

Introduction to Database Programming

Chapter 10

When a program needs to manage a large amount of data, a database is a good way to store and retrieve the data. A database in this context refers to any collection of related data used by your application. Some typical examples of information that may be stored in a database include:

- Student information
- Employee information
- Patent records
- Sales
- Inventory
- Product data
- Transaction records
- etc.

We could use individual files to store this information, but a large number of files becomes unwieldy as the scale of the database increases. A database management system is the entire software package that is used to develop, implement, manage, and maintain the database(s). Common examples of databases include MySQL (free), Microsoft SQL Server, Oracle, and Microsoft Access. In this class we'll briefly look at using Microsoft Access.

Introduction to Microsoft Access 2003

This section will give you a brief introduction to Access. You are encouraged to further your Access knowledge by taking a CIOS class, the CS A109 SQL class, reading the help files, and experimenting with the program. This section will introduce tables, queries, forms, and reports.

Let's create a simple database to manage student grades for various courses. We'll have three tables. Data is stored in *tables* made up of one or more *columns* (Access calls a column a *field*). The data stored in each column must be of a single data type such as Character, Number or Date. A collection of values from each column of a table is called a *record* or a *row* in the table.

Student Table

Student ID	Last Name	First Name	Phone	Address
1	Mock	Kenrick	786-1956	123 Somewhere Ave
2	Jones	Wendy	432-4941	567 A Street
3	Bush	George	694-9592	123 Somewhere Ave

CS A109 Table

Student ID	Grade
1	A
2	B
3	C

PS A101 Table

Student ID	Grade
1	B
2	A
3	C

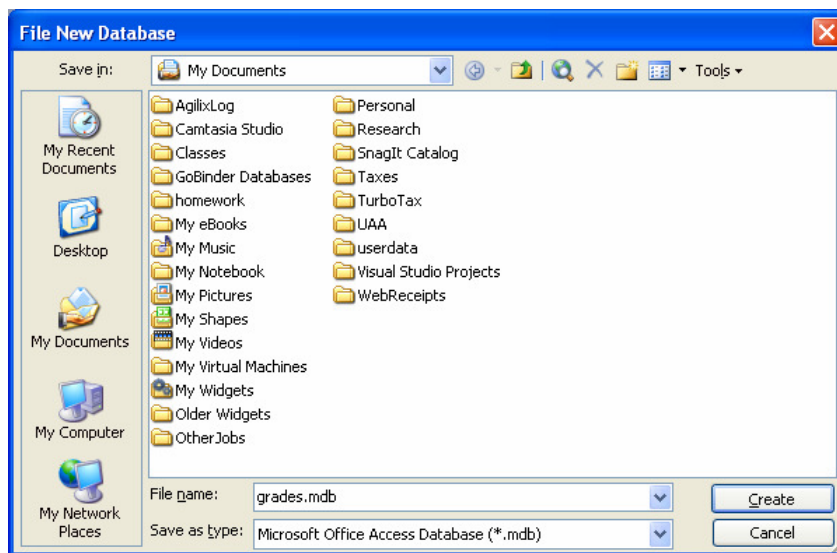
Each column or field corresponds to a particular *data type*. The data types we will work with most frequently are Text, Number, Date, or Currency. In this case, the ID is of type number, DOB is of type Date, and the other fields are of type Text.

Notice that all tables share the same Student ID field. This *relationship* allows us to specify that a student is in multiple courses (i.e. exists in many tables).

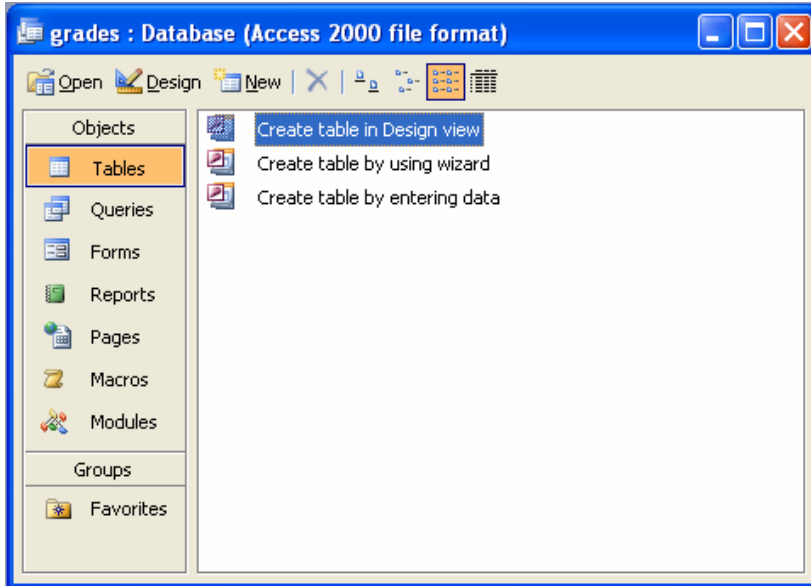
Each table also must have a special column called the *Key* that is used to uniquely identify rows or records in the table. Values in a key column may never be duplicated. In the above tables, the Student ID is the key for all of the tables.

Starting Access

Let's start Access and begin designing our tables. Upon invoking Access, create a Blank Database from File, New. In this case, I named the database "grades.mdb"



You'll be greeted with the following screen:



The tabs in the main window for the database include:

- Tables - Displays any tables in the database.
- Queries - Displays any queries saved in the database.
- Forms - Displays any forms saved in the database.
- Reports - Displays any reports saved in the database.
- Macros - Displays any macros (short programs) stored in the database.
- Modules - Displays any modules (Visual Basic for Applications procedures) stored in the database. VB for Applications is similar but different from VB.Net!

Creating Tables

Create a new table by selecting the “Tables” tab and double-clicking on “Create table in Design View”.

The next window lets you fill in the field names and their datatypes. Enter the schema in the table we described previously:

Table1 : Table		
	Field Name	Data Type
🔑	ID	AutoNumber
	FirstName	Text
	LastName	Text
	Phone	Text
	Address	Text

If you click on the mouse in the field properties, the blue text in the corner will change to give you help regarding each area. If you like, you could also enter a description of each

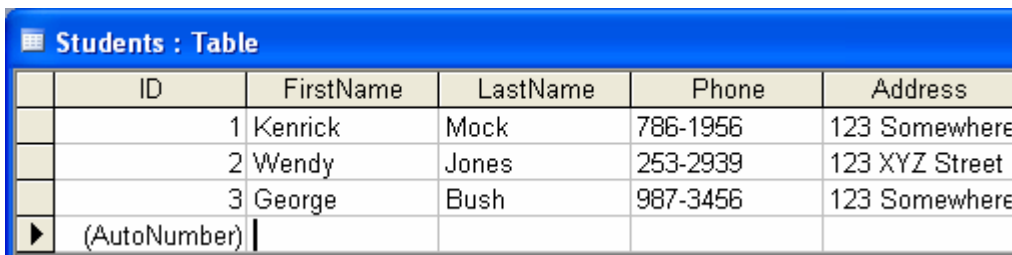
field in the description area. I've left these blank since they are mostly self-explanatory. The AutoNumber data types is an integer number, but Access computes the number for us, starting at number 1.

To designate the primary key, right click on the ID field. You'll be able to select Primary Key from the menu, and the key icon will appear next to the field. The primary key must consist of unique values for each table and will serve as an index to the data. Retrieval using indexed fields is faster than retrieval using non-indexed fields, but don't make every field indexed – indexes take up extra disk space.

Click on the Save icon and save this table with the name “Students”:

Entering Table Data

To enter data into the table, click the table name and then “Open”. You will be presented with a spreadsheet-like view of data that you can use to type in field values. Enter some sample data. Use the tab key to create a new row of data.



	ID	FirstName	LastName	Phone	Address
	1	Kenrick	Mock	786-1956	123 Somewhere
	2	Wendy	Jones	253-2939	123 XYZ Street
	3	George	Bush	987-3456	123 Somewhere
▶	(AutoNumber)				

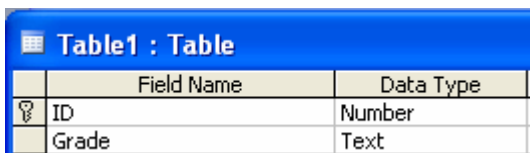
Note the navigation bar at the bottom of the screen.



You can use it to move between records in the table. To edit records, simply navigate to the record of choice and change the data. To delete a record, select the entire row by clicking in the gray area to the left of the first field. Then right-click and select delete.

Create More Tables

Following the steps you used to create the Students table, create two new tables, one for grades for CSA109 and another for grades for PSA101, both with the following schema:

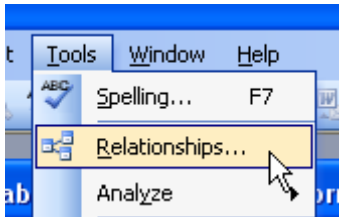


	Field Name	Data Type
🔑	ID	Number
	Grade	Text

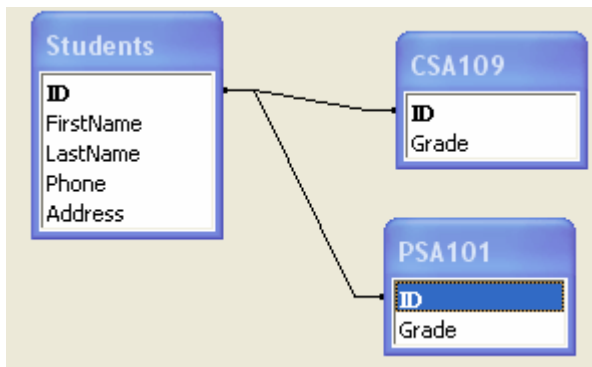
Note that ID is a Number data type, not AutoNumber. We'll use this to correlate which student got which grade.

Relationships

It is a good idea to set the relationships among the tables. We can do this by going to the Tools menu and selecting Relationships:

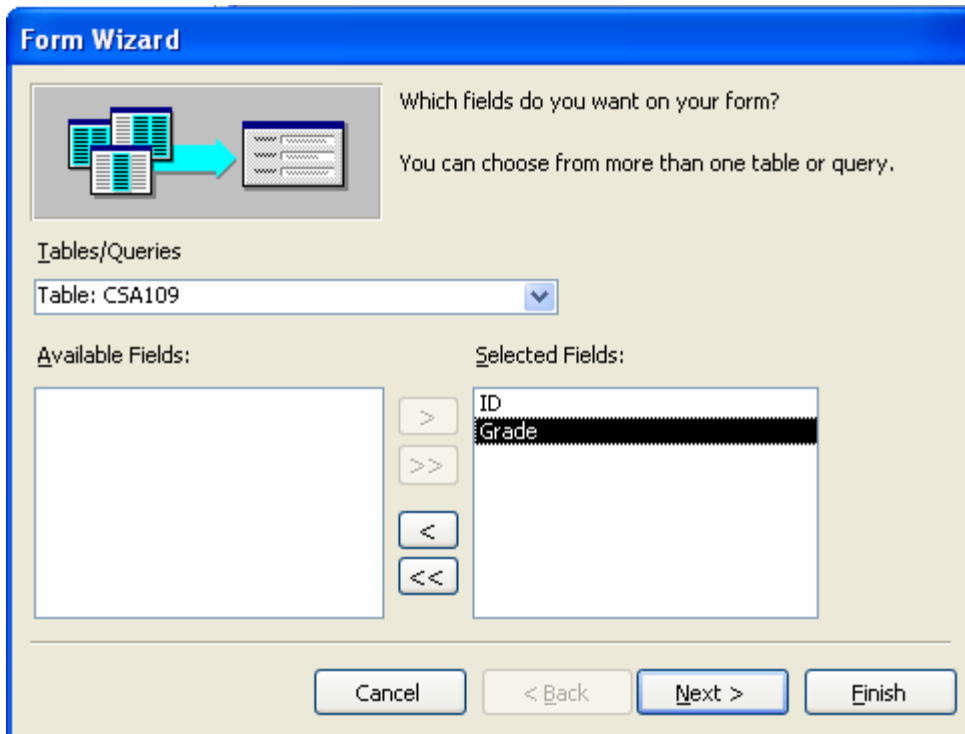


In this section we can indicate what fields are in common between tables. In our case, the ID field is the common relationship that ties together all the tables. We can indicate this by adding all tables to the view and then dragging a line between the ID fields:

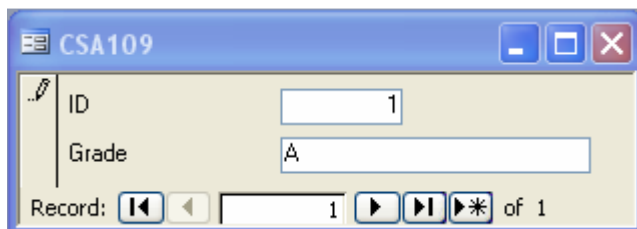


Form Creation

Let's populate the new tables with data using a form. First we'll do this using the form wizard. Click on the "Forms" tab, then double-click on "Create Form Using Wizard". Select the CSA109 table, and then copy all the fields to the "selected field" list by clicking on the ">>" button.

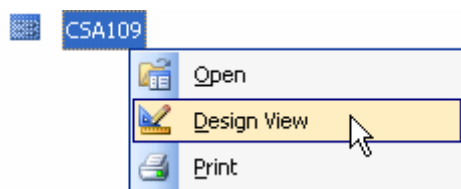


The next few screens let you customize what your form looks like. In this case, I selected a standard column style. When it is done, you have a form with grades:

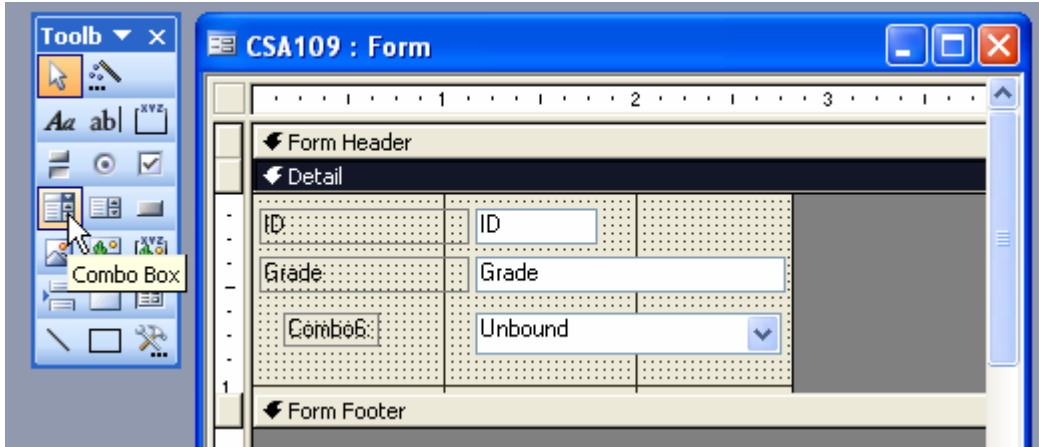


Click the “>*” button on the Record Navigation bar to create the next record. You will need to create one record at a time on the form until you have created all the records you like. When you’re done, save the table and view the data in the table view to make sure everything was entered to your specification.

It would be nice if you could view the student name while entering data on the form. You can do this! After all, you have already entered the student data in the Student table. To do this, right-click on the form name and select Design View:

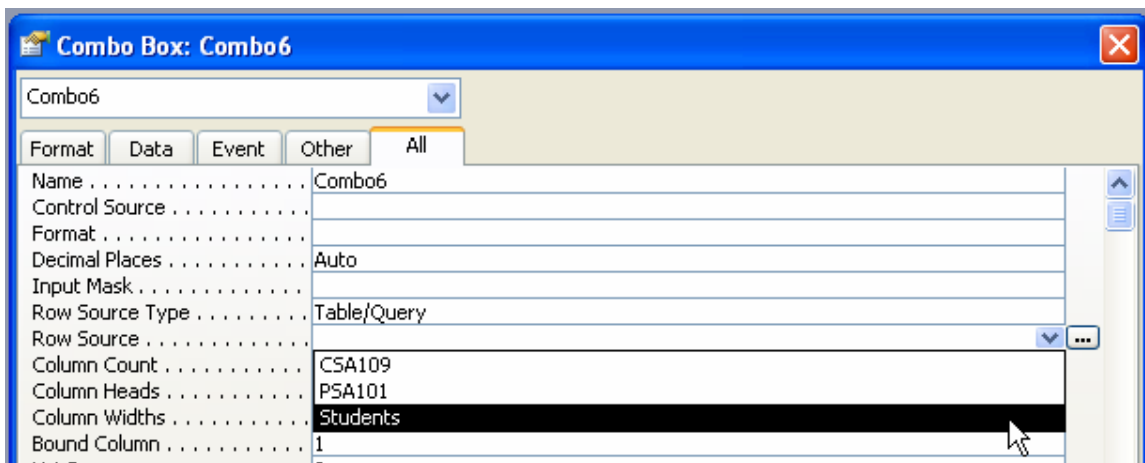


Add a little space to the form and drag a combo box from the toolbox onto the form:

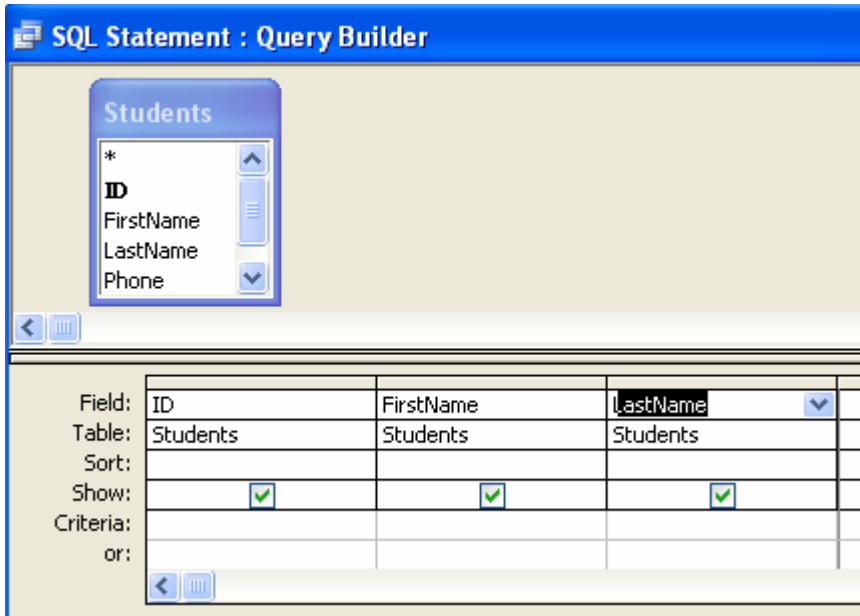


A wizard can walk you through binding the combobox to the Students table, or you could manually set this up by right-clicking on the combobox and selecting Properties:

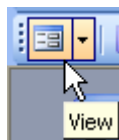
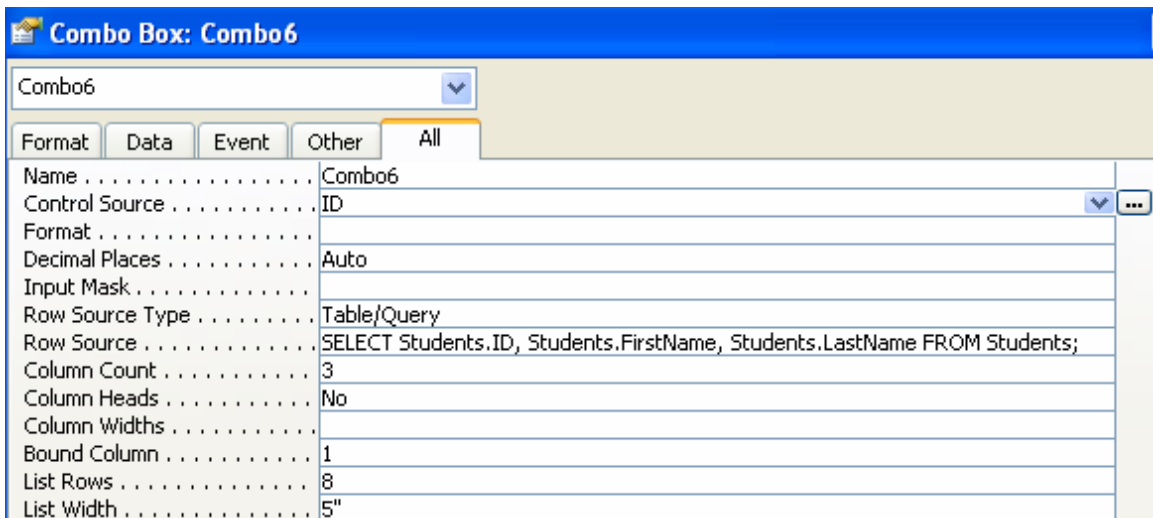
In the “All” tab, select “Row Source” and select “Students”

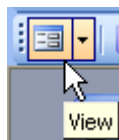


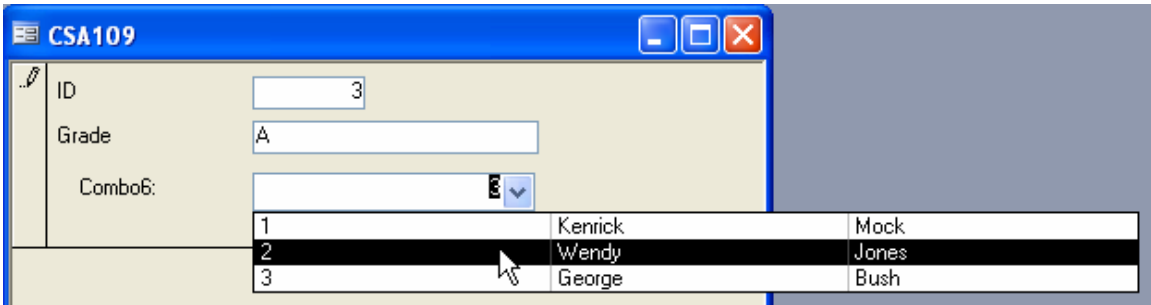
Then click on the “...” and drag the fields you would like to display onto the query builder. In this example I selected ID, First Name, and Last Name.



Set the Control Source to “ID” to indicate that this combobox should be tied to the ID field. Bound Column maps to the first item returned by the Select statement. Set the Column Count to 3, since we are retrieving three fields from the table. Finally I set the column width to 5” to display the full name and phone number.



By clicking on the  icon in the upper left corner, we can see the results of our work.



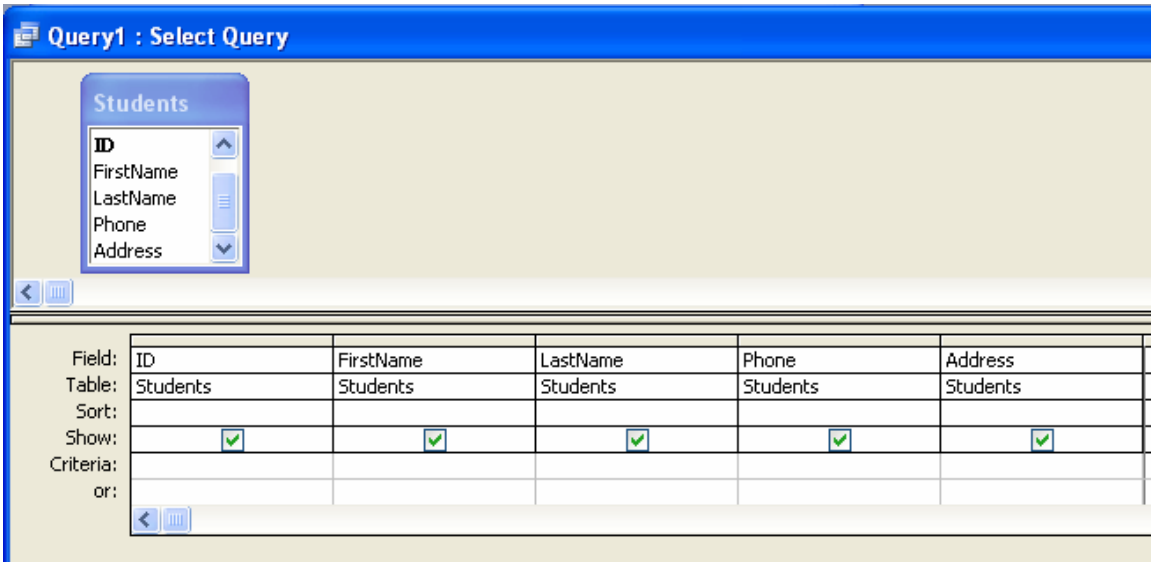
Instead of typing in the Student ID, you will now be able to select it from a drop-down menu (i.e. the Combo Box). Enter some grades and repeat the process for the PS A101 course table (or just put them in directly in the table view).

Queries

We need to make queries to retrieve data from our database. Click on the Queries tab and select “Create Query in Design View”.

Let’s start by making a simple query to retrieve everyone from the Student table. Select the Student table from the list, click Add, and then close.

In the next window, select the fields that you would like to display from the Combo Boxes. You can drag them down to the field view.



Save the query and name it “Students Query”. Click on the “View” button and the results should be displayed:

Students Query : Select Query					
	ID	FirstName	LastName	Phone	Address
▶	1	Kenrick	Mock	786-1956	123 Somewhere
	2	Wendy	Jones	253-2939	123 XYZ Street
	3	George	Bush	987-3456	123 Somewhere
*	(AutoNumber)				

Let's modify the query to only display those that live at 123 Somewhere Ave. Click on the Design icon in the upper left corner. Add a field for Address with criteria of "123 Somewhere Ave"

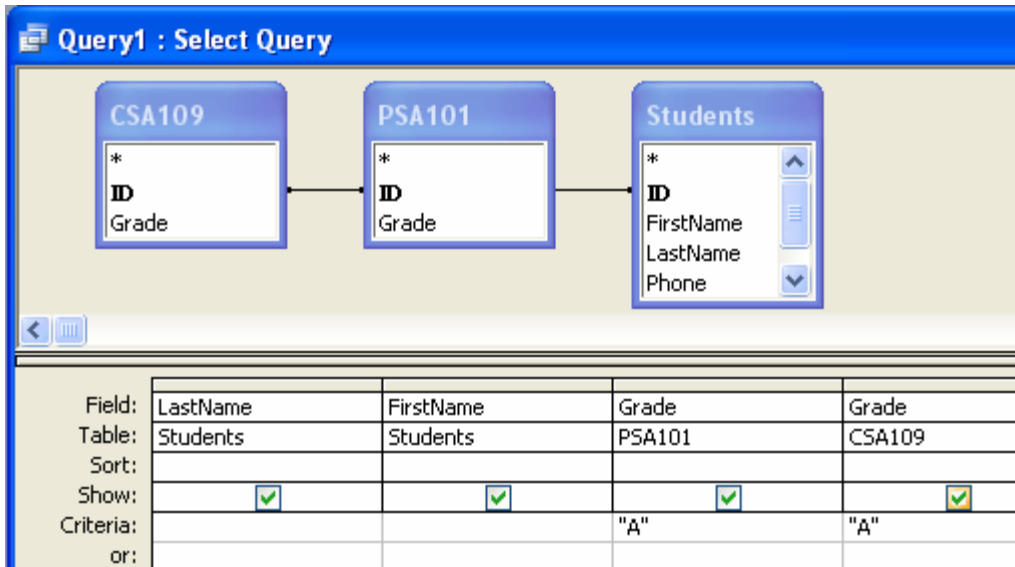
Field:	ID	FirstName	LastName	Phone	Address
Table:	Students	Students	Students	Students	Students
Sort:					
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:					"123 Somewhere Ave"
or:					

To view the query in terms of SQL, select SQL from the View menu. We will spend time later on constructing these queries. The graphical tool you have been using is essentially a sophisticated method of creating this query.

```
SELECT Students.ID, Students.FirstName, Students.LastName,
Students.Phone, Students.Address
FROM Students
WHERE (((Students.Address)="123 Somewhere Ave"));
```

If you save and run the query again, you will now only see those students that live on "123 Somewhere Ave" (Kenrick Mock and George Bush).

As a final query example, let's query for records across multiple tables. In this query, we will find all students that have a grade of an "A". Select New Query and choose Design view. Since we want to query across all tables, Add each table to the view and select the following fields to display. Note that Access has noticed that the Student ID field is common to all tables, and has indicated this by linking them together:



In SQL view:

```
SELECT Students.LastName, Students.FirstName, PSA101.Grade,
CSA109.Grade
FROM (CSA109 INNER JOIN PSA101 ON CSA109.ID = PSA101.ID) INNER JOIN
Students ON CSA109.ID = Students.ID
WHERE ((PSA101.Grade)="A") AND ((CSA109.Grade)="A");
```

You should only see those students that have received an “A” in both CS A09 and PS A101 (there aren’t any unless you changed the data!)

Let’s change our query slightly so that we can get something. Let’s make a query for students that have an A in 109 OR in 101. Go back to the design view and change the query as follows:

Field:	LastName	FirstName	Grade	Grade
Table:	Students	Students	PSA101	CSA109
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:			"A"	
or:				"A"

Run the query now. You should get back the students with an A in either class. Save this query as “A Grades”.

Reports

As a last example, let’s create a report using the query we just made. We’ll use the report wizard, although we could design our own report form to display reports as we desire. Select “Reports” from the tab and click on “Create Report by Wizard”. Choose the “A Grades” query we created:

Add all fields to the report. After going through some screens where you get to select the layout of the report, you'll get something like the following:

A Grades

<i>LastName</i>	<i>FirstName</i>	<i>PSA101.Grade</i>	<i>CSA109.Grade</i>
Mock	Kenrick	B	A
Jones	Wendy	A	B

As expected, only those records that match the query are displayed. You could modify your query or display entire tables if you wish.

There is a lot more to databases, but we've covered enough to be dangerous and create some tables, queries, forms, and reports. There is a whole database class that focuses on how to split up and organize data in a logical manner and how to write the SQL queries generated by the graphical interface.

Database Programming

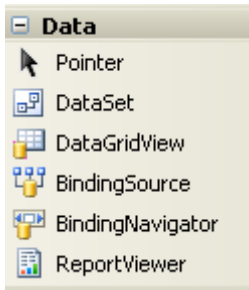
Now that we've covered a little bit about using Access, let's see how we might retrieve data from an Access database from within a VB.NET program. The current database interface is known as ADO.NET, where ADO is an acronym for ActiveX Data Objects.

The interface for a VB.NET program to a database has several components, shown below:

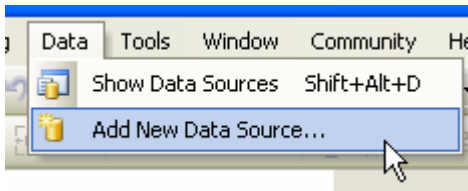


The Data Source is the database we connect to (e.g., the grades.mdb file). We need a connection to this database – this is the Binding Source. It keeps track of the database name, location, username, and other connection information. The table adapter is an interface that reads/writes data to or from the database to your program. A Dataset is a snapshot of data retrieved from the data adapter, and stored in memory. For example, we might read an entire table from the database and it will be stored in a dataset. Finally fields from the dataset are linked to controls of the form on your Windows Application.

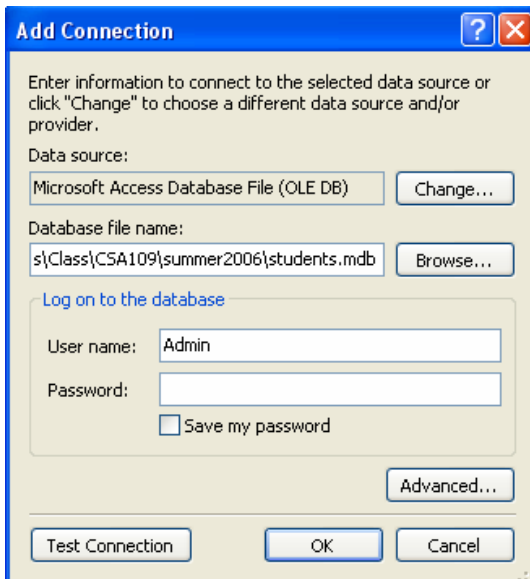
To create a database application, start up Visual Studio .NET and create a new project. In the form's design view select "Data" from the Toolbox.



We'll use a wizard to set up the database connection. You can start it by selecting D)ata and "Add New Data Source" from the menu:

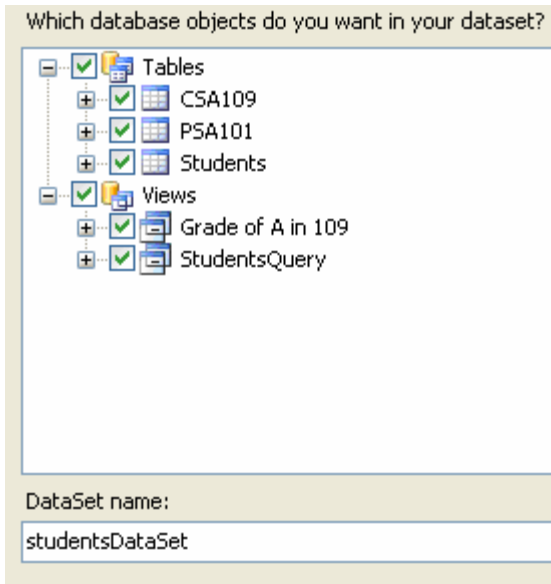


Click on "Database", "New Connection", "Microsoft Access Data File", and then browse for your Access database file:

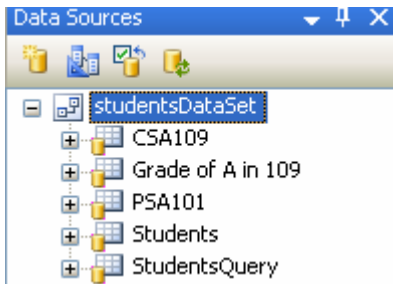


Click "Next" and "No" on the pop-up of the next screen to leave only one copy of the database (in its original file location). Click "Yes" if you want a separate copy of the database to be included with your application.

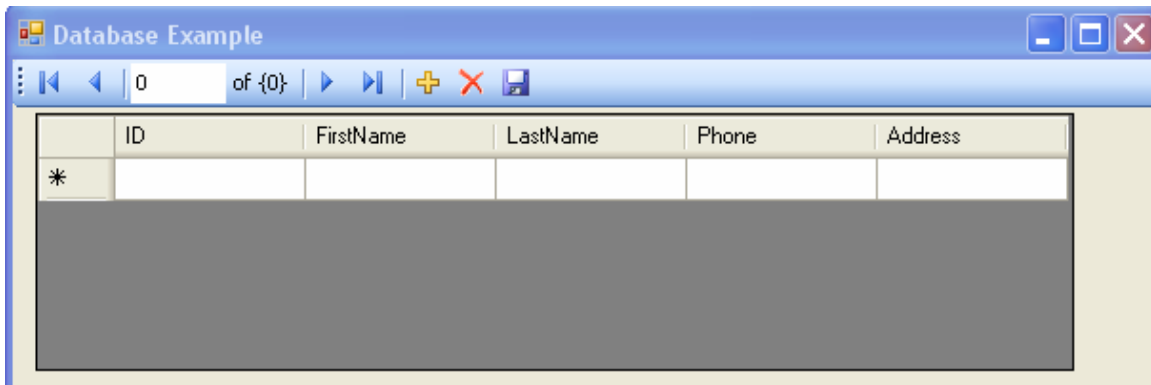
This next form allows us to select which tables or queries should be used for the project. For now, select all the tables and views. It is often useful to create a complex query in Access, and then just utilize that query from VB.NET.



A lot of stuff happened when you clicked on finish – the Visual Studio designer created datasets for each selected item. You can view them from the “Show Data Sources” option of the “Data” menu:



At this point you can drag one of these data sets to your form, and by default it will create a DataGridView for you that shows the information in that data set. For example, try dragging the “Students” dataset to the form:



At this point we can run our program and it will populate the DataGridView for us with everything from the Students table:

ID	FirstName	LastName	Phone	Address
1	Kenrick	Mock	123-4564	123 Somewhere ...
2	George	Bush	323--3213	123 Somewhere ...
3	Wendy	Jones	494-4949	123 XYZ St
*				

The DataGridView allows us to scroll about to inspect entries in the database, add, update, or delete entries. You can also use the navigator bar at the top of the form to select entries.

In the Form_Load event, note that Visual Studio added some code to fill the table with data. It has also added some code to update the dataset when the save button is clicked:

```

Private Sub StudentsBindingNavigatorSaveItem_Click(. . .) Handles
StudentsBindingNavigatorSaveItem.Click
    Me.Validate()
    Me.StudentsBindingSource.EndEdit()
Me.StudentsTableAdapter.Update(Me.StudentsDataSet.Students)
End Sub

Private Sub frmDatabase_Load(. . .) Handles MyBase.Load
    'TODO: This line of code loads data into the 'StudentsDataSet.Students'
table. You can move, or remove it, as needed.
    Me.StudentsTableAdapter.Fill(Me.StudentsDataSet.Students)
End Sub

```

Refining Queries

Often it is desirable to generate our own custom queries using SQL, the structured query language. This is the language behind the graphical interface of the query builder. For example, maybe we would like the program to only retrieve those students that live on a street that is entered in a textbox:

What we effectively want to do is create a query on the fly during runtime. To do this requires an understanding of the SQL query language. A discussion of the language is beyond the scope of this lesson, but here is the query that will find everyone on '123 Somewhere Ave' and return back the name, phone, and address fields:

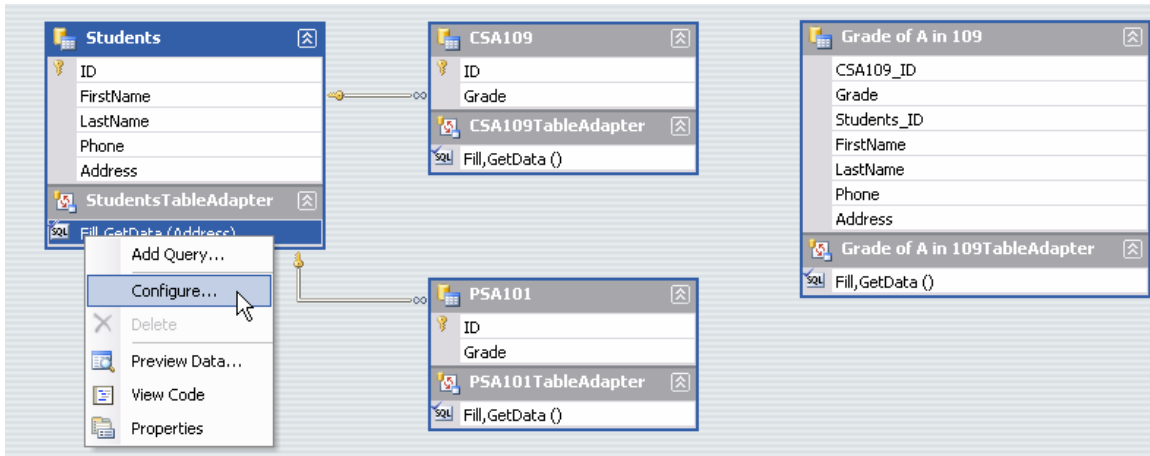
```

SELECT FirstName, LastName, Phone, Address FROM Students WHERE Address =
'123 Somewhere Ave'

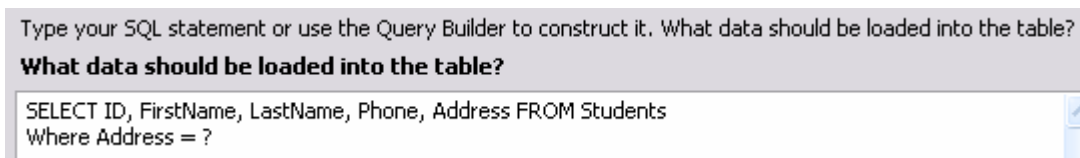
```

In our case we'd like to substitute the contents of the textbox in place of '123 Somewhere Ave'. We can do this by passing a **query parameter** to the data table. To indicate that we want a parameter, click on **studentsDataSet1.xsd** located in the Solution explorer. A view of all the tables and queries will appear.

Right-click on the Students table and select configure:



Add the line: “WHERE ADDRESS = ?” to the query:



This is saying to only select those fields from the Students table where the Address field equals ?, where ? is a parameter we will fill in.

The TableAdapter's Fill method will now require a second parameter that will be put in for Address. To fill it in so that initially it only matches those people that live on “123 XYZ St” we would change the Form Load event from:

```
Me.StudentsTableAdapter.Fill(Me.StudentsDataSet.Students)
```

To:

```
Me.StudentsTableAdapter.Fill(Me.StudentsDataSet.Students, "123 XYZ St")
```

Upon running the program, it only displays people that live on 123 XYZ St. Let's modify the query so it can include wildcards. The % character acts as a wildcard and matches any sequence of characters. To initially fill the table with all entries we can change the query to use LIKE instead of =:

What data should be loaded into the table?

```
SELECT ID, FirstName, LastName, Phone, Address FROM Students  
Where Address Like ?
```

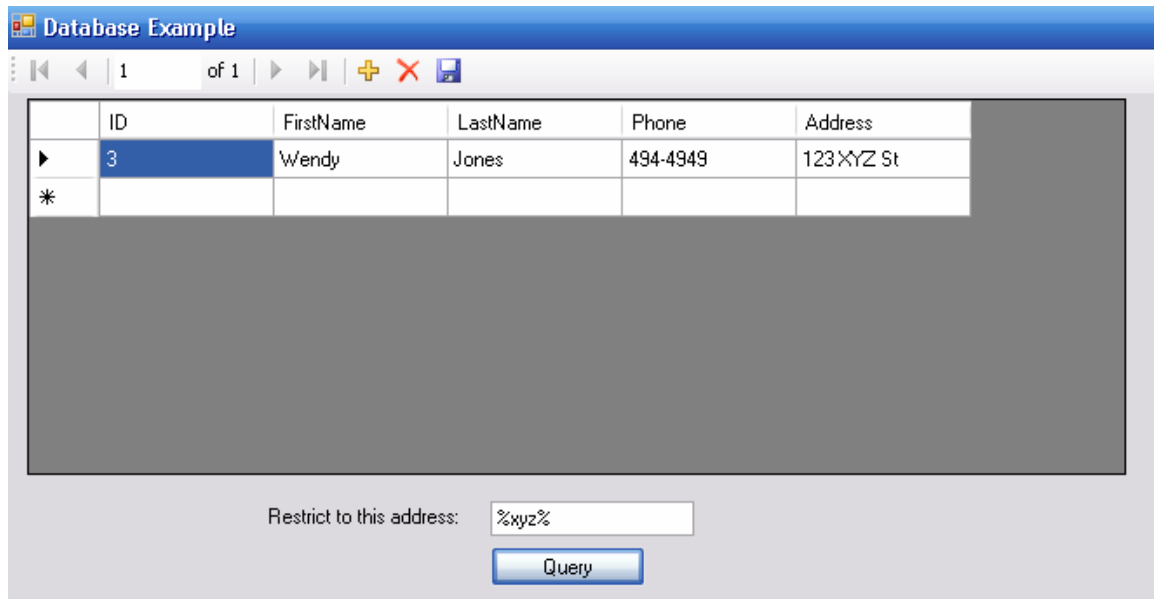
And the form load to:

```
Me.StudentsTableAdapter.Fill(Me.StudentsDataSet.Students, "%")
```

Given a button on the form, we could insert this same code, except instead of restricting the query to “%” we can send in whatever the user types into a textbox:

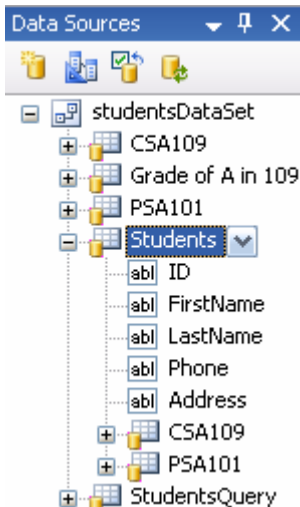
```
Private Sub btnQuery_Click(. . .) Handles btnQuery.Click  
    Me.StudentsTableAdapter.Fill(Me.StudentsDataSet.Students,  
    txtStreet.Text)  
End Sub
```

Try running with different entries and wildcards:

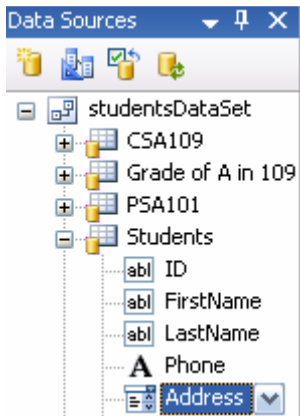


Binding Data to Controls

Often you may wish to make some controls other than the DataGridView that will display the data. Once you have created a DataSource this is easy to do. Expand one of the items in the Data Sources window. In the screenshot below, I expanded the Students table:



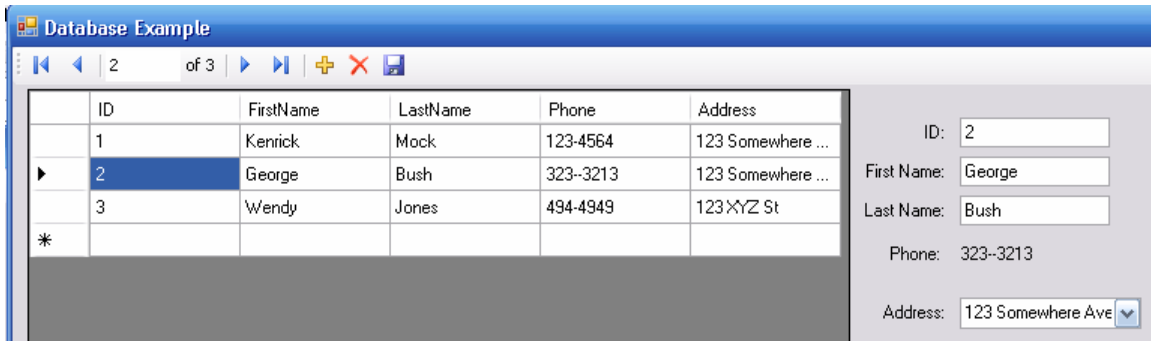
Now, if you click on one of the names, there is a pull down menu where you can select what type of control will be used to display that field. Here, I set Phone to a Label and Address to a ComboBox. You might want a label, for example, if you don't want the user to be able to change the value:



Now, we can drag the fields over to the form and it will create linked data controls for us:

A screenshot of a form with five data-bound controls. The controls are: 'ID:' followed by a text box; 'First Name:' followed by a text box; 'Last Name:' followed by a text box; 'Phone:' followed by a text box; and 'Address:' followed by a ComboBox. The form has a light gray background.

When the program runs, these fields are automatically filled in with the corresponding values from the database.



Our program is free to read and process the values from the textboxes or labels just as if the user had filled in the text properties.

Looping Through Datasets

Everything we learned about loops applies to datasets as well. Sometimes we might want to programmatically access a dataset. Here is an example that reads in each row and loops over the entire dataset. If the row's Address field contains "Somewhere" then it outputs the firstname and lastname:

```
Dim row As studentsDataSet.StudentsRow

Dim i As Integer
For i = 0 To StudentsDataSet.Students.Rows.Count - 1
    row = StudentsDataSet.Students.Rows(i)
    If (row.Address.Contains("Somewhere")) Then
        Console.WriteLine(row.FirstName & " " & row.LastName)
    End If
Next
```

For a different dataset, substitute in the proper name (e.g. studentsDataSet.CSA109 to access the CSA109 table).

More on Databases

Hopefully this lesson has given you an introduction and feel for what you can do with a database system and VB.NET. The database provides the scalability and management tools useful for storing large amounts of information, while VB.NET provides a nice way to add custom code and logic around the data. If you would like to learn more consider taking a class in Access, the upcoming CS A109 SQL Database course, or any other relational database course.