Unit VI: Data Access with ADO.Net, accessing data with Server Explorer, Accessing Data with data Adaptors and Data sets, Creating a new data connection, creating and populating Data set, displaying data in Data Grid, selecting a data provider, Data accessing using Data adapter Control, Binding Data to Controls.

ADO stands for Microsoft ActiveX Data Objects. ADO.NET is one of Microsoft's Data Access Technologies, using which we can communicate with different data sources. It is a part of the .NET Framework, which connects the .NET Application (Console, Windows, Web Form, etc.) and different data sources. The Data Sources can be SQL Server, Oracle, MySQL, Microsoft Access, etc. ADO.NET consists of a set of predefined classes that can be used to connect, retrieve, insert, update, and delete data from data sources.

ADO.Net Object Model

ADO.NET, which stands for ActiveX Data Objects for .NET, is a data access technology provided by Microsoft as part of the .NET Framework. It allows .NET applications to communicate with and manipulate data from various data sources, such as databases and XML files. ADO.NET supports both connected and disconnected data access architectures.

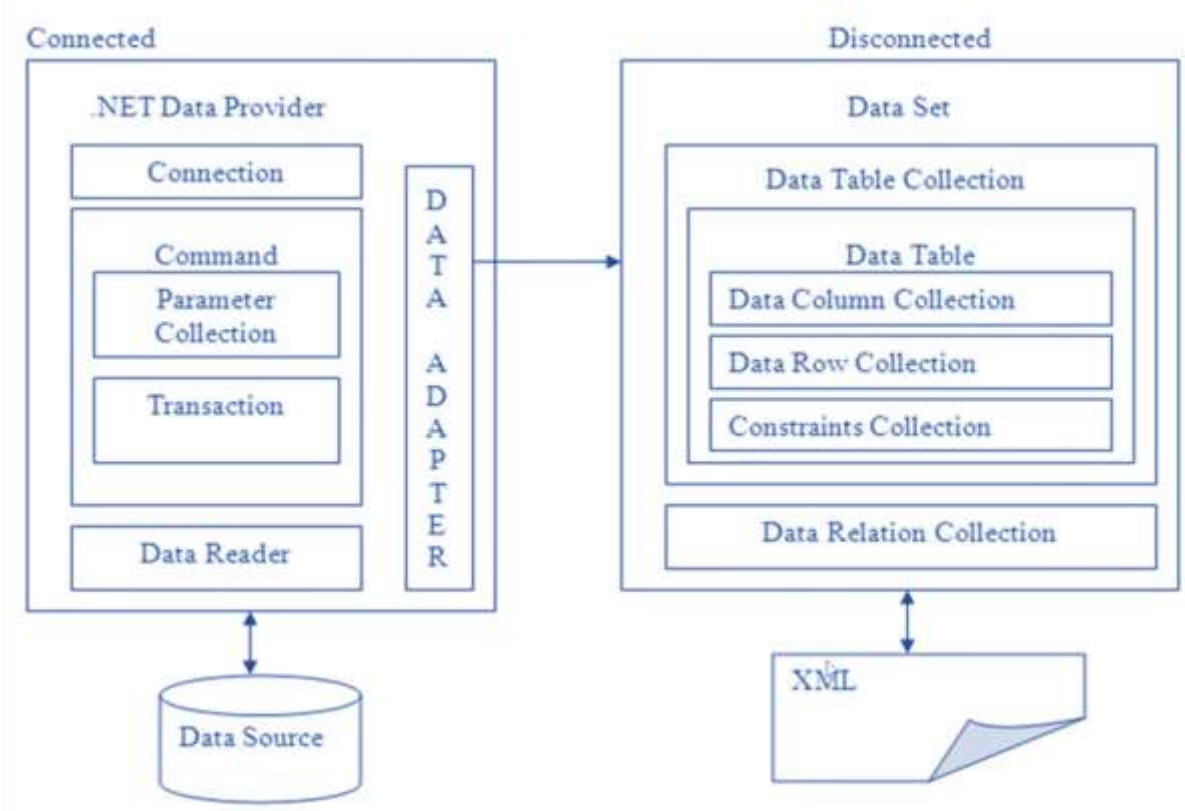


Fig: Connected-Disconnected Architecture(ADO.Net Object Model)

ADO.Net Object model relies on two components

i. .Net Data Provider
ii. DataSets

1. Connected Architecture:

In the connected architecture of ADO.NET, the application maintains a continuous connection to the data source throughout the entire duration of the data access operation. Here's how it works:

Connection Object: A connection object is used to establish a connection to the data source. You create a connection object by specifying the connection string, which contains information such as the data source location, authentication credentials, and other parameters required to establish a connection.

Command Object: Once the connection is established, you create a command object to execute SQL commands against the data source. The command object represents a SQL statement (such as SELECT, INSERT, UPDATE, DELETE) or a stored procedure.

DataReader Object: When you execute a SELECT statement using the command object, it returns a DataReader object, which allows you to sequentially read the results of the query. The DataReader provides a read-only and forward-only stream of data from the data source.

DataAdapter Object (Optional): In connected architecture, DataAdapter is not commonly used. It's more associated with the disconnected architecture. However, it can be used to fill a DataSet in connected scenarios as well.

Transaction Object: ADO.NET also supports transactions in connected scenarios, allowing you to group multiple database operations into a single atomic unit of work.

Once the data access operation is complete, you close the connection to release the resources and free up the connection pool.

2. Disconnected Architecture:

In the disconnected architecture of ADO.NET, the connection to the data source is opened only when data needs to be retrieved or updated. The connection is closed immediately after the operation is complete. Here's how it works:

DataAdapter Object: In disconnected architecture, the DataAdapter object plays a crucial role. It acts as a bridge between the DataSet (or DataTable) and the data source. The DataAdapter fills the DataSet with data from the data source and updates the data source with changes made to the DataSet.

DataSet Objects: A DataSet is an in-memory cache of data retrieved from the data source. The DataSet allows you to work with data in a disconnected environment, without the need to maintain an open connection to the data source.

DataTable collection: It can contain multiple DataTable objects, each representing a table of data. It contains DataColumn Collection, DataRow Collection, Data Constraints Collection.

- **DataColumn Collection:** The DataColumnCollection represents the collection of DataColumn objects for a DataTable. DataColumn objects define the structure of the columns in the DataTable.

- **DataRow Collection:** The DataRowCollection represents the collection of DataRow objects within a DataTable. DataRow objects represent individual rows of data in the DataTable.
- **DataConstraints:** DataConstraints represent rules enforced on the data in a DataTable to maintain its integrity. There are several types of constraints:
    - **UniqueConstraint:** Ensures that values in one or more columns of a DataTable are unique across all rows.
    - **ForeignKeyConstraint:** Enforces referential integrity by requiring that values in specified columns match values in a parent DataTable.
    - **PrimaryKeyConstraint:** Ensures that values in one or more columns of a DataTable are unique and not null.
- **DataRelation collection:** DataRelation objects can be used to define relationships between DataTables within a DataSet.

In disconnected architecture, you fetch data into the DataSet, close the connection, work with the data in the DataSet (including making changes), and then reconnect to the data source to apply those changes (typically using the DataAdapter)

Accessing data with Server Explorer

Accessing data with Server Explorer in Visual Studio allows you to connect to various data sources, explore databases, tables, views, stored procedures, and more. Here are the detailed steps to access data with Server Explorer:

- **Open Visual Studio:** Launch Visual Studio on your computer.
- **Open Server Explorer:** Server Explorer is typically located in the "View" menu of Visual Studio. You can access it by following these steps:
    - Go to the top menu of Visual Studio.
    - Click on "View".
    - Select "Server Explorer" from the dropdown menu.
- **Connect to Data Source:** Once Server Explorer is open, you need to establish a connection to the data source you want to explore. Here's how you can do it:
    - In Server Explorer, right-click on "Data Connections".
    - Select "Add Connection" from the context menu.
- **Choose Data Source Type:** Upon selecting "Add Connection", a dialog box will appear where you can choose the type of data source you want to connect to. The available options might include SQL Server, Oracle, MySQL, and others.
- **Provide Connection Details:** After selecting the data source type, you need to provide connection details specific to your data source. These details typically include:
    - Server name or IP address
    - Authentication method (Windows Authentication or SQL Server Authentication)
    - Username and password (if using SQL Server Authentication)
    - Database name
- **Test Connection:** Once you've entered the connection details, you can test the connection to ensure that the connection parameters are correct. Click on the "Test

Connection" button to verify that Visual Studio can establish a connection to the specified data source.

- **Establish Connection:** After successfully testing the connection, click the "OK" button to establish the connection. Visual Studio will connect to the data source and display it under the "Data Connections" node in Server Explorer.
- **Explore Data Source:** Once the connection is established, you can expand the nodes under the data source to explore its contents. This typically includes databases, tables, views, stored procedures, and other objects depending on the type of data source you've connected to.
- **Perform Operations:** You can perform various operations on the data source such as viewing table data, executing queries, modifying data, creating new tables, and more. Right-click on the objects in Server Explorer to access context menus with available options.
- **Close Connection:** When you're done working with the data source, you can close the connection by right-clicking on the connection in Server Explorer and selecting "Close Connection".

Data Adapter

Data Adapter object act as a bridge database and data source for retrieving and saving data. It is used to fetch data from database and **strictly tied with DataSet class** to create in-memory representation of the data. It is disconnected oriented architecture. It has commands like Select, Insert, Update and delete. **Select command** is used to retrieve data from database and **insert, update and delete commands(ExecuteNonQuery)** are used to send changes to data in dataset to database.

- A Data Adapter represents a set of data commands and a database connection to fill the dataset and update a SQL Server database.
- A Data Adapter contains a set of data commands and a database connection to fill the dataset and update a SQL Server database. Data Adapters form the bridge between a data source and a dataset.
- Data Adapters are designed depending on the specific data source. The following table shows the Data Adapter classes with their data source.

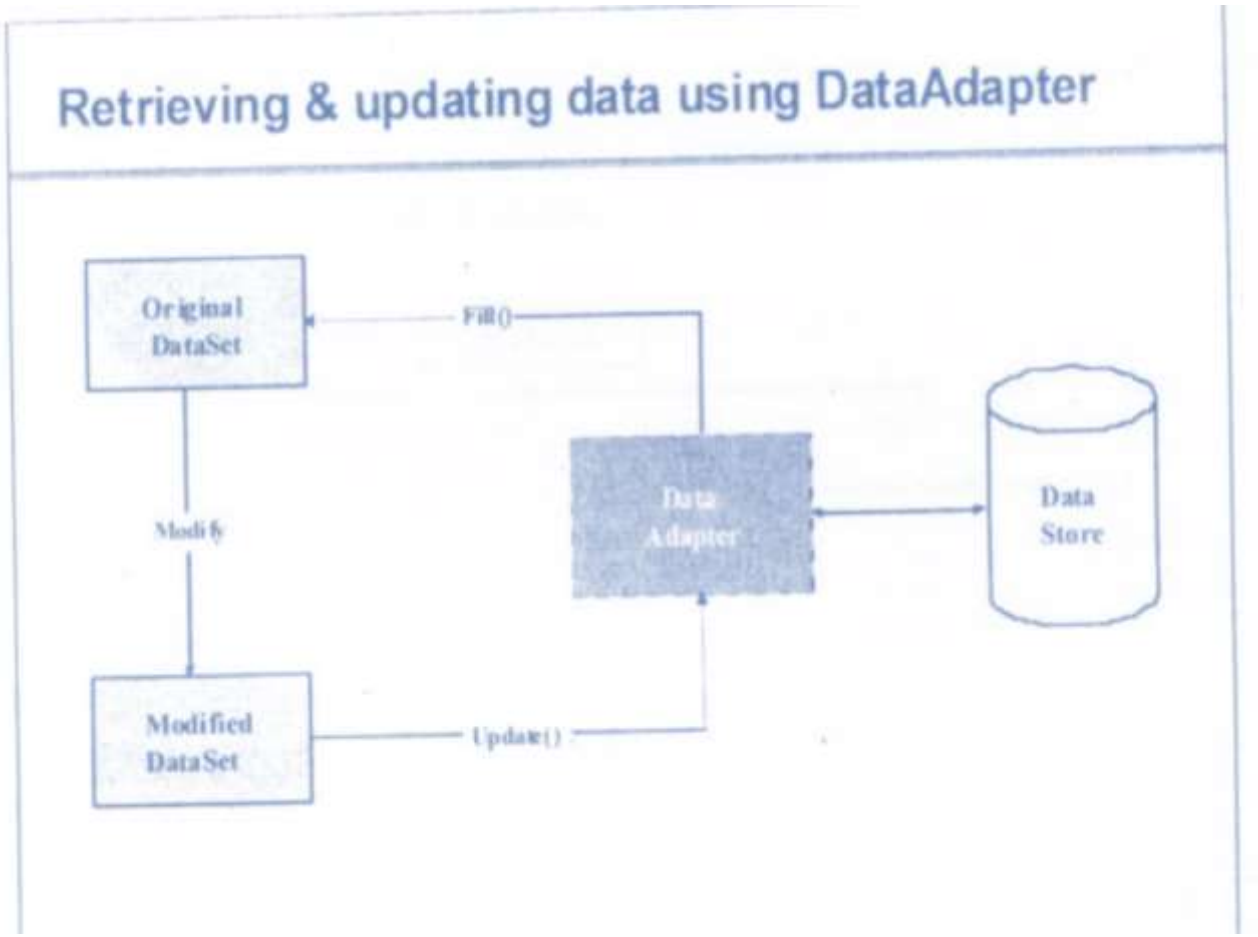
Provider-Specific Data Adapter classes	Data Source
SqlDataAdapter	SQL Server
OledbDataAdapter	OLE DB provider
OdbcDataAdapter	ODBC driver
OracleDataAdapter	Oracle

A Data Adapter supports mainly the following two methods:

- **Fill ()**
  The Fill method populates a dataset or a data table object with data from the database. It retrieves rows from the data source using the SELECT statement specified by an associated select command property. The Fill method leaves the

connection in the same state as it encountered it before populating the data. If subsequent calls to the method for refreshing the data are required then the primary key information should be present.

- **Update ()**
  The Update method commits the changes back to the database. It also analyzes the RowState of each record in the DataSet and calls the appriopriate INSERT, UPDATE, and DELETE statements. A Data Adapter object is formed between a disconnected ADO.NET object and a data source.



[Fig: Retrieving and updating data using DataAdapter]

Example:

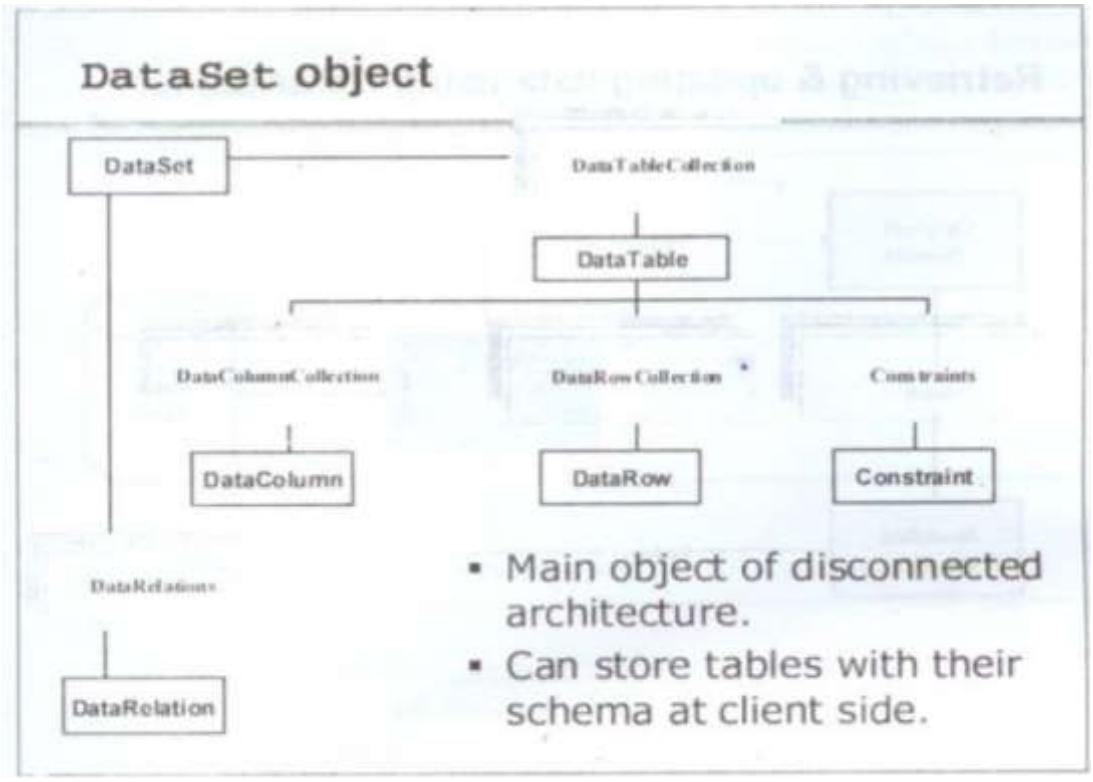
```
Dim da as OledbDataAdapter
Dim ds as New DataSet
da=new OledbDataAdapter("Select * from Stud",conn)
da.fill(ds)
```

DataSet Object

- In the disconnected scenario, the data retrieved from the database is stored in a local buffer called DataSet. It is explicitly designed to access data from any data source. This class is defined in the System.Data namespace.
- A Data Set object is an in-memory representation of the data. It is specially designed to manage data in memory and to support disconnected operations on data.
- A Data Set is a collection of DataTable and DataRelations. Each DataTable is a collection of DataColumn, DataRows, and Constraints.
- A DataTable, DataColumn, and DataRows could be created as follows.

Example

```
1. DataTable dt = new DataTable();
2. DataColumn col =new DataColumn();
3. Dt.columns.Add(col2);
4. DataRow row = dt.newRow();
```



[Fig: DataSet]

Creating a new data connection

Creating a new data connection in an ADO.NET Windows application involves establishing a connection to a database using the appropriate ADO.NET provider. Below are the steps to create a new data connection in a Windows application:

1. **Add Namespace:** Ensure that you have added the necessary namespace at the top of your code file to access ADO.NET classes.

using System.Data.SqlClient; // For SQL Server connections

Depending on your database type, you may use different namespaces such as ***System.Data.OracleClient*** for Oracle databases or ***System.Data.OleDb*** for OLE DB data sources.

2. **Instantiate Connection Object:** Create an instance of the connection object for the chosen data provider.

   SqlConnection connection = new SqlConnection();

   Replace SqlConnection with the appropriate connection class for your chosen data provider.

3. **Set Connection String:** Set the connection string property of the connection object with the necessary details to connect to the database.

   string connectionString = "Data Source=YourServer;Initial Catalog=YourDatabase;User ID=YourUsername;Password=YourPassword;";
   connection.ConnectionString = connectionString;

   Modify the connection string with appropriate values for your server, database, username, and password.

4. **Open Connection (Optional):** Open the connection to the database using the Open() method of the connection object.

   connection.Open();

   This step is optional as the connection will automatically open when you execute a command if it's not already open.
5. **Perform Database Operations:** Once the connection is established, you can execute database operations such as querying data, inserting, updating, or deleting records.

6. **Close Connection:** After performing database operations, it's essential to close the connection to release resources and free up database connections.

   connection.Close();

   Closing the connection is crucial to prevent resource leaks and ensure efficient use of database connections.

Selecting Data Provider

Rather than providing a single set of objects to communicate to a variety of data stores, ADO.NET makes use of multiple data providers. Simply put, a data provider is a set of types (within some .NET assembly) that understand how to communicate with a specific data source. Data provider are provider which helps us to connect with different databases such as SQL, Oracle, OLEDB, ODBC etc.

Types of Data Provider

1. **SQL Server Data Provider**
   SQL Server Data Provider is use to connect with SQL server namespace to connect SQL Server Data Provider is: *System.Data.SQLClient*.

2. **OLEDB Data Provider**
   OLEDB stands for Object Linking and Embedding Database. With the help of OLEDB we can connect both non-relational as well relational database. Used for accessing data from various sources like Microsoft Access, Excel, SQL Server, Oracle, etc. OLEDB namespace to connect OLEDB Data Provider is: *System.Data.OLEDB*.

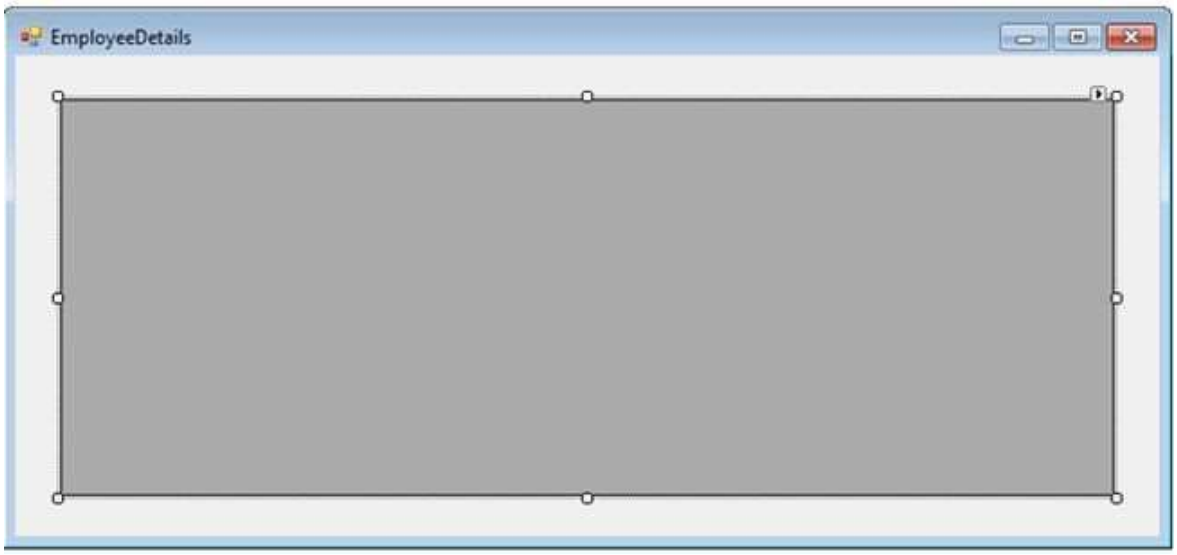
3. **ODBC Provider:**
   Open Database Connectivity (ODBC) is a standard API for accessing database management systems. It allows you to connect to a wide range of databases, but it's generally slower compared to other providers. ODBC namespace to connect ODBC Data Provider is: *System.Data.ODBC.*

4. **Oracle Data Provider**
   Oracle Data Provider is use to connect with Oracle. Namespace to connect Oracle Provider is: *System.Data.OracleClient*.

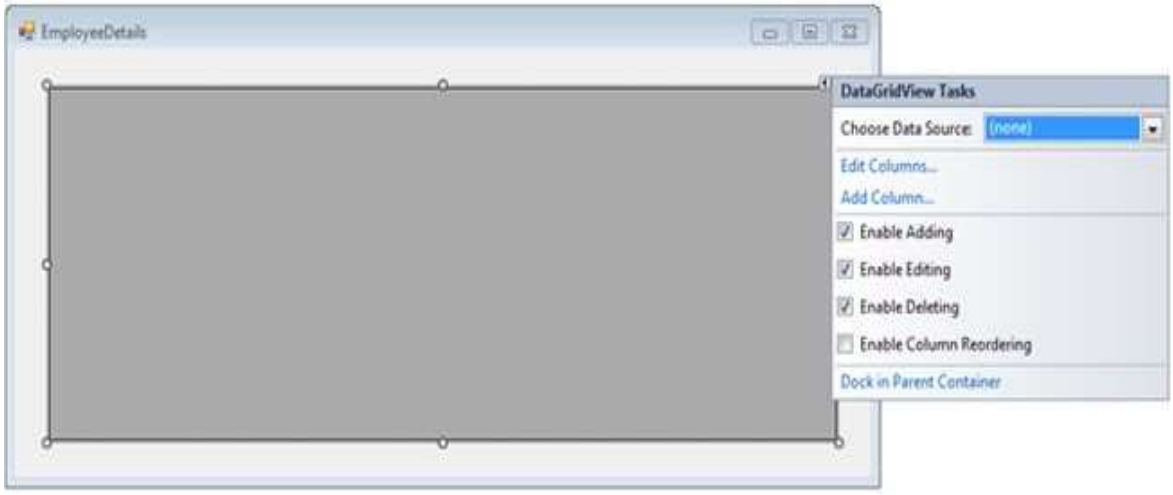
DataGrid View

This is a control in Visual Basic .NET, which provides you with an interactive user-interface to show information graphically. The DataGridView control is basically made of rows and columns. Additionally, every column has its header, where the header text can be changed. The header text can be changed in the designer. It can also be changed programmatically. Vertical and horizontal scrollbars appear automatically whenever they are needed. The scrollbars are used to scroll through the pages to see more content. So, a DataGridView control may contain information of multiple pages.

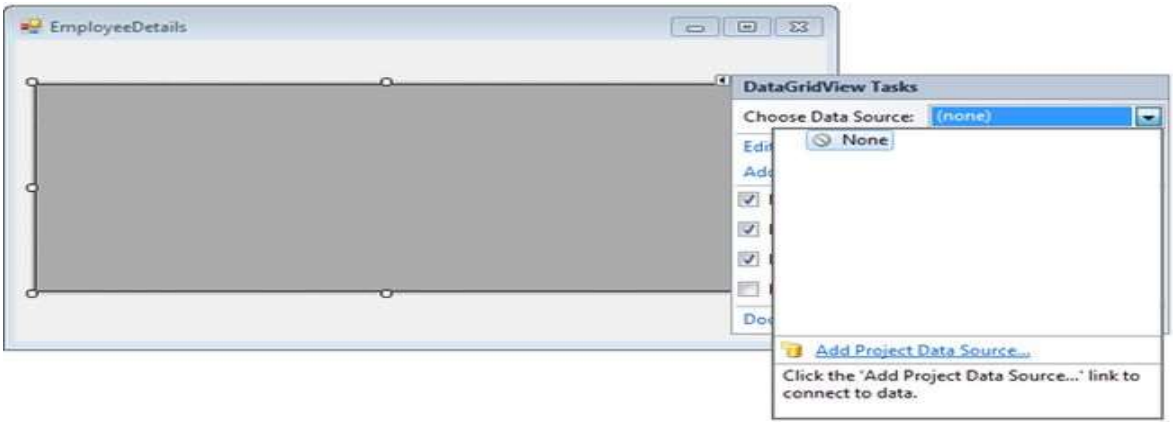
1. Drag and drop a DataGridView from the Toolbox under the Data tab



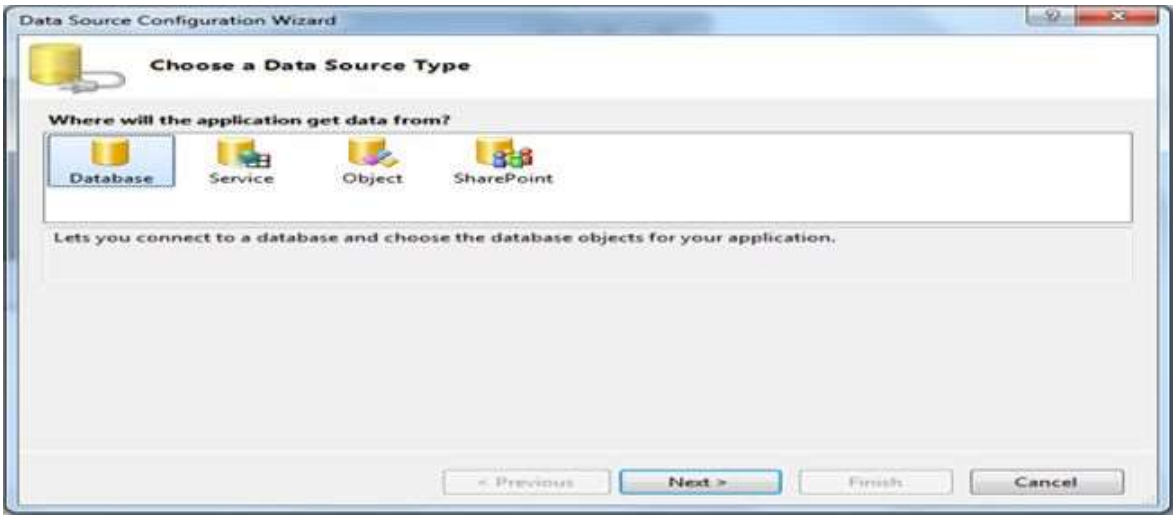
2. Click on the DataGridView task pop-up menu as shown in the following figure.



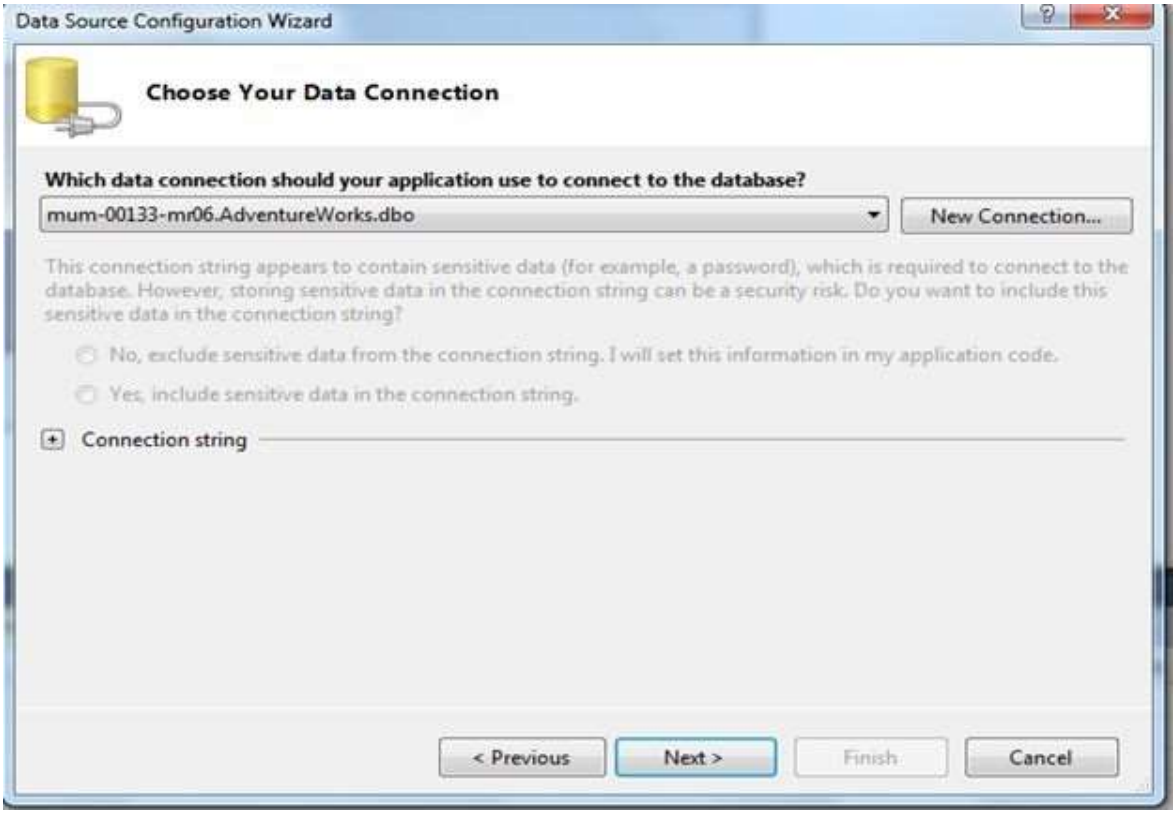
3. Select the choose Data Source drop-down list and then select the Add Project Data Source from the DataGridView task pop-up menu as shown in the following figure.



4. In the database configuration wizard select database and click on "Next" as shown in the following figure.

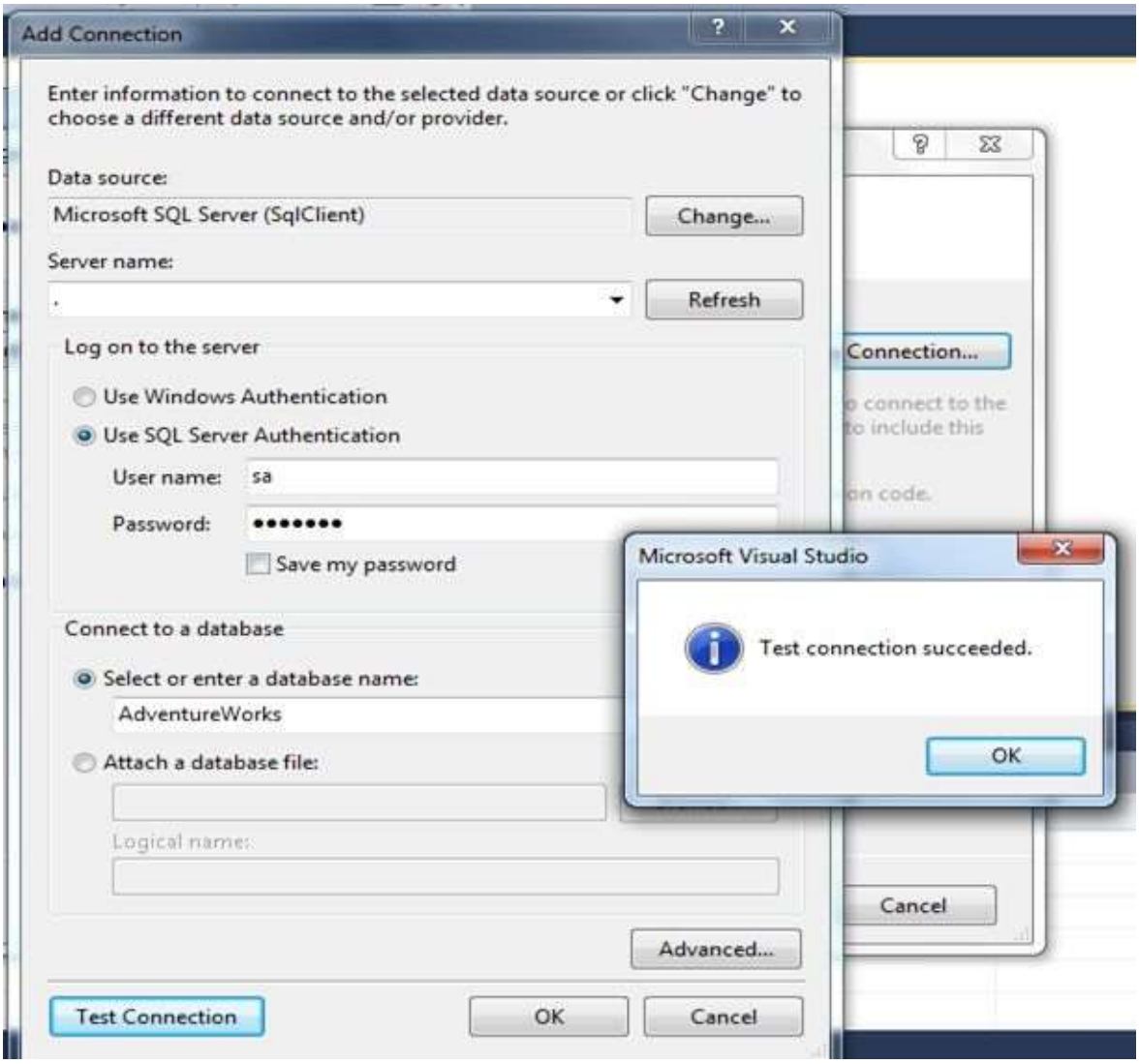


5. After clicking on the Next button you will get the following output.



6. Click on "New Connection" to create a new connection to your data source. To add a connection:

   - Provide the server name (in my case it is (.))
   - If your server is not using Windows authentication then select "Use SQL Server authentication".
   - Provide the username and password.
   - Provide the database name AdventureWorks from the Select or enter a database name drop-down list.
   - Click on the Test Connection Button. If everything goes well you will see a message box saying that Test the connection succeeded as shown in the following figure.

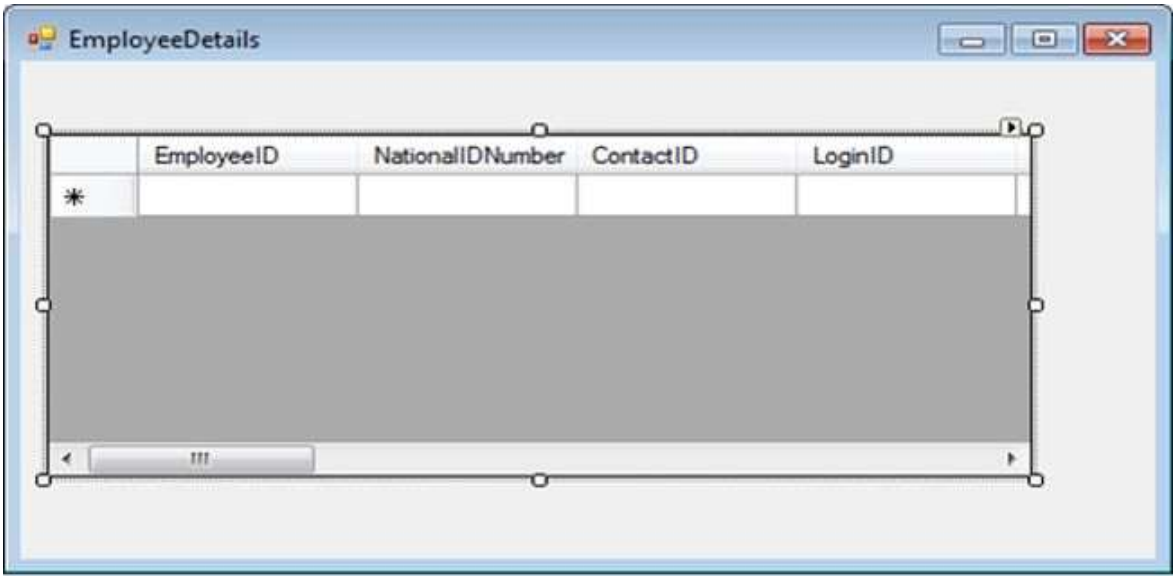


7. Click the OK button.
8. Click the OK button on the Add Connection dialog box.
9. Select Yes, Include sensitive data in the connection string, and click the "Next" button in the DataSource configuration wizard. You will get a page as displayed in the following figure.

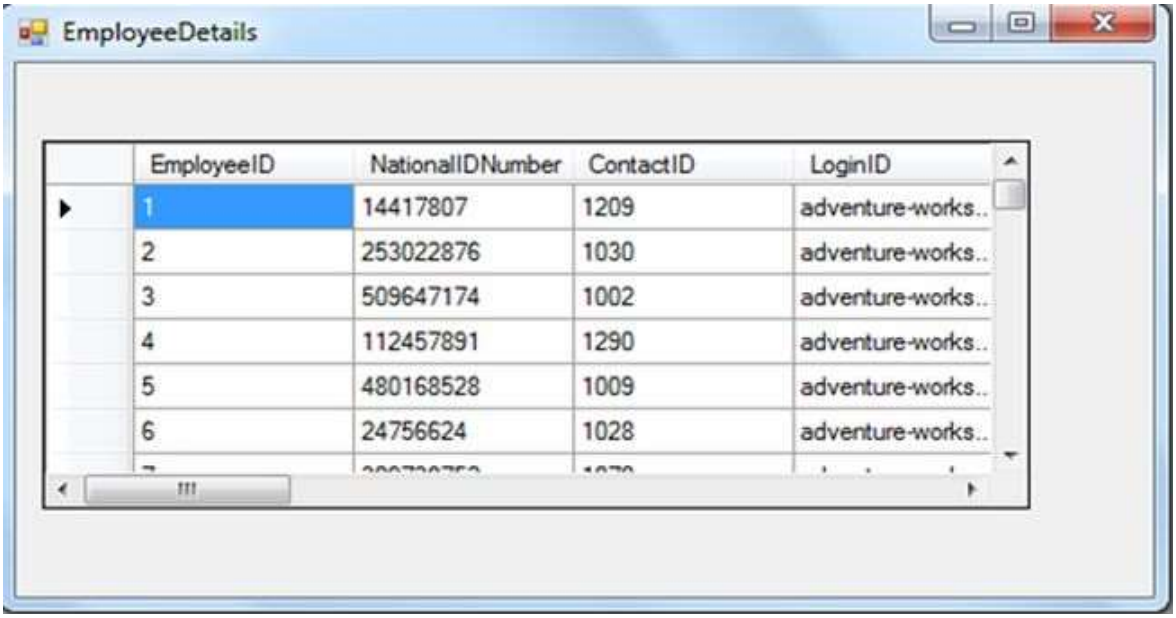
10. Ensure that the Yes, save the connection as the check box is selected and the AdventureWorksConnection string appears in the TextBox.

11. Click the "Next" button. The Choose Your Database Objects page is displayed as shown in the following figure.

12. Click the "Finish" button. The form is displayed, as shown in the following figure.



13. Press F5 to execute the application. You will get the following output.



Data Reader Object

A DataReader object is used to obtain the results of a SELECT statement from a command object. For performance reasons, the data returned from a data reader is a forward-only stream of data. This means that the data can be accessed from the stream in a sequential manner. This is good for speed, but if data needs to be manipulated then a dataset is a better object to work with.

Example

```
1. dr = cmd.ExecuteReader();
2. DataTable dt = new DataTable();
3. dt.Load(dr);
```

It is used in Connected architecture.

- Provide better performance.
- DataReader Object has Read-only access.
- DataReader Object Supports a single table based on a single SQL query of one database.
- While DataReader Object is Bind to a single control.
- DataReader Object has Faster access to data.
- DataReader Object Must be manually coded.
- we can't create a relation in the data reader.
- whereas Data reader doesn't support.
- The data reader communicates with the command object.
- DataReader cannot modify data.

Data Binding

The user can bind values to the respective controls in ADO.NET. Depending on the type of binding offered, they are distinguished as follows:

1. Simple Data Binding
2. Complex Data Binding

Simple Data Binding

Simple data binding allows you to bind a control to a single data element. The most common use of simple data binding involves binding a single data element, such as the value of a column in a table, to a control on a form. You use this type of data binding for controls that show only one value. Uses of simple data binding include binding data to text boxes and labels. Consider a scenario where a Windows Forms form needs to be created to display employee details in the following way.

Example:

Code:

```
Imports System.Data.OleDb
public Class Form1
Dim con As OleDbConnection = New OleDb.OleDbConnection
Dim cm As OleDb.OleDbCommand
Dim dr As  OleDb.OleDbDataReader
private Sub Form1_Load(sender As Object, e As EventArgs) Handles Form1_Load
con.ConnectionString = " Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Users\RuchikaRathi\Desktop\dbsalary.mdb "
con.Open()
 cm  = New OleDb.OleDbCommand("Select * from employee", con)
dr =cm.ExecuteReader()
MsgBox("Connection Success")
dr.Read()
disp()
End Sub
Sub disp()
        TextBox1.Text=dr(0)
        TextBox2.Text=dr(1)
        TextBox3.Text=dr(2)
End Sub
private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1_Click
If(dr.Read()) Then
disp()
MsgBox("Last Record and Closing Reader Object")
dr.Close()
Me.Close()
End If
End Sub
```

End Class

OUTPUT :

Complex Data Binding

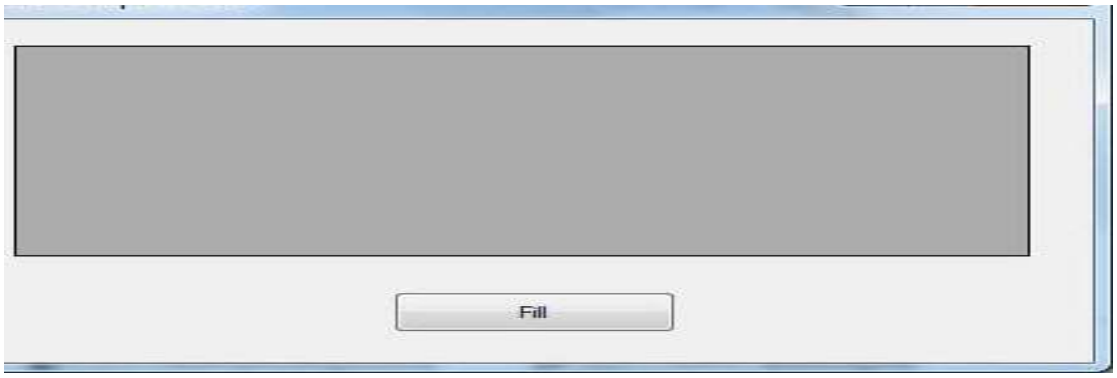
Complex data binding allows you to bind more than one data element to control. Using the column example, complex data binding involves binding more than one column or row from the underlying record source. Controls that support complex data binding include data grid controls, combo boxes, and list boxes. Consider a scenario where a Windows Forms form needs to be created to display records in the following way.

```
Imports System.Data.OleDb
Public Class Form1
    Dim cs As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
                       Source=C:\Users\VVA\Desktop\dbsalary.mdb"
     Dim dt As New DataTable
     Dim ds As New DataSet
     Dim da As New OleDbDataAdapter
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles MyBase.Load

         Dim q As String = "Select * from table1"
         da = New OleDbDataAdapter(q, cs)
```

```
        ds.Tables.Add(dt)
        da.Fill(dt)
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal
    e As System.EventArgs) Handles Button1.Click
        DataGridView1.DataSource = dt.DefaultView
    End Sub
End Class
```

Design Time:



Run Time:

After comparing both coding of Simple binding and complex binding we can observe that to display record we use the **Data reader** and **Command** object(To display single record of employee at time). While in complex binding we use to display record is **Data Adapter** and **Dataset object**(to display multiple record at a time)**.**