Unit 8 Databases in VB .NET

Structure:

8.1 Introduction

In the previous unit, we had a discussion about File handling techniques of VB .NET. We explored the various classes that support the File reading and writing operations.

This unit introduces the user to data access techniques used in VB.NET. It works with Visual Studio 2005 and higher versions. To understand this unit the user needs to be familiar with the knowledge of databases and its concepts. In this unit we are going to discuss how VB .NET supports with database connectivity. How to set a connection with the back end, and also to open a connection to establish the communication. We are also going to discuss the support of structured query language in the database connectivity and the role of data adapter and the data set.

Objectives:

After studying this unit, you will be able to:

- explain the usage of Visual Studio.Net in data access
- describe the Data Form Wizard of Visual Basic.Net
- describe the objects of ADO classes
- explain the usage of connection strings
- describe the role of Data Adapters in data access

8.2 Introduction to Data Source in VB .NET

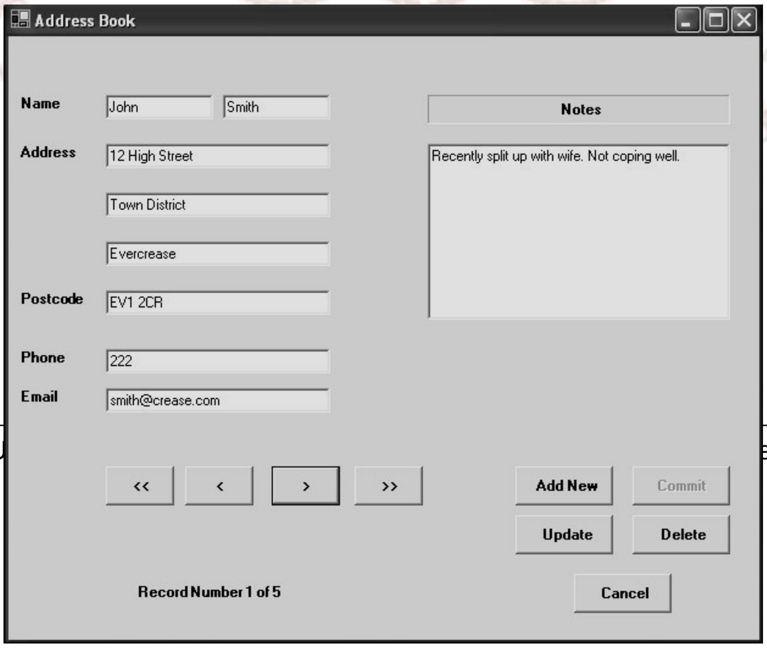
VB.Net allows you to connect and access database or a data source through many ways. The technology used to interact with a database or data source is called ADO.NET.

One or more files are used to store the information that makes up a database. Each entry in these files could have its own set of fields for storing data. Each field in a database primarily stores information on one characteristic or characteristic of the entity it stores. This is considered as **<u>the features of database</u>**

The ADO parts stands for Active Data Objects. Forming the foundation of the ADO Base Class are five other majorobjects:

- Connection
- Command
- DataReader
- DataSet
- DataAdapter

We are going to discuss the ADO.NET by creating a simple Address Book project. Here we need to understand how to use ADO to open up the database that you downloaded, and scroll through each entry. Now we are going to use a Wizard to create a program that reads the database and allows us to scroll through it. The wizard will create a Form for us, and allow us to add buttons to the form so that the data can be edited, updated, and deleted. The form that we have decided to design is depicted in Figure 8.1.



Name	John Smith
Address	12 High Street
	Town District
	Evercrease
Postcode	EV1 2CR
Phone	222
Email	smith@crease.com
Notes	Recently split up with wife. Not coping well.

Record Number 1 of 5

Fig. 8.1: Form design

8.3 Setting a Connection String /Establishing the connection using data set and data adapter

If you want to connect to the database you need the connection object. There are a number of different connection objects, and the one you use depends largely on the type of database you're connecting to. Because we are connecting to an Access database, we'll need something called the OLE DB connection object. OLE stands for Object Linking and Embedding, and it is basically a lot of objects (COM – Component Object Model objects) bundled together that allow you to connect to data sources in general, and not just databases. There are number of different OLE DB objects (called data providers), but the one we are going to use is called *Jet*. Others are SQL Server and Oracle. So place a button on your form. Change the Name property to btnLoad. Double click your button to open up the code window. Add the following line:

Dim con As New OleDb.OleDbConnection

If you have the free Visual Basic 2005 Express Edition, you may see a wavy line appear under the line of code. This is because you first need to add a reference to the Data Objects. Do the following steps

- Click Project from the menu bar
- Then click Add Reference
- From the dialogue box, select the .NET tab. Scroll down and select the System.Data item
- Click OK.

At the very top of your code window, before Public Class Form 1, type the following:

Imports System.Data

This will then allow you to work with the various objects in the Database section. Your coding window will look as the code below l

```
Imports Ssytem.Data
        Public Class Form1
                Private Sub btnLoad_click(Byval sender As System.object, -
                ByVal e as System.EventArgs) Handles btnLoad.click
                Dim con As new OleDb.OleDbConnection
                End Sub
        End Class
```

Whichever version you have, though, the variable con will now hold the Connection Object. Notice that there is a full stop after the OleDB part. You will then get a pop up box from where you can select OleDbConnection. This is the object that you use to connect to an Access database.

There are Properties and Methods associated with the Connection Object. We want to start with the ConnectionString property. This can take many parameters . Fortunately, we only need a few of these. We need to pass two things to our new **Connection Object**: the technology we want to use to do the connecting to our database; and where the database is. (If your database was password and user name protected, you would add these two parameters as well.)

The technology is called the **Provider**; and you use "**Data Source**" to specify where your database is. This should be entered on the same line, and not two as it is below. So add this to your code:

con.ConnectionString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source = C:\AddressBook.mdb"

Notice the two parts, separated by a semi-colon:

1st Part: PROVIDER=Microsoft.Jet.OLEDB.4.0
2nd Part: Data Source = C:\AddressBook.mdb

The first part specifies which provider technology we want to use to do the connecting (**JET**). The second part, typed after a semi-colon, points to where the database is. In the above code, the database is on the C drive, in the root folder. The name of the Access file we want to connect to is called **AddressBook.mdb.** (Note that "**Data Source**" is two words, and not one.)

But your coding window should now look like this:

```
Private Sub btnLoad_Click(ByVal Sender as Object,_
            ByVal e as System.EventArgs) _
Handles btnLoad.Click
Dim con as new OleDb.OleDbConnection
Con.ConnectionString = “Provider = Microsoft.Jet.OLEDB.4.0;
DataSource = _ C:\AddressBook.mdb”
```

This assumes that you have copied the AddressBook database over to the root folder of your C Drive. If you've copied it to another folder, change the

"Data Source" part to match. For example, if you copied it to a folder called "databases" you'd put this:

Data Source = C:\databases\AddressBook.mdb

In our code, though, **ConnectionString** is a property of the **con** variable. The con variable holds our Connection Object. We're passing the Connection String the name of a data provider, and a path to the database.

Self Assessment Questions

1. ADO stands for______________________________.
2. ________________ and__________________are the two property mandatory for the connection string.
3. ________________is a collection of bundle of objects that allow you connect to data sources.

8.4 Opening the Connection

Now that we have a ConnectionString, we can go ahead and open the datatbase. This is quite easy - just use the **Open** method of the Connection Object:

```
con.Open( )
```

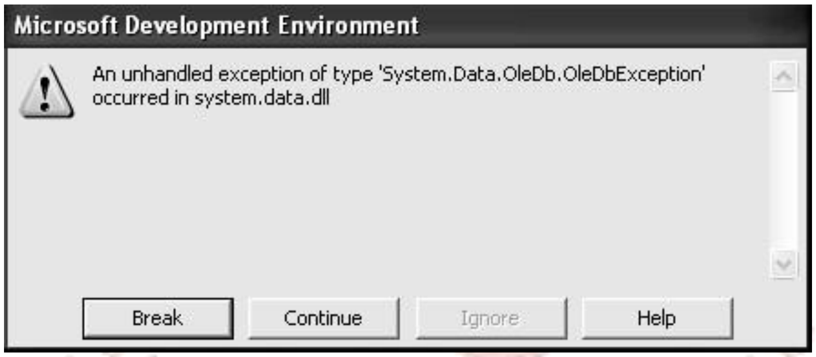
Once open, the connection has to be closed again. This time, just use the Close method:

```
con.Close( )
```

Add the following four lines to your code:

```
con.Open()
MsgBox("A Connection to the Database is now open")
con.Close()
MsgBox("The Connection to the Database is now Closed")
```

Test out your new code by running your program. Click your button and the two message boxes should display. If they don't, make sure your Data Source path is correct. If it isn't, you might see the error message shown in figure 8.2.


Microsoft Development Environment

An unhandled exception of type 'System.Data.OleDb.OleDbException' occurred in system.data.dll

Fig. 8.2: unhandled exception error

The error message is a bit vague, but what it reflects is, it cannot find the path to the database, so it can't Open the connection. The line con.Open in your code will then be highlighted in green. You need to specify the correct path to your database. Now that we've opened a connection to the database, we need to read the information from it. This is where the DataSet and the DataAdapter come in.

In the previous section, you learned how to set up a Connection Object. This was so that you could open a connection to the database itself. But that's not the end of it. The data from the database needs to be stored somewhere, so that we can manipulate it.

ADO.NET uses something called a **DataSet** to hold all of your information from the database (you can also use a DataTable. But the **DataSet** (and Data Table) will hold a copy of the information from the database.

The DataSet is not something you can draw on your form, like a Button or a Textbox. The DataSet is something that is hidden from you, and just stored in memory. Imagine a grid with rows and columns. Each imaginary row of the DataSet represents a Row of information in your Access database. And each imaginary column represents a Column of information in your Access database (called a Field in Access).

Now we can see the role of Data Adapter. The Connection Object and the DataSet can't see each other. They need a interface between them, so that they can communicate. This role is done by the Data Adapter. The Data Adapter contacts your Connection Object, and then executes a query that you set up. The results of that query are then stored in the DataSet.

The Data Adapter and DataSet are objects. You set them up like this:

Dim ds As New DataSet
Dim da As OleDb.OleDbDataAdapter
da = New OleDb.OleDbDataAdapter(sql, con)

8.5 The Data Adapter

The Data Adapter is a property of the OLEDB object, hence the full stop between the two:

OleDb.OleDbDataAdapter

We are passing this object to the variable called **da**. This variable will thenhold a reference to the Data Adapter. While the second line in the codeabove sets up a reference to the Data Adapter, the third line creates a newData Adapter object. You need to put two things in the round brackets of theObject declaration: Your SQL string and your connection object. OurConnection Object is stored in the variable which we've called **con**. (Like allvariable you can call it practically anything you like. We've gone for something short and memorable.) You then pass the New Data Adapter toyour variable (**da** for us): **da = New OleDb. OleDbDataAdapter (sql, con)** We need something else, though. The **sql** in between the round brackets isthe name of a variable. We haven't yet set this up. We will discuss theseconcepts in section 8.6. But bear in mind what the Data Adaptor is doing:*Acting as a go-between for the Connection Object and the Data Set*

DataAdapter: You can retrieve, insert, update, delete data by Updating DataSet using DataAdapter

The OleDBDataAdapter class is what you need if you're dealing with a database like Oracle or Microsoft Access. Make a new SqlDataAdapter object and set its data source to be an MS-SQL server. The object of command and communication is sent between users via parameters. Create a DataSet instance from the dataset you just made.

The time has come to call upon SqlDataAdapter. The DataSet object supplied to the fill() command.

Code for SqlDataAdapter.Fill() method shown here. The Data Adapter can Fill a DataSet with records from a Table.

You only need a single line of code to do this:

da.Fill(ds, "AddressBook")

```
[illegible]e[illegible]on
[illegible]ing = [illegible]a Source=(LocalDB)\v11.0;
[illegible]ame=C:[illegible]s\grthamad\Documents\student.mdf;Integ[illegible]d
[illegible]Conn[illegible]out=30";
SqlDataA[illegible]v SqlDataAdapter("Select * from st[illegible]ord",con);
student D[illegible]v stude[illegible]ataset[illegible]Fill(ds.[illegible]
```

[illegible]base conn[illegible]nd exe[illegible] comm[illegible]s ellab[illegible]e. [illegible]reate[illegible]nnection in [illegible]way[illegible] did w[illegible]e dat[illegible]d[illegible]a adapter [illegible]com[illegible]and[illegible] dat[illegible]as[illegible]ommand in [illegible]e ca[illegible]ata[illegible]ses and [illegible]case of MS SQL data[illegible]n't need any values[illegible] if you only [illegible]e use the Exe[illegible] use the Exec[illegible]mm[illegible]

Read data using data reader explained here:

.NET Framework provides two types of Connection classes

Communication using command and data reader.

First we establish a connection, secondly we will execute the command and then read the data

If you want to connect to the database you need the connection object. There are a number of different connection objects, and the one you use depends largely on the type of database you're connecting to .

So place a button on your form. Change the Name property to btnLoad. Double click your button to open up the code window. Add the following line:

Dim con As New OleDb.OleDbConnection

The technology is called the **Provider**; and you use "**Data Source**" to specify where your database is. This should be entered on the same line, and not two as it is below. So add this to your code:

con.ConnectionString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source = C:\AddressBook.mdb"

The first part specifies which provider technology we want to use to do the connecting (**JET**). The second part, typed after a semi-colon, points to where the database is

Let us discuss about the different Steps to connect database illustrated here.

1. Select TOOLS Connect to Database;

2. Select a server/database name;

3. Click on Test Connection button check connection;

4. Click at the Add Project Data Source link;

5. Select Database as the data source type;

6. This opens the Data Source Configuration Wizard;

7. Add a DataGridView on the form;

8. Click at the Choose Data Source combo box;

9. Choose Dataset as the database model;

10. Choose the connection already set up;

11. Save the connection string;

12. Choose the database object, Customers table Finish button;

13. Choose the database object, Customers table Finish button;

Let us see the different advantages of database in VB.Net given here.

The, database systems provide data storage integration as well as data protection, They are widely used in both large and small organizations due to their many advantages over conventional file-based data storage systems, it offers dependable connectivity to a wide variety of data stores, including OLE DB and XML-exposed databases and SQL Server. There are some shared features between ADO and ADO.NET, but the two are fundamentally different and built on very different pillars. It also introduces a new way of getting a single value from a query's results when you expect only one row and one column to return. The ADO.NET command object has an ExecuteScalar method which returns the first row and column's value from its associated query.

Self Assessment Questions

4. ADO.NET uses Dataset to hold all of your information from the database. State [True/False].
5. ____________________ is the interface communicate between the connection object and the dataset.

8.6 Structured Query Language

SQL (pronounced SeeKwel), is short for Structured Query Language, and is a way to query and write to databases (not just Access). The basics are quite easy to learn. If you want to grab all of the records from a table in a database, you use the SELECT word. Like this:

SELECT * FROM Table_Name

SQL is not case sensitive, so the above line could be written:

Select * from Table_Name

But your SQL statements are easier to read if you type the keywords in uppercase letters. The keywords in the lines above are **SELECT** and **FROM**. The asterisk means ‘All Records’. Table_Name is the name of a table in your database. So the whole line reads:

SELECT all the records FROM the table called Table_Name
You don't need to select all (*) the records from your database. You can just select the columns that you need. The name of the table in our database is **tblContacts**. If we wanted to select just the first name and surname columns from this table, we can specify that in our SQL String:

SELECT tblContacts.FirstName, tblContacts.Surname FROM tblContacts
When this SQL statement is executed, only the FirstName and Surname columns from the database will be returned.

Because we want to SELECT all (*) the records from the table called tblContacts, we pass this string to the string variable we have called sql:
sql = "SELECT * FROM tblContacts"

So add the following code to your database project:

```
Dim ds As New DataSet
Dim da As OleDb.OleDbDataAdapter
Dim sql As String
sql = "SELECT * FROM tblContacts"
da = New OleDb.OleDbDataAdapter(sql, con)
```

(If you're using the free 2005 Express edition, you might see DataSet with a wiggly line under it. This is because you need to set a reference to something called System.Xml.dll. To do that, click **Project > Add Reference** from the menu bar. The on the **NET** tab of the dialogue box that appears, scroll down and click on **System.Xml.dll**. Then click OK.)

Now that the Data Adapter has selected all of the records from the table in our database, we need somewhere to put those records - in the **DataSet**.

To elaborate on SQL let us know the features of SQL queries:
Multiple objectives can be met with just one SQL query. In addition to isolating and locating specific data, SQL queries can also be used to compute and highlight information, such as when automating administrative tasks. These transactions, fundamental to every database, are managed with TCL commands like commit, rollback, and save point. In addition to filtering, sorting, grouping, and joining data from multiple tables, SQL provides a wide variety of instructions for data retrieval from databases.

<u>Categories of SQL Queries:</u>

Let us try to understand the different categories of SQL queries, there are three categories
1.Data Control Language
2.Data Manipulation Language
3. Data definition Languagae

First one is Data Control Language (DCL) is utilized to find out and repudiate database rights and consents. Its main statements are GRANT and REVOKE. DCL is used to manage user access. The DCL command permits or prohibits a user from accessing data in a database schema.

The second one is Data Manipulation Language: A data manipulation language (DML) is a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database, and

The third one is called as **Data Definition Language:** Data definition Language is a programming language for creating and modifying data structures.

Let us know in detail about Data control language,

•

•

Let us Know about Data Manipulation Language, The Structured Query Language (SQL) provides a collection of commands for performing such database operations as adding, updating, and removing recordshe primary focus of a data manipulation language is the modification of information already stored in a database. Most SQL statements fall into this category.DML Examples:

- To add new rows to a table, use the insert command.
- The update command is used to modify information already present in a table.
- The Delete command removes data from a table.

Let us know about Data Definition Language, The SQL language provides a collection of commands for defining and updating a database's structure, such as adding, rearranging, and removing tables. they are used to define the schema of the database and also deals with database schema description. DDL is used to create and modify the objects in the database. Let us see the examples of DDL commands

The database, together with its various tables, triggers, etc., can be created using the CREATE statement.
To remove data from a database, use the drop command.
The database's structure can be changed with the ALTER command.
The term "truncate" refers to a database operation that deletes all data from a table.
Data dictionary comments are stored in the "comment" field.
To change the name of a database object, use the rename function.

Different types of SQL queries:

Let us understand different types of SQL Queries. The various types of SQl queries are

- Select Query which is used for selecting certain attribute and give us out the result and selection sometimes based on the condition that is defined.
- secondly Action Query :For the most part, the queries are much like any other SQL query and contained within a Manifold query component. Due to the nature of the SQL commands they employ, they are given the unique label "action" to emphasise their activity in updating the tables or communicating with the user.
- Parameter Query: it is also called as prepared statement , You can pre-compile a SQL statement using a parameterized query and then run it with the "parameters" (think: variables) you provide.

- The last type of SQL is called aggregate query, produces a single result from a multi-value calculation. Many aggregate operations such as average, count, total, minimum, maximum, etc. are available in SQL. Except for the count function, NULL values are not taken into account by an aggregate function.

➢ **<u>Now, let us understand the importance of Select Query.</u>** In database queries, a "select query" is a datasheet view of information. In order to retrieve information from a MySQL database, one must use the select query command. The select query is the most basic type of query and the one most often used in Microsoft Access databases. Its application ranges from selecting and displaying data from a single table to selecting and displaying data from multiple tables.Let us see some example on select query

[illegible]T field1[illegible]2,...fieldN FROM table_name1, table_name2[illegible]WHERE[illegible]

[illegible]ELECT * F[illegible]yee; //This select statement dis[illegible]formatio[illegible] table

➢ **<u>Let us understand the importance of Action Query</u>**. A large number of records can be changed or moved with a single action using a special type of query called an action query. Action queries can be used to delete individual records, add rows to another table, modify data in a set of records, or even create entirely new tables. A query can do anything from answering a simple question to performing computations and combining data from different columns. Further to understand in detail lets discuss by taking one example :

DELETE * FROM employee where employeename='XXXXXXX'; // This statement will delete the employee record whose name is 'XXXXXXX'.

➢ After knowing the definition and its features of Action query let us discuss the different types of Action Queries

1. Append Query – have the set of a query and "adds" (or includes) them to a present table.
2. Delete Query – removes all records in a hidden table from the set results of a query.
3. Make Table Query – as the name suggests, it creates a table dependent on the set consequences of a query.
4. Update Query – takes into account at least one field in provided table to be refreshed.

<u>Action query examples are discussed below</u>

1. Example for DELETE Query: DELETE FROM employee WHERE id = 101; here the query deletes a particular row by checking the condition were emp id should be equal to 101, if that record is found that particular row gets deleted from the employee table.

2. Example For UPDATE QUERY: UPDATE employee SET salary= salary+ 5000 WHERE salary <25000;
, here in the given example the row gets updated for which the salary condition is satisfied that means whose salary values is less than five thousand then their salary will be updated to newer value as such theier existing salary plus five thousand will give their updated salary value.

3. Example for MAKE COPY Query:
Create a new query, use the employee and department tables.
From the Query Type button on the toolbar, select Make Table.
Select required fields from employee and department tables.
Check the results which are returned.
If results are as expected, then click on run button to create a new table.
These above queries can serve as an example for make copy query

- **<u>Parameter Query:</u>**

let us understand the importance of Parameter Query. The results you seek can be obtained using a parameter query in Microsoft Access by using one of the many different query methods available. It facilitates the rapid generation of a query that may be readily adjusted to represent a different search term. You may easily generate a query with its help. Parameter queries can be opened in Access, and when you do so, the programme will prompt you for a search phrase before presenting you with relevant results.

```
        create procedure getSalesperson
@sp varchar(25) as
select SalesPerson, amount from Sales where SalesPerson = @sp;  GO
```
This is an example for parameterized query

- **<u>Let us understand the importance of Aggregate Query</u>**. One way to get useful information about groups and subgroups out of a large dataset is to use an aggregate query. One more name for an aggregate query is a summary query or a totals query. To better understand and apply the data, you can do an aggregate query, which is comparable to asking the database how to "group by" the data. Let us see aggregate queries through an example

The various types of Aggregate functions. So there are different types of aggregate functions such as

1.Sum : which will return the sum of all the specified attribute values and gives us the sum value
2.Avg: it returns the average of the selected column
3.Min: this retrns the minimum value among the selected attribute column values
4.Max: this returns the maximum value among the selected attribute column values
5.First : this returns the first value from the selected attribute column values
6.Last : this returns the last value from the selected attribute column values
7.Groupby: this will group the column value based on the matching criteria.

Role of SQL queries:

Useful SQL commands in visual programing:

SQL commands are instructions. It is utilized to communicate with the database. It is also utilized to check specific tasks, functions, and queries of data. The Execute Scalar method is used to execute a database operation and returns a scalar value. A database query's results can be iterated very rapidly with the help of ExecuteReader(). With ExecuteNonQuery(), you can execute code without involving the database in any way. Returning additional data sets from a SqlDataAdapter that may be used in code or imported into a GridView control.

Examples of SQL commands is disced in detail here:

1. [illegible] Cities [illegible] ID INT, City Name VARCHAR(200), State [illegible]

After executing the first query, a new table was i.e., cities. With the mentioned attributes will be created

2. [illegible] Citie[illegible] ty Name, State) VALUES (1, 'Los Angel[illegible] lifornia[illegible]

: After executing insert query, a row will be inserted into cities table with the mentioned values.

3. [illegible] Cities [illegible] shingt[illegible] VHERE [illegible] ID = 4

In this above Update query modify the state of the record to Washington whose city id is 4.and previous value will be replaced with new value washington

4. [illegible] ROM C[illegible]es W[illegible] ty ID

Here in the given Delete query it deletes the record of a particular city , whose city id is

[illegible] Queries

The various advantages of SQL Queries.

- Due to its useful features and widespread adoption, SQL has become an indispensable tool.
- This language is trusted and effective for interacting with databases. Here are a few of SQL's many benefits:
- Huge data sets can be obtained rapidly and efficiently, thanks to faster query processing.
- Data manipulations such as inserting, removing, and rearranging take essentially no time at all.

- No Coding Experience Necessary: Retrieving data does not necessitate a big amount of coding.
- SQL is a user-friendly language because it employs all the fundamental terms

(SELECT, INSERT INTO, UPDATE, etc.) and has simple syntactical norms.

- They have well defined standards, good portability facilities is there by using SQL statements and also they support multiple data views and they are Simple in operation, even the most intricate questions can be answered in a matter of seconds.

8.7 Filling the Dataset

The Data Adapter can Fill a DataSet with records from a Table. You only need a single line of code to do this:

```
da.Fill(ds, "AddressBook")
```

As soon as you type the name of your Data Adapter (**da** for us), you will get a pop up box of properties and methods. Select Fill from the list, then type a pair of round brackets. In between the round brackets, you need two things: the **Name** of your DataSet (**ds**, in our case), and an identifying name. This identifying name can be anything you like. But it is just used to identify this particular Data Adapter Fill. We could have called it 'Bacon Sandwich', if we wanted:

```
da.Fill(ds, "Bacon Sandwich ")
```

The code above still works. But it's better to stick to something a little more descriptive than "Bacon Sandwich"!

Add the new line after the creation of the Data Adaptor:

```
da = New OleDb.OleDbDataAdapter(sql, con)
da.Fill(ds, "AddressBook")
```

And that's it. The DataSet (**ds**) will now be filled with the records we selected from the table called **tblContact**. There's only one slight problem - nobody can see the data yet! We'll tackle that in the next part.

In the previous section, we saw what Data Adaptors and DataSets were. We created a Data Adapter so that it could fill a DataSet with records from our database. What we want to do now is to display the records on a Form, so that people can see them. So this:

- Add two textboxes to your form
- Change the **Name** properties of your textboxes to **txtFirstName** and **txtSurname**
- Go back to your code window

Add the following two lines:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(0).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(0).Item(2)
```

You can add them after the line that closes the connection to the database. Once the DataSet has been filled, a connection to a database can be closed.

Before the code is explained, run your program and click the button. You should see "John Smith" displayed in your two textboxes.

So let's examine the code that assigns the data from the DataSet to the textboxes. The first line was this:

txtFirstName.Text = ds.Tables("AddressBook").Rows(0).Item(1)

It's rather a long line! But after the equals sign, you type the name of your DataSet (**ds** for us). After a full stop, select **Tables** from the popup list. The **Tables** property needs something in between round brackets. Quite bizarrely, this is NOT the name of your database table! It's that identifier you used with the Data Adapter Fill. We used the identifier "**AddressBook**". If we had used "Bacon Sandwich" then we'd put this:

ds.Tables("Bacon Sandwich")

But we didn't, so our code is:

ds.Tables("AddressBook")

Type a full stop and you'll see another list popping up at you. Select **Rows** from the list. In between round brackets, you need a number. This is a Row number from the DataSet. We want the first row, which is row zero in the DataSet:

ds.Tables("AddressBook").Rows(0)

Type full stop after Rows(0) and the popup list appears again. To identify a **Column** from the DataSet, you use **Item**. In between round brackets, you type which column you want:

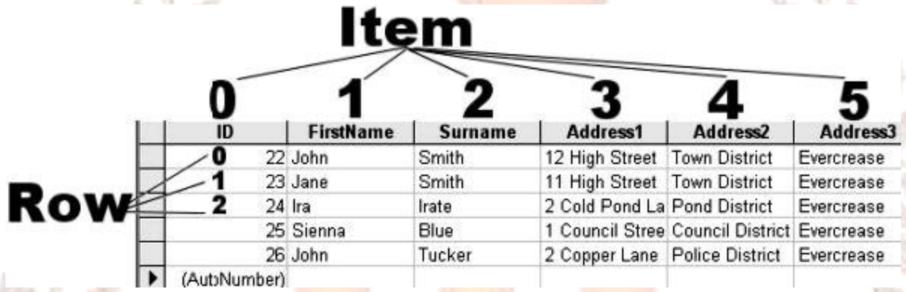
ds.Tables("AddressBook").Rows(0).Item(1)

In our Access database, column zero is used for an ID field. The **FirstName** column is the second column in our Access database. Because the Item collection is zero based, this is item 1 in the DataSet.

You can also refer to the column name itself for the Item property, rather than a number. So you can do this:

```
ds.Tables("AddressBook").Rows(0).Item("FirstName")
ds.Tables("AddressBook").Rows(0).Item("Surname")
```

If you get the name of the column wrong, then VB throws up an error. But an image might clear things up. The figure 8.3 below shows what the items and rows are in the database.



ID	FirstName	Surname	Address1	Address2	Address3
22	John	Smith	12 High Street	Town District	Evercrease
23	Jane	Smith	11 High Street	Town District	Evercrease
24	Ira	Irate	2 Cold Pond La	Pond District	Evercrease
25	Sienna	Blue	1 Council Stree	Council District	Evercrease
26	John	Tucker	2 Copper Lane	Police District	Evercrease
(AutoNumber)					

Fig. 8.3: Database Row & Column

The image shows the **Rows** and the **Items** in the Access database Table. So the **Items** go down and the **Rows** go across.

However, we want to be able to scroll through the table. We want to be able to click a button and see the next record. Or click another button and see the previous record. You can do this by incrementing the Row number. To see the next record, we'd want this:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(1).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(1).Item(2)
```

The record after that would then be:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(2).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(2).Item(2)
```

So by incrementing and decrementing the Row number, you can navigate through the records. Let us see how that's done.

You saw in the previous section that you can navigate through the records of a database by incrementing or decrementing the Row number of the DataSet. In this section, we're going to see a more practical example of how to do that. It's better if you start a new project for this. With a new form open, do the following:

- Add two Textboxes. Change the **Name** properties to **txtFirstName** and **txtSurname**
- Add four Buttons. Change the **Name** and **Text** properties to these:

Button Name	**Button Text**
btnNext	Next Record
btnPrevious	Previous Record
btnFirst	First Record
btnLast	Last Record

After the completion of above process, your form look like as shown in figure 8.4.

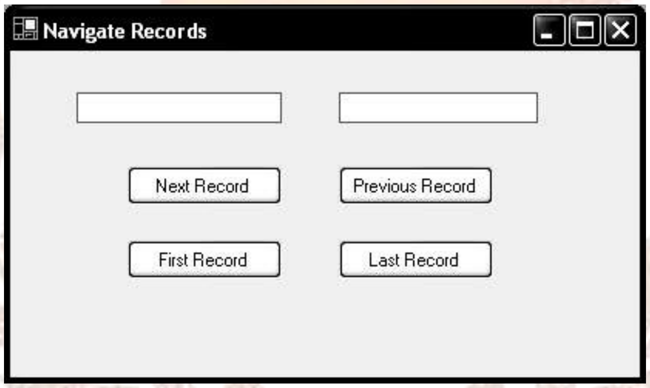


Fig. 8.4: Navigation window

Press F7 to see your code window, and add the following code to the **Form1 Declarations** area:

Declarations area:

```
Public class Form1
Dim inc as Integer
Dim MaxRows as Integer

Dim con As New OleDb.OleDbConnection
Dim ds as New DataSet
Dim da as OleDb.OleDbDataAdapter
Dim sql as String
```

(VB 2005 Express Edition users: don't forget to add the references! Click **Project > Add References**. Locate **System.Data.dll** and **System.Xml.dll** on the **NET** tab. Select these items and click OK. Then add **Imports System.Data** at the very top of your code window.) Your code will look like this:

```
Imports Syste.Data
Public class Form1
    Dim inc As integer
    Dim conn As New OleDb.OleDbConnection
    Dim ds As New DataSet
    Dim da As OleDb.DataAdapter
    Dim sql As String
```

Setting up the variables will be done before the code starts. There's one for the Connection Object, one for the DataSet, and one for the Data Adaptor. We've also set up two Integer variables (**inc** and **MaxRows**), and a String variable (**sql**).

When the Form Loads, we can connect to our database, use the data Adaptor to grab some records from the database, and then put these records into the DataSet. So in the Form1 Load Event, add the following code as shown in figure 8.6

```
Private Sub Form1_Load (ByVal sender As object Byval e As System._
    EventArgs) Handles Me.Load
Con.ConnectionString = “Provider = Microsoft.Jet.OLEDB.4.0;Data source_
    = c:\AddressBook.mdb”
Con.open()
Sql = “SELECT * FROM tablContacts”
Da=New OleDb.OleDb.OleDataAdapter(sql,con)
Da.Fill(ds, “AddressBook”)
Con.close()
MaxRows = ds.Tables(“AddressBook”.Rows.Count
Inc=-1
End Sub
```

You've met all the code before, except for these two lines:

MaxRows = ds.Tables("AddressBook").Rows.Count
inc = -1

In the MaxRows variable, we can store how many rows are in the DataSet. You get how many rows are in yout DataSet with **Rows.Coun**t:
MaxRows = ds.Tables("AddressBook").Rows.Count

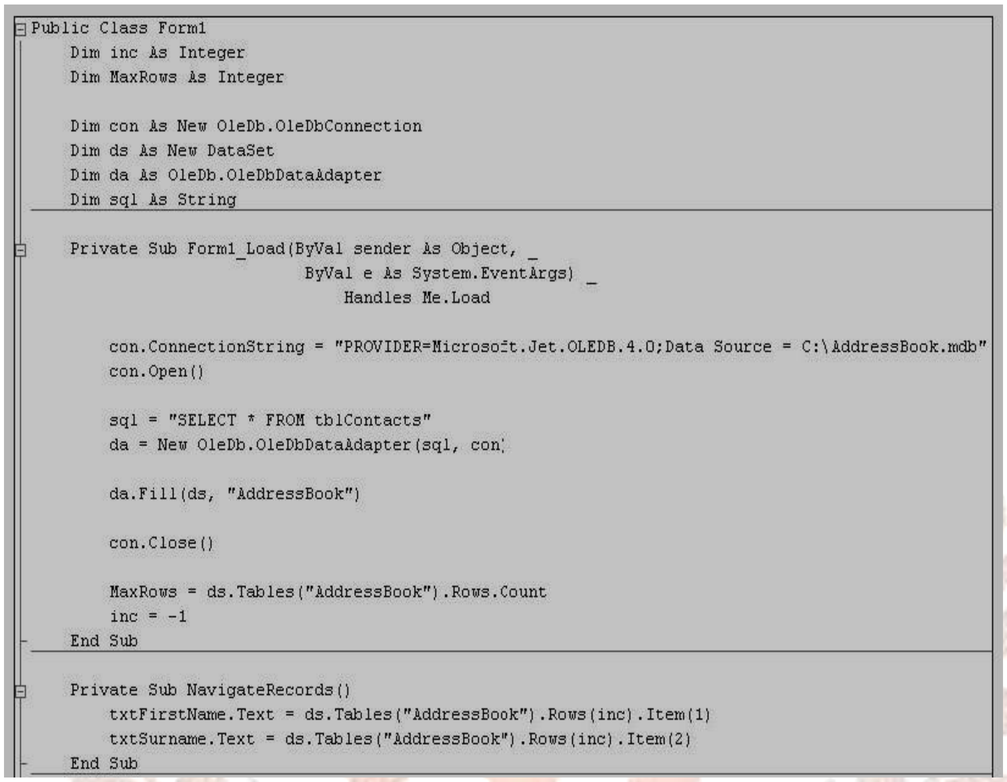
So the Rows property has a Count Method. This simply counts how many rows are in the DataSet. We are passing that number to a variable called **MaxRows**. You can then test what is in the variable, and see if the **inc** counter doesn't go past it. You need to do this because VB throws up an error message if try to go past the last row in the DataSet. (Previous versions of VB had some called an **EOF** and **BOF** properties. These checked the **End of File** and **Before End** of File.

To navigate through the records, we're going to use that **inc** variable. We'll either add 1 to it, or take 1 away. We'll then use the variable for the **Rows** in the DataSet. It's better to do this in a Subroutine of your own. So add this Sub to your code:

```
Private Sub NavigateRecords()
txtFirstName.Text = ds.Tables("AddressBook").Rows(inc).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(inc).Item(2)
End Sub
```

The important part is **Rows(inc)**. This moves us through the **Rows** in the DataSet. We're then placing the values into the two Textboxes.

The whole of your code so far should look as shown in figure 8.5 (Express Edition user will have the **Imports System.Data** line at the very top):



```
Public Class Form1
    Dim inc As Integer
    Dim MaxRows As Integer

    Dim con As New OleDb.OleDbConnection
    Dim ds As New DataSet
    Dim da As OleDb.OleDbDataAdapter
    Dim sql As String

    Private Sub Form1_Load(ByVal sender As Object, _
                        ByVal e As System.EventArgs) _
                            Handles Me.Load

        con.ConnectionString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source = C:\AddressBook.mdb"
        con.Open()

        sql = "SELECT * FROM tblContacts"
        da = New OleDb.OleDbDataAdapter(sql, con)

        da.Fill(ds, "AddressBook")

        con.Close()

        MaxRows = ds.Tables("AddressBook").Rows.Count
        inc = -1
    End Sub

    Private Sub NavigateRecords()
        txtFirstName.Text = ds.Tables("AddressBook").Rows(inc).Item(1)
        txtSurname.Text = ds.Tables("AddressBook").Rows(inc).Item(2)
    End Sub
```

Fig. 8.5: Complete Data Access code

In the next unit, we will discuss to add the code for the buttons.

Self Assessment Questions

6. Name the language through which we can write and read the data in the data base.
7. In SQL statement * indicates__________________.
8. __________________is the term used in VB .NET to indicate column.
9. Name the method that gives the number of records in the particular data set.

8.8 Summary

- This unit starts with the mechanism of data access and the data form wizard in Visual Studio.Net.
- It demonstrates the concepts of connections strings, opening the connection, and so on necessary to connect and access the data to any data source.

- Open() and Close() are the two connection methods to open and close the connection with the database.
- Unit describes the concepts of Structured Query language and how to retrieve the data from data base.
- Explored the concepts of filling the data sets.

8.9 Terminal Questions

1. Describe the Data form Wizard in Visual Studio.Net/
2. Describe the concept of setting a connection string with an example.
3. Discuss the procedure involved in opening a connection.
4. What is Data adapter? Explain its role in database.
5. Explain the concept of filling the dataset.

8.10 Answers

Self Assessment Questions

1. Active Data Objects.
2. Provider and Data Source
3. OLE
4. True
5. Data Adapter.
6. SQL
7. All column in the table.
8. Item.
9. Rows.Count

Terminal Questions

1. Connection, command, DataReader, DataSet and DataAdapter are the various five objects of ADO .NET. For more details refer to section 8.2.
2. Connection string is the important object to connect with the database. For more details refer to section 8.3.
3. Connection. Open() and Connection.Close() are the two important methods involved in opening and closing connection with the database. For more details refer section 8.4.
4. The Data Adapter is a property of the OLEDB object. It creates the connection between connection object and the data set. For more details refer section 8.5.

5. The data adapter will fill the data sets with the records that are retrieved from table. For more details refer section 8.7.

E-Reference:

- http://visualbasic.about.com/od/usingvbnet/a/begdbapp7.htm
- http://msdn.microsoft.com/en-us/library/ms254947.aspx
- http://www.codeproject.com/Articles/9664/SQL-and-VB-NET-Code-Generator
- http://www.homeandlearn.co.uk/net/nets12p5.html