:\WINDOWS\SYSTEM\system.mdw
```

Of course, if you are using a Jet 4.0 database with Access 2000 (Version 9.0), you will need to adjust the registry key to include the appropriate version numbers.



Security with Access: In order to use security with a Jet database, you must have installed Access 97 (or Access 2000 for a Jet 4.0 database). The file `SYSTEM.MDW` will automatically be installed.

SIDs, user ids, personal ids, and passwords

All information inside a database file is secured on the basis of a Security ID, or SID. A SID is derived by encrypting the user id and a special value known as a personal id. If you create a user id and don't bother to assign a personal id, it is possible for someone else to create another workgroup file with the same user id and their own password and user id to gain access to database resources for which they do not have permission.

The personal id is used to ensure that the SID is unique and can't be duplicated, so you should always choose a value that is hard to duplicate. Simply choosing someone's first name as the user id and their last name for the personal id is not a good idea.

By default, the user id Admin exists in all database systems with the same personal id. This means that Admin has the same SID no matter which computer they use. The same is true for the Users group. However, this is not true for the Admins group. The SID associated with the Admins group is always unique for each workgroup file.

Since anyone in the Admins group has access to the entire database, the first thing you want to do is create a new user id and assign it to the Admins group. Then you should remove the Admin user id from the Admins group to ensure that someone doesn't use their own workgroup file to override your database's security.

Permissions, SIDs, and ownership

The workgroup file exists simply to return a SID for a particular user and another SID for each group that the user is a member of. This value is used as a key into some of the system tables inside the database file to determine which permissions the user has.

If the user has overlapping permissions from multiple groups and/or explicit permission assignments, the least restrictive permission will apply. Thus, if a user has read and write permission to the Customers table via the Clerks group and has been explicitly granted read permission to the Customers table, the user will have read/write access to the Customers table.

Whenever an object is created in the database, the user id of the individual that created the object is used as the object's owner. The object's owner always has access to the object. You can and should change the ownership of the objects in your database to a user id other than Admin to prevent a security problem. I suggest that you use the Admins group as the object's owner, which means that anyone in the Admins group can always access the object.

Thoughts on Microsoft Jet

Microsoft Jet is a low-end database system designed primarily for use with Microsoft Access. Much of the design revolves around features that are important to Access users, while features that might be useful to Visual Basic programmers aren't present. Jet is not upwards-compatible with SQL Server and most other databases, due to these limitations.

Jet is also limited to a maximum of 255 concurrent users. However, its true maximum is far less, depending on how frequently the database is accessed. While many improvements have been made to make Jet more efficient, it is still limited by its file-oriented approach to synchronization for multiple users.

Just because Jet isn't perfect doesn't mean you should avoid using it. The Jet database engine is a standard part of the Visual Basic Professional Edition. This means that you don't have to purchase it separately. Also, there aren't any licensing charges associated with Jet.

This makes it ideal for those applications that are cost-sensitive, such as shareware, or part of a large distributed application where you want to cache data on the local computer.

Jet has also been around for a long time, though the ability to access it using ADO hasn't. Until recently, DAO included the ability to create database objects through its object model, while ADO could only create a database object by executing the appropriate SQL statement using the `Command` object. Since Visual Basic 6 was released, ADO has gone from version 2.0 to 2.1, and now to 2.5. The newer versions of ADO now include an independent object library known as ADOX, or ADO Extensions for Data Definition Language and Security. These objects improve on DAO's ability to create database objects while using an OLE DB interface to the database. So if you need to build database objects directly from your code, you no longer have to use DAO.

Summary

In this chapter you learned:

- about the various versions of the Jet database system.
- that the tools in Visual Basic do not support Jet 4.0.
- that you can use DAO 3.6 or ADO with the appropriate provider to write programs that work with Jet 4.0.
- about the architecture of the Jet database.
- how to connect to a Jet database.
- about the security capabilities for a Jet database.

✦ ✦ ✦