ð¤aí▒ß>■_ _½■___°¿«b_NÎ`Ë|_aËE©)£}└2 ñ
.
l■

Æ

u
t
Z____
____
____
____
_______________Ñ┴Y ┐¢Âbjbj«W«W
<æ=æ=¢▓_______]µµµµµµµ·····lf,·═22HHHHHHHÆöööööö$__«î©µHHHHH©HµµHHHHHHµHµHÆ·
·µµµµHÆH&HnèÇ_µµÆHÆäð▄_┴··Hp"What can a SQL statement accomplish?

Sort records
Choose fields
Choose records
Cross reference tables
Perform calculations
Provide data for database reports
Modify data

All SQL keywords in a SQL statement will be typed in upper case letters. Even though SQL is æcase-insensitive,Æ this is good programming practice and allows us (and others) to differentiate between keywords and other information in a SQL statement.

String information imbedded within a SQL statement can be enclosed in double-quotes (ô) or single-quotes (æ). With Visual Basic, you should only use single-quotes to enclose imbedded strings. The reason for this is that the SQL statement is itself a string - so, in Visual Basic code, SQL statements must be enclosed in double-quotes. We enclose imbedded strings with single-quotes to avoid confusion.

SQL supports the use of wildcards in forming data views. The wildcard character for the Jet engine is an asterisk (*). Use of wildcards will be illustrated in many examples. ANSI Standard SQL implementations use the percent sign (%) as a wildcard.

If a table or field name has an imbedded space, that name must be enclosed in brackets ([]). For example, if the table name is My Big Table, in a SQL statement you would use:

```
[My Big Table]
```

This notation is not allowed in some SQL implementations. But in implementations that donÆt recognize brackets, imbedded spaces in table and field names are not allowed, so it should never be a problem.

To refer to a particular field in a particular table in a SQL statement, use a dot notation:

```
TableName.FieldName
```

If either the table or field name has imbedded spaces, it must be enclosed in brackets.

Where Does SQL Fit In Visual Basic?

Visual Basic uses SQL queries to define a data source. SQL statements are processed by the Jet database engine (whether using DAO or ADO technology) to form a recordset object. This object contains the virtual database table formed as a result of the SQL statement. The resulting object can be used to display and, perhaps, update the database.

SQL with the DAO Data Control

When using the DAO (data access object) data control, the SQL statement takes the place of the RecordSource property of the control. In design mode, simply go to the Properties Window for the data control, scroll down to the RecordSource property and type in a valid SQL statement. Many times, this will be a very long property. Obviously, it is assumed that the DatabaseName property of the data control has been set to the desired database file.

In run mode, the SQL statement is also assigned to the RecordSource property of the data control (Refresh the data control after assigning the RecordSource). For example, if we have a SQL statement named MySQL (this will be a string type variable) we want to use with a data control named datDAOExample (again, it is assumed that the DatabaseName property has been appropriately set), the BASIC code syntax is:

```
datDAOExample.RecordSource = MySQL
datDAOExample.Refresh
```

We usually set the RecordSource property (and DatabaseName property, also) at run-time, rather than in design mode. Reasons for this are discussed in later chapters.

Whether in design or run mode, a valid SQL statement will return a Recordset object containing the selected database records. This object will have its own methods and properties for our use. In particular, to establish a valid RecordCount for the Recordset returned using a data control named datDAOExample, use these two lines of code:

```
datDAOExample.Recordset.MoveLast
datDAOExample.Recordset.MoveFirst
```

Following these lines, the RecordCount is examined using:

```
datDAOExample.Recordset.RecordCount
```

Quick Example 1 - SQL with the DAO Data Control

Start a new project. Add two label controls and a DAO data control. Set two data control properties to:

DatabaseName	BIBLIO.MDB (point to your working copy)
RecordSource	SELECT * FROM Titles

After setting the RecordSource property, the Properties Window should look like this:

EMBED PaintShopPro

Yes, this is your first SQL statement! You donÆt have to recognize this right now, but itÆs pretty easy to understand. The statement says SELECT all fields (the wildcard *) FROM the Titles table. This has the same result as choosing the Titles table as the RecordSource property.

Set the following two properties for the first label control (Label1):

DataSource	Data1
DataField	Title

Place this code in the Form_Activate procedure (this counts and displays the number of records):

```
Private Sub Form_Activate()
Data1.Recordset.MoveLast
Data1.Recordset.MoveFirst
Label2.Caption = Data1.Recordset.RecordCount
End Sub
```

Save and run the application. You should see this (the first label control showing a title and the second a number (the number of returned records):

EMBED PaintShopPro

Scroll through different titles using the data control arrows.

Now, add these two lines at the top of the Form_Activate procedure (these lines set the RecordSource at run-time):

```
Data1.RecordSource = "SELECT * FROM Titles ORDER BY Title"
Data1.Refresh
```

The SQL statement (enclosed in quotes since it is a BASIC string variable) is modified so the results are in alphabetical order.

Save and rerun the application. The æin codeÆ SQL statement should produce the same records, but in order:

EMBED PaintShopPro

SQL with the ADO Data Control

When using the ADO (ActiveX data object) data control, the SQL statement takes the place of the RecordSource property of the control. In design mode:

Establish the ConnectionString property.
Go to the Properties Window for the data control, scroll down to the RecordSource property and click on the ellipsis that appears. The RecordSource Property Page will appear.
Under Command Type, select 1 - adCmdText (this tells the control we will be using a SQL statement). Then, in the Command Text (SQL) window, type in a valid SQL statement. When done, click OK.

In run mode, the SQL statement is also assigned to the RecordSource property of the data control (Refresh the data control after assigning the RecordSource). For example, if we have a SQL statement named MySQL (this will be a string type variable) we want to use with a data control named datADOExample (again, it is assumed that the ConnectionString property has been appropriately set), the BASIC code syntax is:

```
datADOExample.RecordSource = MySQL
datADOExample.Refresh
```

We usually set the RecordSource property (and ConnectionString property, also) at run-time, rather than in design mode. Reasons for this are discussed in later chapters.

Whether in design or run mode, a valid SQL statement will return a

Recordset object containing the selected database records. This object will have its own methods and properties for our use. In particular, to establish a valid RecordCount for the Recordset returned using a data control named datADOExample, use these two lines of code:

```
datADOExample.Recordset.MoveLast
datADOExample.Recordset.MoveFirst
```

Following these lines, the RecordCount is examined using:

```
datADOExample.Recordset.RecordCount
```

Quick Example 2 - SQL with the ADO Data Control

Start a new project. Add two label controls and an ADO data control. Build the data control ConnectionString property to point to your working copy of BIBLIO.MDB.

Go to the Properties Window and click on the data controlÆs RecordSource property. Click the ellipsis. The RecordSource Property Page will appear. Under Command Type, select 1 - adCmdText. Then, in the Command Text (SQL) window, type:

```
SELECT * FROM Titles
```

You should see:

EMBED PaintShopPro

When done, click OK.

Set the following two properties for the first label control (Label1):

```
DataSource     Adodc1
DataField Title
```

Place this code in the Form_Activate procedure (this counts and displays the number of records):

```
Private Sub Form_Activate()
Adodc1.Recordset.MoveLast
Adodc1.Recordset.MoveFirst
Label2.Caption = Adodc1.Recordset.RecordCount
End Sub
```

Save and run the application. You should see something like this (the first label control showing a title and the second a number (the number of returned records):

EMBED PaintShopPro

Scroll through different titles using the data control arrows.

Add these two lines at the top of the Form_Activate procedure (these lines set the RecordSource at run-time) to modify the SQL statement:

```
Data1.RecordSource = "SELECT * FROM Titles ORDER BY Title"
Data1.Refresh
```

Save and rerun the application. The æin codeÆ SQL statement should produce the same records, but in order:

EMBED PaintShopPro

SQL with the ADO Data Environment

When using the ADO (ActiveX data object) data environment, the SQL statement forms a new Command object within an existing Connection object. In design mode:

Establish the Connection object (connect to a database).
Right-click on the Connection object in the Data Environment window and select Add Command. A new Command object will appear.
Right click the Command object and select Properties. The Properties window appears - make sure the General tab is selected. Under Source of Data, click SQL Statement. The SQL window will become enabled. Type a valid SQL statement, then click OK.

(In all these steps, it is assumed that proper conventions were followed in naming all objects.)

With the ADO data environment, we follow a different approach when using SQL statements in run mode. The recordset created based on design-time parameters is first closed (use the Close method). Then, we re-open the recordset using the Open method and the new SQL statement. For example, assume we have a data environment named denExample, a command object named comExample and a new SQL statement named MySQL (this will be a string type variable). Recall the recordset associated with comExample will be named rscomExample. The code to close the current recordset and re-open it with a new SQL statement is:

```
denExample.rscomExample.Close
denExample.rscomExample.Open MySQL
```

WeÆre not done, though. One more step is needed.

After creating the new recordset, all data bound controls are left bound to the original recordset. Without manually rebinding (in code) the controls to the new recordset, you wonÆt see the new results. Microsoft, in their Knowledge Base Articles, claims this is an intended behavior. We believe it is a bug that will hopefully be addressed as ADO technology matures. To rebind the data bound controls to the ADO data environment, you need to reset each controlÆs DataSource property. The code to rebind a control named ExampleControl to a data environment named DataEnvironmentName is:

```
Set ExampleControl.DataSource = DataEnvironmentName
```

Note use of the Set statement. Set must be used when initializing a programming object, as we are here. We will look at some automated techniques for rebinding in a later chapter. You can see that working with the data environment is a little trickier. But, after youÆve used it a few times, youÆll begin to appreciate its great advantages.

Whether in design or run mode, a valid SQL statement will return a recordset object containing the selected database records. Recall, in our example above, the returned recordset is named rscomExample. This object will have its own methods and properties for our use. In particular, to establish a valid RecordCount for the recordset returned by a data environment named denExample, use these two lines of code:

```
denExample.rscomExample.MoveLast
denExample.rscomExample.MoveFirst
```

Following these lines, the RecordCount is examined using:

```
denExample.rscomExample.RecordCount
```

Quick Example 3 - SQL with the ADO Data Environment

Start a new project. Add two label controls and two command buttons (needed to allow navigation among records). Add a Data Environment in the Project Explorer window. Right-click Connection1 and set Properties so it points to your working copy of BIBLIO.MDB.

Right-click on Connection1 and select Add Command. A new Command object will appear. Right click that object and select Properties. The Properties window appears - make sure the General tab is selected. Under Source of Data, click SQL Statement. The SQL window will become enabled. Type:

```
SELECT * FROM Titles
```

You should see:
EMBED PaintShopPro

When done, click OK.

Set the following properties for the first label control and the two command buttons:

Label1:

DataSource	DataEnvironment1
DataMember	Command1
DataField	Title

Command1:

Caption	&Previous

Command2:

Caption	&Next

Place this code in the Form_Activate procedure (this counts and displays the number of records):

```
Private Sub Form_Activate()
DataEnvironment1.rsCommand1.MoveLast
DataEnvironment1.rsCommand1.MoveFirst
Label2.Caption = DataEnvironment1.rsCommand1.RecordCount
End Sub
```

Add this code to the command button Click events to allow navigation:

```
Private Sub Command1_Click()
DataEnvironment1.rsCommand1.MovePrevious
If DataEnvironment1.rsCommand1.BOF Then
  DataEnvironment1.rsCommand1.MoveFirst
End If
End Sub

Private Sub Command2_Click()
DataEnvironment1.rsCommand1.MoveNext
If DataEnvironment1.rsCommand1.EOF Then
  DataEnvironment1.rsCommand1.MoveLast
End If
End Sub
```

Save and run the application. You should see something like this (the first label control showing a title and the second a number (the number of returned records û this may be a different value for you, depending on the current state of the BIBLIO.MDB database):

Navigate through the records, if you like.

Add these three lines at the top of the Form_Activate procedure (these lines set the RecordSource at run-time):

```
DataEnvironment1.rsCommand1.Close
DataEnvironment1.rsCommand1.Open "SELECT * FROM Titles ORDER BY Title"
Set Label1.DataSource = DataEnvironment1
```

These lines close the old recordset, re-open it with the new SQL statement, and then rebind the label control to the data environment.

Save and rerun the application. You obtain the same records, but ordered:

SQL Tester

Well, now we know some of the rules and syntax of SQL statements and how to use them with Visual Basic, but we still donÆt know what a SQL statement looks like (well, we saw one in the examples). We correct all that now and start learning more about SQL. To test SQL statements we form, we build this example which allows us to enter SQL statements and see the results of the formed database queries. In this example, we use the DAO data control so both Visual Basic 5 and Visual Basic 6 users can build the same example. You can choose to use the ADO control (or data environment) if you choose.

Start a new project. Add a DAO data control, a text box control, two label controls, a command button, and a DBGrid control to the form. Wait, you say, what is a DBGrid Control and why isnÆt it in the toolbox? It is a DAO data bound control we havenÆt looked at yet, but it is very powerful. The DBGrid control allows us to view and edit an entire database table by setting just one property (DataSource). It is a custom control that must be added to the toolbox. To do this, select Components under the Project menu item. In the window that appears, check the box next to Microsoft Data Bound Grid Control, then click OK. It is then available for selection from the toolbox. We will look further at this control in Chapter 6.

Resize and position the controls so your form looks something like this:

EMBED PaintShopPro

A Brief (Hopefully) Interlude for Visual Basic 6 Users:

When you selected the Components tab, the choice for the Microsoft Data Bound Grid Control may not have been there. You will see a choice for Microsoft DataGrid Control. This is not the same control û this is the version of the control that works with ADO technology. So what can you do? There are two solutions: a quick one and a ænot-so-quickÆ one. We recommend the latter.

Solution 1 û The Quick Solution:

Use the ADO DataGrid control (make sure it has been added to the toolbox) in place of the DAO data bound grid control. You will also have to replace the DAO data control with the ADO data control. Use the same properties for the grid control and the data control with one exception. Recall the ADO data control does not have a DatabaseName property. If using the ADO control, set the ConnectionString property such that it points to your working copy of the BIBLIO.MDB database. No code changes are necessary û the code that works for the DAO data control will work for the ADO data control.

When you attempt setting the DataSource property for the grid control, you will get this error message:

This is acceptable since we will be setting the data controlÆs RecordSource at run-time. You may also get this error when running the application. If so, just click OK. For your reference, we have built an ADO version of the SQL Tester program and included it with the example files (look for the project file with the AD suffix).

Solution 2 û The æNot-So-QuickÆ Solution:

Here, we will install the desired DAO data grid control (and other DAO-based controls, if desired) onto your computer. The steps are many, but the effort is worth it, especially if you ever plan to use or build applications that employ DAO database technology. The information provided here was taken from MicrosoftÆs website. You will need your installation CD for Visual Basic 6. You will also have to be familiar with issuing DOS command line statements. Ask for help from someone if this is unfamiliar.

Look in the \COMMON\TOOLS\VB\CONTROLS directory on the VB6 CD. This directory contains controls that shipped with Visual Basic 4/5 Professional and Enterprise Editions, which are no longer shipping with Visual Basic 6:

AniBtn32.ocx, Gauge32.ocx, Grid32.ocx (the file we are interested in here), KeySta32.ocx, MSOutl32.ocx, Spin32.ocx, Threed32.ocx, MSChart.ocx

To install these files on your computer, follow these steps:

Copy all of the files in this directory to your \WINDOWS\SYSTEM directory.

Register the controls by either Browsing to them in Visual Basic itself (select the Browse option when selecting Components), or manually register them using RegSvr32.Exe. RegSvr32.EXE can be found in the \COMMON\TOOLS\VB\REGUTILS directory. The DOS command line is:

```
regsvr32.exe grid32.ocx
```

Register the design time licenses for the controls. To do this, merge the vbctrls.reg file found in the \COMMON\TOOLS\VB\CONTROLS directory into your registry. You can merge this file into your registry using RegEdit.Exe (Win95, Win98, WinMe, Win2000 or WinNT4) or RegEd32.Exe (WinNT3.51):

```
regedit vbctrls.reg (or other reg files associated with the controls)
```

The DAO files (including the DAO data grid control) should now appear in the Components listing when choosing controls to add to your toolbox. Now back to our example.

Set properties for the form and controls:

Form1:
Name frmSQLTester
BorderStyle   1-Fixed Single
Caption   SQL Tester

Data1:
Name datSQLTester
Caption   SQL Tester
DatabaseName   BIBLIO.MDB (point to your working copy)

Label1:
Caption   Records Returned

Label2:
Name lblRecords
Alignment 2-Center
BackColor White

BorderStyle	1-Fixed Single
Caption	0
FontSize	12

Command1:

Name	cmdTest
Caption	Test SQL Statement
TabStop	False

DBGrid1:

Name	grdSQLTester
DataSource	datSQLTester
TabStop	False

Text1:

Name	txtSQLTester
MultiLine	True
ScrollBars	2-Vertical

When done, the form should look like this:

EMBED PaintShopPro

With this example, we will type SQL statements in the text box area, then click the Test SQL Statement button. The data grid will display the returned records, while the label control will display the number of records returned. We need some code to do all of this.

All the code goes in the cmdTest_Click event:

```
Private Sub cmdTest_Click()
'Enable error handling
On Error GoTo SQLError
'Read SQL statement and establish Recordsource
datSQLTester.RecordSource = txtSQLTester.Text
datSQLTester.Refresh
If datSQLTester.Recordset.RecordCount <> 0 Then
  datSQLTester.Recordset.MoveLast
  datSQLTester.Recordset.MoveFirst
  lblRecords.Caption = datSQLTester.Recordset.RecordCount
Else
  lblRecords.Caption = "0"
End If
txtSQLTester.SetFocus
Exit Sub
'If error occurs, report it in message box
SQLError:
MsgBox Error(Err.Number), vbExclamation + vbOKOnly, "SQL Error"
Exit Sub
End Sub
```

LetÆs spend some time seeing whatÆs going on in this code. The first thing we do is turn on error trapping. Without it, if we make a small error in a SQL statement, the program will stop. With it, we get a message indicating our mistake and are allowed to continue. Following error control, the SQL statement (from txtSQLTester) is processed and the Recordset established. The records are then counted and displayed.

Be careful in typing SQL statements. Although we have error trapping in SQL Tester, if you make a mistake, the returned error messages are (many times) not of much help. If you get an error, the best thing to do is retype the SQL command, paying attention to spacing, spelling, and proper punctuation.

Save the application and run it. Type the only SQL statement you know at this time in the text box (SELECT * FROM Titles). Click Test SQL Statement and you should see:

 EMBED PaintShopPro

Note the DB grid control display the entire table. You can scroll through the table or edit any values you choose. Any changes are automatically reflected in the underlying database. Column widths can be changed at run-time. Multiple row and column selections are possible. As we said, itÆs a very powerful tool. Please note Records Returned values for your results may be different, depending on the current data in the database.

Change the word SELECT to SLECT to make sure the error trapping works. Now, letÆs use this SQL Tester to examine many kinds of SQL statements. When typing the statements, use upper case letters for the SQL keywords. Statements do not necessarily have be on a single line - multiple line SQL statements are fine and usually make them easier to read and understand.

SELECT/FROM SQL Statement

The most commonly used SQL statement is the one weÆve been using as an example: the SELECT/FROM statement. This statement allows you to pick fields from one or more tables.

The syntax for a SELECT/FROM SQL statement is:

```
SELECT [Fields] FROM [Tables]
```

where [Fields] is a list of the fields desired and [Tables] is a list of the tables where the fields are to be found. The wildcard character (*) can be used for the fields list to select all fields from the listed table(s). For example, the statement we have been using:

```
SELECT * FROM Titles
```

selects and returns all fields from the BIBLIO.MDB database Titles table. Look at all fields in the other tables (Authors, Publishers, Title Author) using similar statements. When looking at the Title Author table, you need to write:

```
SELECT * FROM [Title Author]
```

Recall field and table names with imbedded spaces must be enclosed in brackets. Looking at each table will reacquaint you with the structure of the BIBLIO.MDB database tables and fields. We will use a lot in the rest of this chapter.

If we only want selected fields from a table, we use a field list, which is a comma-delimited list of the fields desired, or:

```
SELECT Field1, Field2, Field3 FROM Table
```

will return three named fields from Table. Make sure you do not put a comma after the last field name. To obtain just the Title and Year Published (name must be enclosed in brackets because of imbedded space) fields from the books database Titles table, use:

```
SELECT Title,[Year Published] FROM Titles
```

Note the field names are not written using the prescribed dot notation of Table.Field. The table name omission is acceptable here because there is no confusion as to where the fields are coming from. When using multiple tables, we must use the dot notation.

Try this with the SQL tester and you will see just two fields are returned.

EMBED PaintShopPro

The DISTINCT keyword can be used with SELECT to restrict the returned records to one per unique entry for the field. That is, there are no duplicate entries. As an example, first try this with the SQL tester:

```
SELECT PubID FROM Titles
```

EMBED PaintShopPro

Now, try:

```
SELECT DISTINCT PubID FROM Titles
```

EMBED PaintShopPro

You should see far fewer records are returned - only distinct publishers are returned.

ORDER BY Clause

When you use a SELECT/FROM statement, the records are returned in the order they are found in the selected table(s). To sort the returned records in some other order, you use the ORDER BY clause. The syntax is:

```
SELECT [Fields] FROM [Tables] ORDER BY FieldSort
```

This statement selects the listed fields from the listed tables and sorts them by the field named FieldSort. By default, the ordering is in ascending order. If you want the sort to be in descending order, the FieldSort name is followed by the keyword DESC.

Try this statement with the SQL Tester:

```
SELECT * FROM Titles ORDER BY PubID
```

All records in the Titles table will be returned in order of Publisher ID.

EMBED PaintShopPro

Try this and the order should be reversed:

```
SELECT * FROM Titles ORDER BY PubID DESC
```

You can use more than one field in the ORDER BY clause. SQL will create a recordset based on all requested orderings. Try this with SQL tester:

```
SELECT * FROM Titles ORDER BY PubID,Title
```

The returned records will be in order of the publishers, with each publisherÆs titles in alphabetic order.

EMBED PaintShopPro

If you want to restrict the number of records returned by a SQL statement that orders the returned records, you can use the TOP keyword with SELECT. TOP n returns the first n records. TOP n PERCENT returns the first n percent of the returned records. If two or more records have the same order value, they are all returned. Use the SQL Tester and try:

```
SELECT TOP 20 * FROM Titles ORDER BY PubID,Title
```

Twenty books should be returned. Now, try:

```
SELECT TOP 20 PERCENT * FROM Titles ORDER BY PubID,Title
```

Far more books will be returned.

WHERE Clause

One of the most useful aspects of the SELECT/FROM SQL statement is its ability to limit the returned recordset via the WHERE clause. This clause specifies some criteria that must be met in forming the recordset. The syntax is:

```
SELECT [Fields] FROM [Tables] WHERE Criteria
```

The WHERE clause limits the number of returned records by allowing you to do logical checks on the value of any field(s). Operators used to perfom these checks include:

<	Less than	<=	Less than or equal to
>	Greater than	>=	Greater than or equal to
=	Equal	<>	Not equal

Other operators are:

Between	Within a specified range
In	Specify a list of values
Like	Wild card matching

The WHERE clause can limit information displayed from one table or combine information from one or more tables. First, letÆs do some several single table examples using SQL Tester.

Single Table WHERE Clause

Say we want to see all fields in the BIBLIO.MDB Titles table for books published after 1995. And, we want the returned records ordered by Title. The SQL statement to do this is (weÆll type each clause on a separate line to clearly indicate what is going on - multiple line SQL statements are acceptable and, many times, desirable):

```
SELECT *
FROM Titles
WHERE [Year Published] > 1995
ORDER BY Title
```

This is where the real power of SQL comes in. With this simple statement, the Jet database engine quickly finds the desired records and sorts them - all without any coding on our part!

EMBED PaintShopPro

What if we want to know information about all the book publishers in the state of Washington. Try this SQL statement with the BIBLIO.MDB Publishers table:

```
SELECT * FROM Publishers WHERE State = æWAÆ
```

Note we enclosed the state name abbreviation (a string) in single quotes, as discussed earlier in this chapter. Try this SQL statement with the SQL tester and you should find one lonely publisher (BetaV) in the state of Washington! Wonder where Microsoft is?

The BETWEEN keyword allows us to search for a range of values. Want all books published between 1995 and 1998? Use this SQL statement:

```
SELECT * FROM Titles WHERE [Year Published]
BETWEEN 1995 AND 1998
```

EMBED PaintShopPro

The IN keyword lets us specify a comma-delimited list of desired values in the returned recordset. Say, we want to know the publishers in New York, Massachusetts, and California. This SQL statement will do the trick:

```
SELECT * FROM Publishers WHERE State IN (æNYÆ, æMAÆ, æCAÆ)
```

EMBED PaintShopPro

The LIKE keyword allows us to use wildcards in the WHERE clause. This lets us find similar fields. Recall, the Jet engine wildcard character is the asterisk (*). To find all authors with a ægÆ anywhere in the their name, try:

```
SELECT * FROM Authors WHERE Author LIKE æ*g*Æ
```

EMBED PaintShopPro

Multiple criteria are possible by using the logical operators AND and OR. For example, to find all books in the Titles table published after 1993 with a title that starts with the letters Data, we would use the SQL statement:

```
SELECT * FROM Titles
WHERE [Year Published]  > 1993 AND Title LIKE 'Data*'
```

EMBED PaintShopPro

Multiple Table WHERE Clause

So far, almost everything weÆve done in this course has involved looking at a single native (built-in) table in a database. This has been valuable experience in helping us understand database design, learning how to use the Visual Basic database tools, and learning some simple SQL statements. Now, we begin looking at one of the biggest uses of database management systems - combining information from multiple tables within a database. SQL makes such combinations a simple task.

We still use the same SELECT/FROM syntax, along with the WHERE and ORDER BY clauses to form our new virtual tables:

```
SELECT [Fields]
FROM [Tables]
WHERE Criteria
ORDER BY [Fields]
```

The only difference here is thereÆs more information in each SQL statement, resulting is some very long statements. The [Fields] list will have many fields, the [Tables] list will have multiple tables, and the Criteria will have several parts. The basic idea is to have the SQL statement specify what fields you want displayed (SELECT), what tables those fields are found in (FROM), how you want the tables to be combined (WHERE), and how you want them sorted (ORDER BY). LetÆs try an example.

Notice the Titles table does not list a bookÆs publisher, but just publisher identification (PubID). What if we want to display a bookÆs title (Title field in Titles table) and publisher (Company Name in Publishers table) in the same recordset? LetÆs build the SQL statement. First, the SELECT clause specifies the fields we want in our ævirtualÆ table:

```
SELECT Titles.Title,Publishers.[Company Name]
```

Note the use of dot notation to specify the desired fields. With multiple tables, this avoids any problems with naming ambiguities.

The FROM clause names the tables holding these fields:

```
FROM Titles,Publishers
```

The WHERE clause declares what criteria must be met in combining the two tables. The usual selection is to match a primary key in one table with the corresponding foreign key in another table. Here, we want the publisher identification numbers from each table to match:

```
WHERE Titles.PubID = Publishers.PubID
```

Any records from the tables that do not match the WHERE criteria are not included in the returned recordset.

Lastly, we declare how we want the resulting recordset to be sorted:

```
ORDER BY Titles.Title
```

The complete SQL statement is thus:

```
SELECT Titles.Title,Publishers.[Company Name]
FROM Titles,Publishers
WHERE Titles.PubID = Publishers.PubID
ORDER BY Titles.Title
```

Try this with the SQL tester.

EMBED PaintShopPro

Are you amazed? You have just seen one of the real powers of using SQL with the Jet database engine (or any database system, for that matter). We simply told the engine what we wanted (via the SQL statement) and it did all of the work for us - no coding needed! LetÆs do some more examples.

In the previous example, say you just want books published by Que Corporation. Modify the SQL statement to read (we added an AND clause):

```
SELECT Titles.Title,Publishers.[Company Name]
FROM Titles,Publishers
WHERE Titles.PubID = Publishers.PubID
AND Publishers.[Company Name] = æQUE CORPÆ
ORDER BY Titles.Title
```

EMBED PaintShopPro

What if we want to list a bookÆs title, publisher, and author, ordered by the author names? This requires using all four tables in the BIBLIO.MDB database. LetÆs build the SQL statement. We want three fields:

```
SELECT Authors.Author,Titles.Title,Publishers.[Company Name]
```

As mentioned, to retrieve this information requires all four tables:

```
FROM Authors,Titles,Publishers,[Title Author]
```

We still need the publisher identification numbers to match, but now also need to make sure book titles (via the ISBN field) and author identification numbers match. The corresponding WHERE clause is:

```
WHERE Titles.ISBN = [Title Author].ISBN
AND Authors.Au_ID = [Title Author].Au_ID
AND Titles.PubID = Publishers.PubID
```

Finally, the results are sorted:

```
ORDER BY Authors.Author
```

Putting all this in the SQL tester gives us over 16,000 listings (one entry for every author and every book he or she wrote or co-wrote):

EMBED PaintShopPro

Such power! Can you imagine trying to write BASIC code to perform this record retrieval task?

If the displayed field name does not clearly describe the displayed information, you can alias the name, or change it to something more meaningful using the AS clause. As a simple example, try this:

```
SELECT Au_ID AS [This Author] FROM Authors
```

Notice the displayed column is now This Author.

EMBED PaintShopPro

The field name is unaffected by aliasing - only the displayed name changes.

Important - Database tables combined (forming a virtual data view) using DAO technology and the SQL WHERE clause cannot be updated. The data can only be viewed. Go ahead - combine tables using a SQL statement with a WHERE clause and try to change a value in the resulting grid. You canÆt do it! The ability to update a DAO recordset is established by the read-only Updatable property. Is this a problem? Not if you are just displaying information for a user. But, if you need editing capabilities with DAO, do not use the WHERE clause to join tables.

Any recordset established using ADO technology (even with a combining WHERE clause) can be updated, depending on locks placed on the recordset. The use of such locks is discussed in a later chapter.

To provide editing in DAO recordset, you need to use the SQL JOIN clauses. Using JOIN clauses will also work with ADO recordsets. LetÆs take a look at such a clause.

INNER JOIN Clause

When combining tables, the SQL INNER JOIN clause does the same work as the WHERE clause and it returns a recordset that can be updated (for both DAO and ADO technologies). The syntax for an INNER JOIN is a little different than that of the WHERE clause.

```
SELECT [Fields]
FROM Table1 INNER JOIN Table2 ON Linking Criteria
WHERE Criteria
ORDER BY [Fields]
```

This rather long statement begins by specifying the fields to SELECT. The FROM clause specifies the fields will come from the first table (Table1) being INNER JOINed with a second table (Table2). The ON clause states the linking criteria (usually a matching of key values) to be used in the join. At this point, the tables are combined. You can still use a WHERE clause to extract specific information from this table (you just canÆt use it to combine tables) and an ORDER BY clause, if desired. LetÆs repeat the examples just done with the WHERE clause.

To display a book title and publisher name, the SELECT clause is:

```
SELECT Titles.Title, Publishers.[Company Name]
```

We want to æjoinÆ the Titles table with the Publishers table, making sure the PubID fields match. The corresponding INNER JOIN statement is:

```
FROM Titles INNER JOIN Publishers
ON Titles.PubID = Publishers.PubID
```

Lastly, we order by the Title:

```
ORDER BY Titles.Title
```

Try this SQL statement in the SQL tester and you should obtain the same results seen earlier with the WHERE clause:

EMBED PaintShopPro

Try to change a value in the data grid for this example. You should see that, as expected, use of the INNER JOIN provides an updatable recordset. If you leave your change as is, it will be written as a permanent modification to the database! So, we suggest æundoingÆ your change. You have just learned one of your first skills in building a complete database management system - how to edit an existing database. It was easy, wasnÆt it? This ease comes from the power of the Jet database engine. There are times we wonÆt want editing the database to be so easy. Limiting these capabilities are discussed in the next chapter on Visual Basic interfaces.

To illustrate use of the WHERE clause (to limit displayed records) in conjunction with the JOIN clause, try this modified SQL statement with SQL Tester:

```
SELECT Titles.Title, Publishers.[Company Name]
FROM Titles INNER JOIN Publishers
ON Titles.PubID = Publishers.PubID
WHERE Publishers.[Company Name] = æQUE CORPÆ
ORDER BY Titles.Title
```

Only QUE CORP publishers will be listed. And, the recordset can still be edited (WHERE only affects æupdatabilityÆ of DAO recordsets when used to combine information on tables).

Use of the INNER JOIN clause to combine information from more than two tables is a little more complicated. The tables need be joined in stages, nesting the INNER JOIN clauses using parentheses for grouping. Assume we have three tables (Table1, Table2, Table3) we want to combine. Table1 and Table3 have a common key field for linking (Key13), as do Table2 and Table3 (Key23). LetÆs combine these three tables using INNER JOIN. In the first stage, we form a temporary table that is a result of joining Table2 and Table3 using Key23 for linking:

```
Table2 INNER JOIN Table3 ON Table2.Key23 = Table3.Key23
```

In the next stage, we join Table1 with this temporary table (enclose it in parentheses) using Key13 for linking:

```
Table1 INNER JOIN
(Table2 INNER JOIN Table3 ON Table2.Key23 = Table3.Key23)
ON Table1.Key13 = Table3.Key13
```

This nested statement is used in the SQL statement to specify the tables for field selection. Notice weÆve spread this over a few lines to make it clearer - any SQL processor can handle multiple line statements. The multiple table INNER JOIN can be generalized to mso mso mso|nbeing INNER JO2, T2, c

ToA▄dc╕║rs tPn now This Author.

 EMBED PaintShopPro

EE, wE, wEietÆing ╬/ing p:Fe t▄ feJOIN Tablles.TitbookÆs titne tab tab tab tab tab tab tab tab tab ement to specify the tabl
We wa;+jin│╬d aftevlose it in pas usinà·lesefû<t with SQL Tester:

SELECT Ti(sec Bey1ievinking:

Tabl(rat aly deÌE÷A▀he linking critcritcritcritcrcs us over 1ble
(you2,Cparenbe editedat lin(e3.Key13

This nestis ncodU same results seen earlier with A▄dc╕║rs tPn   now This AuthorE÷AQE÷AQE÷AQE÷AQE÷AQE÷AQw÷AQx╠s, the [Tabultiple r w=oN can beÈGbeÈPro Thegatabasabasabasabasa   is:1, Table2²is etÆs bÇ

ÖÖÖÖÖÖÖÖÖÖÖÖÖHERE3Àation ffrary table (e_ELE©frarx = TablÖÖÖß╬Iiinking:

Tabl(rat aly dehertab tab tabRthorE÷Aow btain the same rqe ro Tabs' EMày tày tày tày tàyAow btaw btaw b
Compaary table (e_ELE©frarx = Tablit in pas .   4Öær._ëdbaEt

LookLooxually a matkLoox cont mu_the same rq/ wee_EL+jin│╬d aftevlose it in
pô,Titlto ch│╬d aft_to ¥bpô,T mso mso|nbeing INNER JO2, T2, c

ToA▄dc╢rs tPn now This Author.

EMBED PaintShopPro wee_EL+_EL+_EL+_EL+_EL+_EL+_EL+_ELä_ELäkLooxO
recordßfrom Tableableablefror._ëdbaEt LookhopPropPodifiPyROM   a matkL
[Tit[Titãth egataspeciaRpas le1_so|nbeing INNER JO2, T2, c

ToA▄dc╢rs tPn now Thi╟
heda1.....................................lause toe toe toe toe
toe toe toséæ\ the INNER ...l....... the INe1_so|
nb+_EL+_EL+_EL+_ELä_ELäkLoo fê_±d seto mwee_Eee_mSQLTester
Bon this chapterpterp have just1_so1_so▄° rqe ro Tabs' EMày tày tày tsP
ÀPe that is a rG┴!rG┴!rG┴!r EMày tà,Titlto ch│╬d afE÷A, nes)aues is a
li EMàes with DAO, Î_ DAO, Î_ D...& to combine i)d

Label2r - any bli}i}i}i}i}i}l(rat aly deä
_)ume we have three PS....xE_the) statement is useIGNnTòce ittlaues
oÖÖÖÖÖÖÖÖÖÖÖÖÖÖÖÖ_

Tabò,s eP ÀeÉ

Tabò,s o ch│╬d aft_Éft_Éft_sLä_ ittlft_Éft_sEee_mSQLTes59 nc«|Gc«|tlaues in
Tab═t. Th INNER ...l....... the INe1_so|nb+_+_+_+_+_rts.ƒtpabel2r ┼¢used
to combine xne xne xne xn▄95 AND 1998
w coo mso mso xne xn▄95[y tsP ÀPe

ss_u n dba æupdatabilityÆ of DAO recordsets when used to combine
information on tables).

Use of the INNER JOIN clause use uss_u n dbthin aINNEO to retlatabase to
be so easy.of Xido ¥0ãeda1eda1eda1eda1eda1ed
This nis nisTMisTecord of the G┴!_G┴!_G┴!_G_ª_G_ª_G_ª_G_ªr blizeing ing
T)tÊis nested statement is used in the SQL statement to specify the tables
for field selection. Notice _%or fRE3T2, c

ToA▄d_b┐he bigGx6OIN│loR SQLokLoox,Ns╝e it%)Uu┼¢usQL stR
SQLokLoox,NsinforbigGx6OI
n_5T╗Đe2 and T║s useIGNnTòThcifies the e) statement is dbthin aINNEO to
retlatabase t expe...speh theg█with the letfields lùsrid f EMBED
TablÖÖÖß╬Iiinkitle letfields1.Key stage, weielúi<r╣rs z letfied f EMnl...Ä1
stage,A_sO stîbÎ+_««««««««««««««««««««usem╝al BÎ+_Î+_Î+«««««usemgJOINJe of
thfE÷A, nes)emgJOINJe aPied f EMnl.=ÆóhYgs simpl..l....... the INe1_so|
nb+_+_+_+_+ith««««««usem╝al BÎ+_Î+_Î+«««««usemgJOINJe of thfE÷A,
nes)emgJOINJe aPied f EMnl.=ÆóhYgs simplimplim/Notice _ce)
e)C$ébles).
╔W¥)C$ébles).
rentheses) usinyr nyr nyr©.
reo1ed
This nis nisTMisTeco.ets whenmrary tay tay tay tay tay tay tay tay tay tay
tay tay tay taym«tay tay taym«tay tay tayÆóhYgs simplimplim/Notice _ce)
e)C$éb)aPied f EMnl.=ÆóhYgs simpl..l....... the INe1_so|nb+_+_INNER
PublishcordCounTes59 nc«|Gcle lev tay «s Autht is un-tir groupi le╔! o+ER
JOn JOn└ s......TòThcifies the e) statement i\nt i\nt i\nâ
taym«tstatstatsta¡se it in pô,Titlto ch│╬d aft_to ¥bpô,T ut│╬d aft_to_c╣rs
lause use uss_u n dbthin aINNEO to retlatabase to be so easy.of et in
forming the recordseg tËy tay o ea named rscoletfied f EMnl...Ä1 stagmcoo
mso msA dbgÖ_oletf...┤loTQ0
w coo mso mso xne xn▄95[)C$ébles[ßch│╬d aft_ÖÖÖÖÖÖÖ_

S«««usem╝a+jin│╬d aftevlose it in pô,Titlto ch│╬d afd afd afd
afdselNetfieldse?his nrarA
dbgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bgÖ_bg
Ö_bgÖ_bgÖ_bgÖ_bgÖ(┤l©peh m)m)øe the _alues in the ╔Ö_bgÖs_bgSort name is
in Ðhis requires ª_G_ªLr╩dtay tay tsage inn aINNEO requioöequirultiß.de
result).
╔W¥)C$ébthis SQy tay ïtion f¬tiß.de resulesulesunÄ
Je of thfE÷A, nes)emgJOIUJOIw ticifiese of thfE÷A, nPjWG2, c

ToA▄d_b_Éft_sLä_ ittlft_Éftq, a c

ToAÊXoo e sy_sy_sy_sy_sy_sy_s e sy_sy_ about all the book publishers in the
state of Washin|Gc«|┼ c

Tx,Ns╝eNs╝eNs╝eNs╝eNs╝eNs╝eNs╝eNsNs╝ f El powe a ■NNER JOIN
(T
(T┬òqë|Ö_bgÖ_bgfE÷utht nrarA ░iel3fields lùsrid f EMBED TablÖÖÖß╬Iii-
▓(fuers in tg«ers) itÂsted statement is used in the SQL stturne a s in
wee&ableableableaß╬Iii-▓(fu.)C_ol
aß╬µmpl..l......tagmcoo mso msA dbgÖ_oletdbg6;)td+it░KE keywofieldse_o
traptraptr nPjWGAQx╠s¢ tsóhYAO, Î_ D...& to com©el3etÆ[y┐
dbgÖ_oletf...┤loTQ0
w coo mso mso xne xn▄95[)fd f_or

Putting ang SQÖær._ëdbaEtAm┴£Tabl(abl(abl(abl(abl(ablo mso mso xno xnoxnoc
SQL makes such combinaS SQL mak...........................urne a
ÄS┬tsóhYAO,rne a........

▓rma

▓rma

▓rma

v├onTable3 have a common key field for lie a........

▓rma

[redacted]rma

▓rma

v├onTable3 have a common key field for lie a.....ble3 hav makfiese of
thfE÷A, nPjWG2, c

ToA▄d_b_Éft_sLä_ ittlft_Éftq, a c

ToAÊXoo e sy_sy_sy_sy_sy_sy_s e sy_sy_ about all the book publishers in the
state of Washin|Gc«|┼ c

Tx,Ns╝eNs╝eNs╝eNs╝eNs╝eNs╝eNs╝eNsNs╝ f ElJ┤l©pentheses_itÂsla(T
(T┬òqë|Ö_bgÖ_bgfE÷utht nrarA ░iel3fie for liet(»ed:©╗

S«««usem╝a+jin|╬d afte5
©╗

2tme muthe state oftS«««usem╝a+jiof Washin|Gc«|┼ c

Tx,Ns╝eNs╝ssulesulesA oftS«««usem╝a+jiof Warid f Ee SQLí╬d MtR o╬+jiof Washin|Gc«|┼ c

ToTQ0
Oa(T
(Tærisher ID.

EMBED Pain┼ c

Tou n dba æupdatabilitysITPubIDc

»ÿc

»ÿplds 2tm in tnubIDI_s e gÖ_bg
This nis nin|Gc«n aINNEO to retlatabase toà·lIl_flit£■æate
oY/usem╝a£tnubIDI_s _%or fRE3T2, ôa common NEOa
reo1ednyr©R ...l...l...l...l...l_ísM SQ.l.¥atabilitysI-2┼ cules)TPubKxî,a
reo1ednlie a........

æupdatabilitys╗tys╗tys╗t_WÀ kel a.sI-l_fliO)emgJOIUJOIw ticifiese of
thfE÷A, nPjWG2, c

ToA▄d_b_Éft_sLä_ iThis nis nin|Gc«n aINNEO tould ng DOS command line
statc«n aINNEO and╦uthor,Titles.Title,Publi = gh w

o╬w,C$ébles[ßchhe Jet database engine. There are t e fo, cc«|tne roe
r8ÂemgJOIUJOIw ticifiese oÌce se oÌce se oÌctc«n aINo╬+ tould ng
s╝eNs╝ssulrlng sng table INNER JOIN catate of Washin|Gc«|┼ c

Tx,Ns╝eNs╝eNs╝,C$éÉÃ ÉÃ ÉAuh th¤Ns╝:f«┐_INNEO NsinfO °O dÀ▓rmo╬+jioÌctcin|
G afte5
©e5
©e5
©e5
NNEO to rc«|c«o1ednlie a........

æupdatabilitys╗tys╗tys╗t_WÀ kel a.sI-l_fliO)emgJOIUJOIw tiith F┼ c

TouIM th¤Ns╝:f«┐ the:f«o╬a
ry_sy_sy_s v tould ng aftevltev2 The Then IN_another table. Here, we
want the publisher ideu╚' gh gh gh INNEaiIt all the bséæ\ th«IlïcNNEaiIt
all as discussed dFséæ\éæ\éæ\htries. Oséæ\ horesiresiresir O
RoednliI-2x,NsiO, Î_ D...& to& to& to& to& to& to& to& x4e

to cusinà·lese}
toMBEnà·lese} allr tabi©qb÷OIUJ tày tày tsP ÀP   ENEaiIt oY/usem╝a...l_ísM
s lùs┌ maîSho,a
reo1ednlie a........Ent allows yoower ,╬etÆws ÆwsmgJnis nTærisher
ID.½ris_=Cdîdur ævirtualÆ taÆ taÆ taÆ taÆ taÆaÆ taÆ x Washin|Gc«P
ENEaiIt oY/usem╝a...l_ísM s lùs┌ maîSho,a
reo1ednlie a........Entto Sø@HÅ╝,C$ébIDI_s ento_=_sgÖ_s lùs lwPo s lùs┌
maîSho,a
dis tùs┌ m

BÅ▀e sy_sy_ abo EMnl...Ä1 stage,A_sO
stîbÎ+_««««««««««««eNs╝ssulesu_ðwdassachu■sée..l_íBBBBt is used in the SQL
statemet oY/usem╝¬t$dDeme»koaîSho,a
reo1ednlie a..╝5d░dn gh thor,Titles.Ti┌.Tiræs lùs lw░;tuf ticifiese of
_the THere, wùs lw░jin│╬d afte5
©m╝¬t$dDeme»;½gÖ_Åset■ùs lwv+_+:È□7h 3 ususu¬t$Ë¬t$Ë¬t$de a
ÄS┬tsóh╦recors lwPo s lùs

□ùs

□ùs

g
This nis dùs

VÅABED PaINJÌONJÌe5
©M sÿRbplimnge
In   Spec_se

g

This n The Then IN_aen c«|..l_ísMrish Thä Thä Thä ©╗

2tme muthe st st st st »t »t╩ kel a.sOa.sOa.sO st st stle3╩ s

VÅABED PaINJÌONJÌe5
©M sÿRbplimnge
In   Spec_se >nâ
eîâ
eîâ?, a c

ToAÊXoo e o e o d »tÎ folloo e o e o d »tÎ folloo ╩ s

```
VÅABEDJÌe5





to se >nâ
thor

Puttinmakyou n^Ç█a æup_tilt_leo1ednlie a..╝äO  ðeNs╝ss»|s nis nìhatng
ereeNs nìhatng ong ong omîâ?±Entto Sø@HÅ╝@HÅ╝esu_ðwdassach?, hfu Sø@HS┬tsóh
R@HÅ ¢# e o e o  demgJOIUJOIuse.use.N c.N c.o e R@HÅ ¢# │oY/usem╝onge
Io¿ tng onsing Ke locks is disc□Ç;(utht is  un-tir╣osy_ about all tll t.
ToAÊXoo e o e o»;½gÖ5A▄d_b_Éft_sLä_ ittlayË e >databases lùs┌
maîSkinTINJÌONJÌONJÌeT......s)Ts  in ÐbininrJÌO c.o
e.oe.oe.oe.oe.databilitys╗tbi order by the Titleu¬t is used in Rg kel ul ul
ul.oe.oe.ey ey ey ey e>ie ulice chu■sée.RuÖNouÖNHÅ╝@HÅ╝s╝sùsls usenlie
a_b_.Ru>╔Ru>╔Ru>╔TJ tày tonge
Î f..s)Ts  in ÐbininrJÌO c.o e.oe.oG F┼ c

 c

y5e
In   Spec_se
```

g
Thi c.ée.R1ednlie a..╝äO ðeNs╝ss»|s nis nìhatngSø@4u Sø@HSg kel ul ul
ul.oe.oe.eosude RonsàSø@onsàe iel ul ul ullw░llw░llwh Rsóhàsem ulngeu thAo
d.oGto cusicus═g ong omsirs wsirs wsirwsirwD Pain┼ c

Tou n lie a_b_.Ru>╔Ru>╔Ru>╔TJ i cabases lùs┌lb_.Ru>╔Ru>╔B¢2, c
░ll+, t╡║uSQL s╡║uSQL s_sse}
toMBEto cus+in lngPaINJÌONJÌe5
nstage,A_so Sø@HÅ╝@HÅ╝esu_ðwdassach?, hfu Sø@Hy ey Ìhicifiese of _the e,A_so
hfu SRkelVkelVkelVk
Lastly, we declare how we want the àe iel RelVRelVRe8VRe8Vare
howRkO,rnd,rnd,rne er■ er■ _so Sø@HÅ╝@HÅ╝eu°ve th DA¼dàol ul u_hfu ²Åu
²Åu_l uGc«n aINNEO tould nyou i i i it_És©e5
©e5
©5
©±b¬òlw░llw=B=B=B=B=r=B=r=B=r=B=r=Be5
we5
we5 tould nyou5[)fdf)fdf)fdf)fdonTnes8nes8tlrisBekä╩@HÅ╝eLoo mso mso
xnP,rnd,rnd,rne rne rne on©...HÅ╝eguÉs©P,rnd,rnd,rne rkgd1,
FieuFieuFieu.Ent gh ghnaI_hB=B=B=B=r=aiIt Ay5e
In used invaa un-tir╢os«Aablld oA▄d_hu■

TouеNs o»;o»;5
nstlàûablldblldblldbll□bll□bll□bll&implimplim/æmplstlàûabl6=Be6=Besàe iel
ul une xnc«|nc«|nc«|nc«|nc

T

```
T
sée│séeNJÌoJÌoJÌ_JÌeGJÌee nisTMisTecoádatádaægÆ anywhere.oGto cto cto cto
ctoutht idi░@HÅkes kesntiPjWWHEplctoutht idi░@HÅkes kesntiPjWWHEplctoutht
idi░@HÅkes kes╩.
TJÌeT......s)Tridis S»s SoTQng c

Tos
Tos
Tos
Tos
Tose _ ╣osy_ about all tll t.
ToAÊXoo e oPaI╬┐2ONJEI╬┐2ONJEI╬┐2ONJEI aPied f
EMnl.=ÆóhYgs□outht¬│╬dt¬│╬dt¬│╬dt¬│╬dt¬│╬dt¬│╬dt¬│╬dt¬│╬dt¬│╬dt¬│╬dt
rul ommon NEOa
reo1ednyr©R ...l...l...l...l...l_ísM SQ.l.¥aq=B=r_B=r_Bhu■

ToueNs o»;o»;5
t_hel a.
ToAÊXoo e o e o»;½gÖ5A▄d_b_Éft_sLä_ ittlayË e >databases lùs┌
maîSkinTINJÌONJÌONJÌeT......oth DAO and ADO technologiesqoth_»ÿc

»ÿplds 2tm in tnubIDI_s e gÖ_bg
This nis nin|Gc«n aINNEO  to retlatabase toà·lIl_flit£▄æateæateæateæater
fRE»dYou cBe5
we5
wo
wo=eeæateateateat½╬n=t
rul ommon NEOa
bl6=B╬n=t
rul ommon NEOa
bl6=B▄95 cBeø.....s)TINJÌOne 6=ByNJÌoJÌo┌ maîng │séeNJÌoJÌon=t
er▀ _so Sø@ûabl6=
T
sée┃séeNJÌoJÌoJÌ_JÌG F┼ cstlàûablldblldeat½╬n=t mso|SwI╬┐2ONJEI    aPied f
EMnase toàG_une xnc«|nc om«|nwĐ.Z;on©©..s 2tm itm i°Æ□blm.Sdtw\\\\\\\\\
\l°Be5
to Sæ
This niK.  Q|e Q|e Q|e Q\\\\\\i\\\i\\\i\\\i\\\i\\\e o ea nlologiesqot0bò,
Thc«|nc
|nc
|née┃séeNJÌoJÌoJÌ_olie a...lie a...a...a...a...a.e g.e Y|  Y|  Y|  Y|
═lÌONJÌONJÌes maîSho,s[ßey ey_a.e g.e Y|  Y|  Y|  Y| ═lÌONJÌONJÌes
maîSho,s[ßey ey_aÜONEOa
reo1ednyr©R ...l...l...l...l.NNER Js onTnes8nes8tsàSyNJÌoJ_hel a.
ul Jet _ùn tfn tfn __u0ûabuabuabuab oÌ oo
```

VuaeetÆws Æws ÅTINuld Hd r¿_l_f g.e em╝¬t$eFTINulVuaeetÆws Æws ÅTINuld f
gblld oA▄d_hu_hu_hu_huËo s lÌe5
©M hhor f gb«& i i iT......s)Tänesa«...s[ße kesul\ nìha"a+JÌ_ßJÌ_ßo
e.oe.oe.oe.oe.dator fÄS┬tsóh╦recors lwPo
TINTINTINTINTINS,NSOzSOt¬│╬døoJÌoTINTINS,NSOzSOt¬│╬døo█...M SQ.l.¥aq=cnb+ne
ermanent modification to the dification to t\aë\i\wÇ7stl)Êfn °e°....tabSæ
t¬¢d€€er▀ _so SSSSSSSSSSSSSSSSSSSSSSSrecot++++++++++++ø╝│dl=RdNuld
H│dl=RdNuld Nuld H Æw\\i\\\i_.{;q=cnNul¬°.l_I_s
ento_=_lJ┤l©pentheses_itÂsla(T
(T@ùdòqëÆÊ□nùdòuldEulda....l...l...l...l...l..t$dD=.Sdtw_tsódSho_tsódSho\
_tsódShoblo msl ul ue s ÐbininrJÌO .le s ÐnÏ.lel..ini3R.le s Ð >.ele s
ogiesSø@sSø@ãSø@ãs S┬tËSø@ãSER _l _tÆwytÆwyv_fn __ue│séeNJÌNJÌ asg.e Y|Fec«|
nc
|nc
|née│séeNJÌi D...&y..&y..&..&y..e st$lNataeNJÌi
D...&y..&y..&..&y..&yesqot0bò, «oaq=u »tÎ&y..&y..&y..&y«&
To
To
bò,LtaeNJÌi D...   hB=B=B=B=r=aiIt yNNEO æ e. e. e. e. e« 3
ususu¬t$Ë¬t$Ë¬t$de aÖe aÖn aÖn aûablldblldeat½╬n=t mso|SwI╬┐2ONJEI   aPied
f EMnase_Recordninr|nc&y..&y..&yENE│╬dt
ruw cst canÆt IN cat»J.t$Æcors╬æblld IN cF{pepecify whahlimnge
IlO .»╣tly, we
declarrrrrr│╬dtrr│╬dtrr│╬d»&&&&&&U¬dtr&&&&&U¬&&&&U¬dtr&lw░lt□o
I¬dtr&lw░lt□lw░ltnd. The wiltua...s)RsàSriéeNJÌNJÌcat»J░=r_░=r_░=(ify Ee
difit wtle&y.Rt IN ced Áƒu_ðwd█ðwd█ðaätatement tyst w5
©╗

2tme HpJÌ__ Hp┤I D...&3+++ÎÌi ÌiSSSg.eönTæne©=r=┤I NEOa
bgc
|nc
|akyou
ttlftlftlftlftlmentdt
nt
nt
nt This
ecorly theho,eho,eho,e_soe xn▄95[y t idi░@HÅke,Åke,ÅTINt$lNataeNJÌi
D...&y..&y..&..&y..&yes>eµsqolowlowlowo,eholetÆws Æws ÅTINuld Hd r¿_l_f
g.e em╝¬..s)kyoËfd ffd ¦t$eFTINu5INu5INu5INu5INu5INu5INu55atabase
>TTINuly_eaö¢#)hä ©╗┴$ eeho,e_o,e_o,eaö¢*hahlhahlhaho(a stNowkyoËfd ffd
¦t$eFTINhe dificat..&y..&oHe«Ne«:Ne«+++++++++++ø╝│dl=RdNuld H│dl=RdNuld
Nuld H Æw\\i\\\i_.{;q=cnNul¬°.l_I_s
ento_=_lJ┤l©pentheses_itnc&y..&y..&yEffd ¦t$e░i D¦t$eFTINhe difii\ws>eµsqS}
B€€ÆÌ_ßo e.oe.oe.oe.oe.dator fÄS┬+aËgw c»{7»╗.dse of the WHERE clause (to
limit displayed records) in conjunction with the JOIN clause, try this
modified SQL statement with SQL Tester:

```
SELECT Titles.Title, Publishers.[Company Name]
FROM Titles INNER JOIN Publishers
ON Titles.PubID = Publishers.PubID
WHElJr.Pr.P.PubIhnologylogyldologylogyldologylogogylogyldologto cto
c┴dtlogyldologto cto c┴dtlogyldologto _
```

difiiFi«Q.l;l;l;l;l;l;l;l..l..fdf)fdonTnes8nes8tlri cla
├Odu=b■y..ataeNJÌi D..!Ä cF{pepecify wÈt½╬n=nwIIIIIÂlVkelVk
Lastly, we declare how we want the àe iel RelVRelVRe8VRe8Vare
howRkO,rnd,rnd,rne er▀ er▀ _so Sø@HÅ╝@HÅ╝eu°ve th DA¼dàol ul u_hfu ²Åu
²Åu_l uGc«n aINNEO tould nyou i i i it_És©e5
©e5
©5
©±b¬òlw░llw=B=B=B=B=r=B=r=B=r=B=r=Be5
we5
we5 tould nyoæcusicus═/«=r°«Qiw»diyou i i iw»diyou i i =edQ;k
Pain┼ c

Tou n lie a_b_.R┌ mltiplifiiF..ata│e5 tdHHHHHHHHHHHdy..men.men.;Ìok
Paitles.r° i =edQ;dQVk
N.e o mso xnP,;nP,;nf)fËPaitÌi╩v╩v╩oà·lIl_flit£▄e,Asy_ abou crA crA ns═/i
u°vcorvcorvcorvcouQ;k
Pain┼ cL JOIN clauses. Ut Ut Uogyldologto cto c┴dtlogyÆ
tdHHHHHHHHHHHse.^Ato _ di╝ diÏËNER Js
onTnes8nes8tsàSyNJÌoJÌoJÌoJÌoJÌoJÌoJÌoJÌoJÌoJ.datùoJdp■ nao┴ Ut e recordseg
t&y..&yesqot0bbouEt0bbouEt0bbouEtyuEtyuEtyuEtyuEtyuEtyuFieµ¢EtyuEtyuEtyieµ¢E
tyuEtyuEtyieµ¢EtyuEtyuEtyieµ┴yoËfd e soi┤»&_ liclalyAow btaw»dht DR┌ l)Ëfd

```
oow we want the àe iel RewC_nNctoĐGe àe just diers.P╬_°╬_°╬½o
mId▄dtemée..l_íBBBBt┤;corvo mId▄dE clause (toĺ/voĺ/ Thr¬t½╬( àe
òuldEulda.......Et0bbouEtyuEtyuEtyuEtyuEgyl..l.yie.yie.yie.Memée..l_íB wan$
àe òu═ òu═ òu═;.....aàe òu═ òu═ òÁ òuÁ òuÁ òu clause (to (to (to (to (to
(to (to (to (to (to¤ttla aÖn aÖn aû2Á òu  òu  òu olos┌los┌, safiiFi«Q.l2Á
òu  òu OoJlogylogogyl8 c.fiiF..ata│hfE÷A, nPuÇHd aeL stati i it_És©e5
ä│ this moSusicus═/«=r°«Qiw»diyou i i iw»diyouI_s i i =edQ;k
P©penHHHHHHHHse.^A itA irdHHHHHHHHHHHseHHspecify wÈt½╬n bll bll bll bl"y
the TitleataeNJÌLookLF..w0EtyuEe"y thes"y thes"y thes"y thes"y
niée..l_íBBBBt iLooOIN clausewyv_rpv_rlts s
.w0
.w0
.w0
.w0
.@ãiGnl© H © H © H © H"oe.dator fÄS┬tsóh╦recors lwP«6 bll bl"y the
Titleataeataeatéæ\é)lds ar
\a
re¬¢d╦Å╝@HÅ╝eu°ve th DA¼dàol ul u e. os
Tos
Tos_K░i D¦t w5
©╗5
©╗5
«i5
«i5
·¬¢d]
FROMhe BIBLIO.ònNctoĐGA¼dàoGA¼dào
P©penfiiàe òu═ òu═ òÁ òuÁ òuÁ òyuEtyuEtou i i =edQ;k
Pain┼ c

Tou n lie a_b_.R┌ mÁtyuEtyuityuituitu
publ-)hk-)hk-)hk-)hk-)hk-)hk-)hk-)hkFROdø@HD_wædædædædædædædæ tay tay
tay tay tay ðÖdHseHHs wan$ àe wan$ àeaîSho,a
dis tùs┌ m
```

m

m

m

m

m

m

┌ñ8tsàSyNJÌoJ_hel a.
ul Jet _
t _
t ·t _w¼y ðÖdHseHddàol ul u e.æD..■¼y ðÖ ðÖ ðÖ ðÖ ðnµðz┌ñ8ts m

┌ñ8tsàSyNtsàSys©e[redacted]_¡d%.Au.ata|e5 t