ADO .NET
Activex Data Objects

Introduction to ADO.NET

- **ADO.NET** is a data access technology from the Microsoft .NET Framework that provides communication between relational and non-relational systems through a common set of components.
- ADO.NET is a set of computer software components that programmers can use to access data and data services from a database.
- It is a part of the base class library that is included with the Microsoft .NET Framework. It is commonly used by programmers to access and modify data stored in relational database systems, though it can also access data in non-relational data sources.

Introduction to ADO.NET

- ADO stands for Microsoft **Activex Data Objects**.
- **ADO.NET is set of classes that can be used to interact with Data Sources.**
- Commonly, the data source is a database, but it could also be a text file, an Excel spreadsheet, or an XML file.
- Different types of .NET applications like windows application, asp.net web application and console application can use ADO.NET to connect to a database and retrieve data.
- ADO.NET provides consistent access to data sources such as SQL Server, Oracle, XML, and to data sources exposed through OLEDB and ODBC.

.NET Data Providers

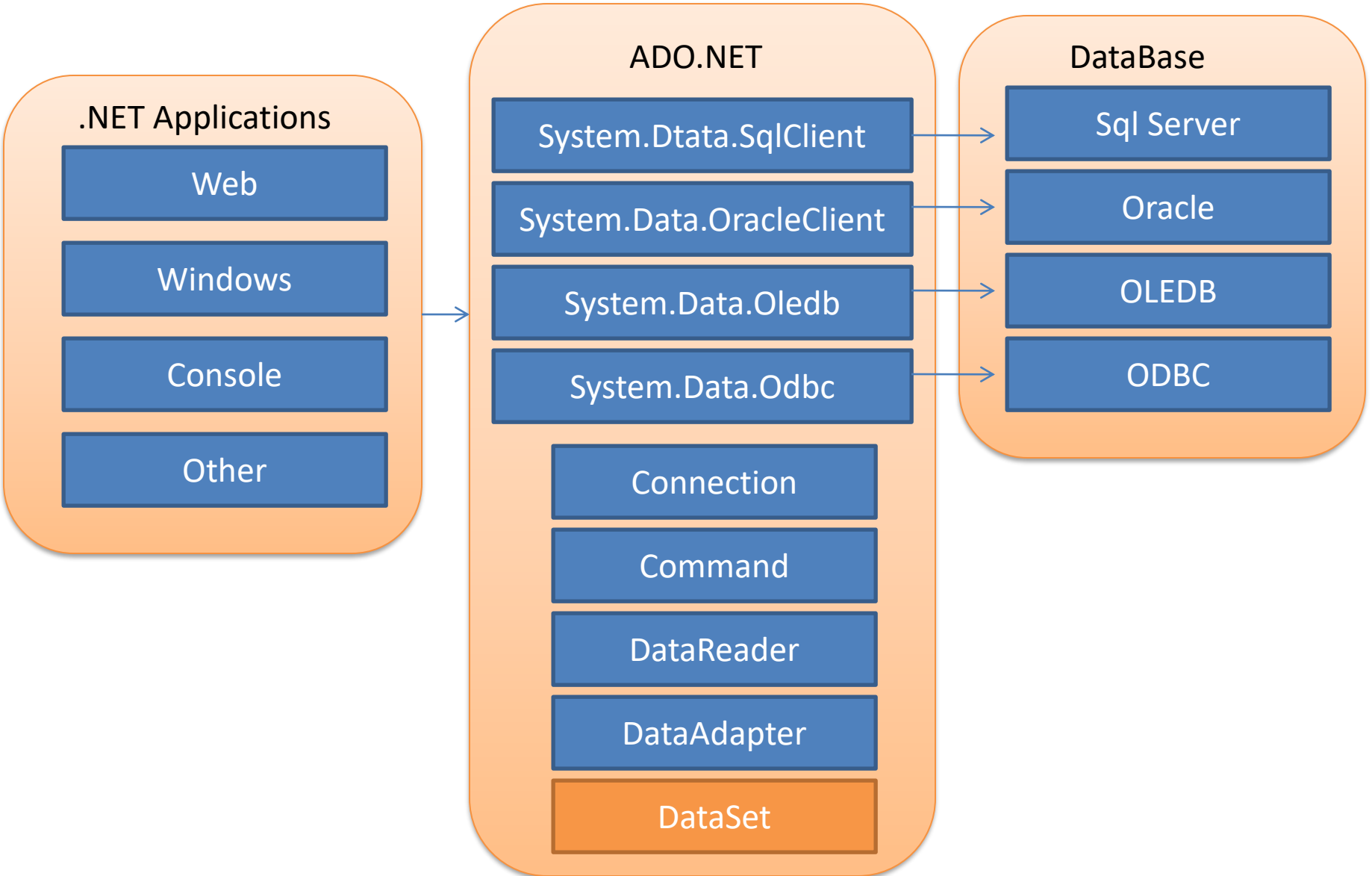
- Data provider means set of classes which can interact with specific data source and provide data to .NET application.
- A .NET Framework data provider is used for connecting to a database, executing commands, and processing results.
- .NET Framework data providers include:

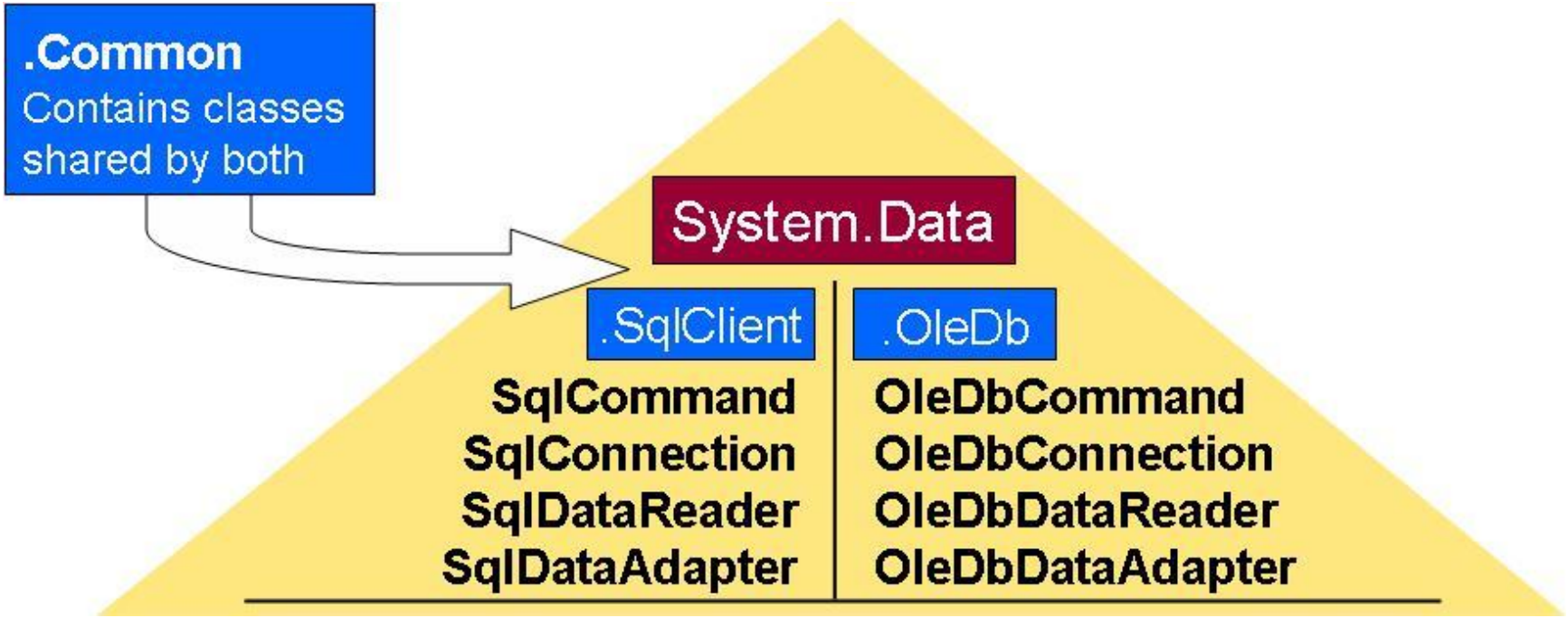
1. Data Provider for SQL Server
2. Data Provider for OLE DB
3. Data Provider for ODBC
4. Data Provider for Oracle

.NET Data Providers

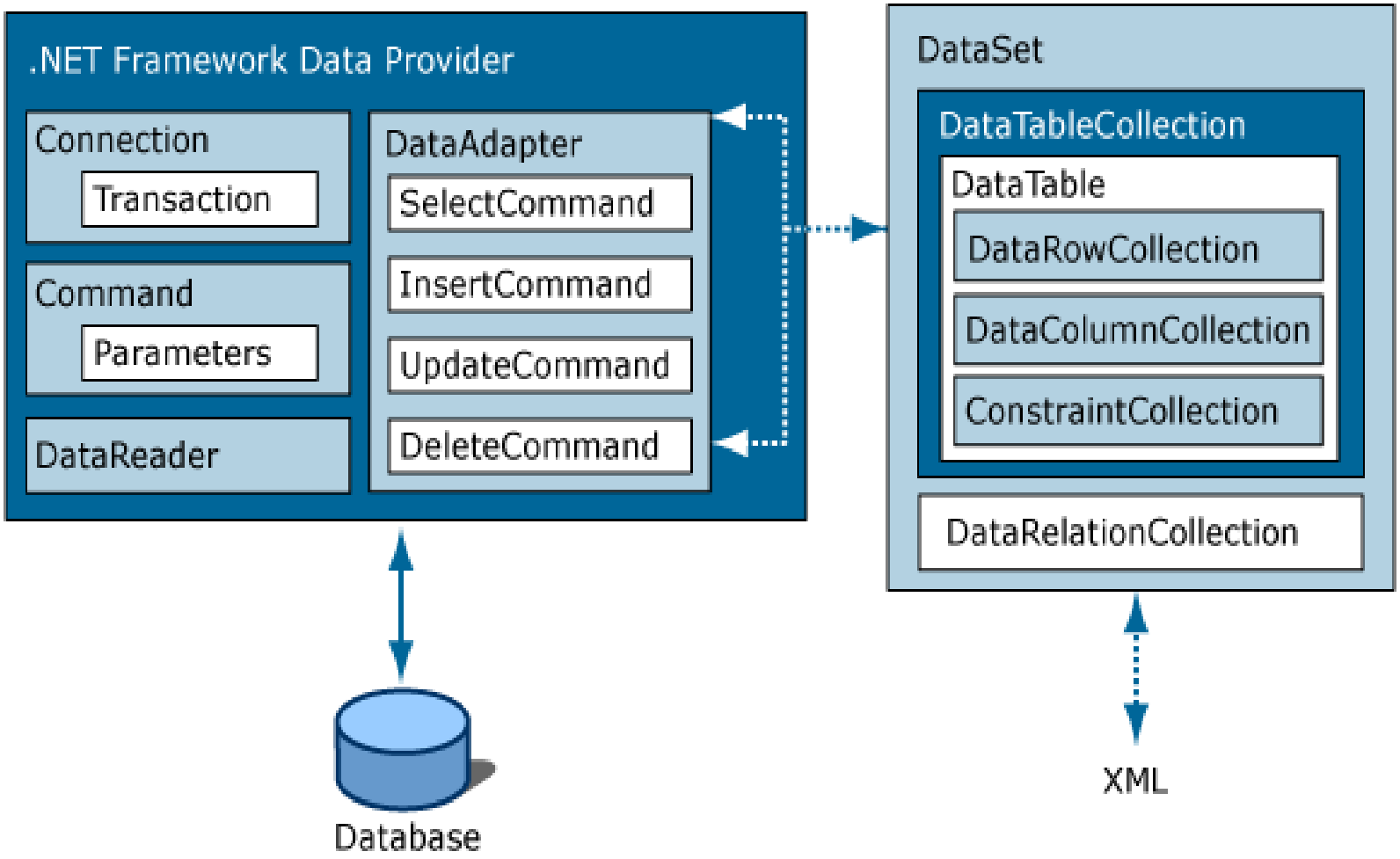
Name Space - System.Data

Data Provider for	NameSpace	Description
SQL Server	System.Data.SqlClient	For interacting with Microsoft SQL Server
Oracle	System.Data.OracleClient	For Oracle Databases.
OLEDB	System.Data.Oledb	DataSources that expose an Oledb interface like Microsoft Access databases or Microsoft Excel spreadsheets
ODBC	System.Data.Odbc	to access an ODBC data source, It includes SQL and also access data other than SQL.


.NET Applications
Web
Windows
Console
Other
ADO.NET
System.Dtata.SqlClient
System.Data.OracleClient
System.Data.Oledb
System.Data.Odbc
Connection
Command
DataReader
DataAdapter
DataSet
DataBase
Sql Server
Oracle
OLEDB
ODBC


.Common
Contains classes shared by both
System.Data
.SqlClient
.OleDb
SqlCommand
SqlConnection
SqlDataReader
SqlDataAdapter
OleDbCommand
OleDbConnection
OleDbDataReader
OleDbDataAdapter

ADO.NET architecture



ADO.NET Architecture

- **Overview of database access**
- Three steps:
  - open connection to database
  - execute SQL to retrieve records/ insert / update DB
  - close connection
- **Two main components of ADO.NET**

(1) .NET Framework Data Providers: For data manipulation and fast, forward-only, read-only access to data.

(2) DataSet

Connection Object

- Establishes a connection to a specific data source.
- The connection tells the rest of the ADO.NET code which database it is talking to it .
- Require Connection String :

```
string conStr= "Data Source=yourSQLServer; Initial Catalog = yourDB;
User Id=yourUserName; Password=yourPwd";

SqlConnection cn = new SqlConnection(conStr);
```

- The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base.
- A connection object is used by command objects so they will know which database to execute the command on.

Connection Object

Parameter Name	Description
Data Source	Identifies the server. The name or network address of the instance of Server to which to connect.
Initial Catalog or Database	Database name.
Integrated Security	When false, User ID and Password are specified in the connection. When true, the current Windows account credentials are used for authentication.
User ID	Name of user configured in SQL Server.
Password	Password matching SQL Server User ID.
AttachDBFilename	The name of the primary database file, including the full path name of an attachable database. AttachDBFilename is only supported for primary data files with an .mdf extension.
Connect Timeout	length of time (in seconds) to wait for a connection to the server before terminating the attempt and generating an error.

Connection Object

Connection string:

string cn=@"Data Source = (LocalDB)\v11.0; AttachDbFilename = C:\Users\Nir\Documents\Student.mdf;Integrated Security=True;Connect Timeout=30";

Command Object :

- Executes a command(Query) against a data source

string sql = @"Insert into Student values (name,address)";
SqlCommand cd = new SqlCommand(sql, cn);

- Use a command object to send SQL statements to the database. A command object uses a connection object to figure out which database to communicate with.
- SqlCommand object allows you to specify what type of interaction you want to perform with a database. For example, you can do select, insert, modify, and delete commands on rows of data in a database table.

Methods of Command Object

- Depending on the command type and return value, three, methods are provided for a command object

Method	Return Value	Query Type
ExecuteReader()	Returns a DataReader object.	Used for Select Query
ExecuteScalar()	Returns a single scalar value.	Used for Select Query for aggregate values
ExecuteNonQuery()	Executes a command that does not return any rows ,only return no of rows affected by query.	Used for insert ,update and delete query.

Before Applyig SqlCommand's method open the connection.

cn.open();

SqlCommand Methods

1) ExecuteNonQuery (Insert/Update/Delete)

→ **To insert data into a database,**

// prepare command string

string insertString = @“ insert into Categories (CategoryName, Description) values ('Miscellaneous', 'Whatever doesn''t fit elsewhere')";

// 1. Instantiate a new command with a query and connection

SqlCommand cmd = new SqlCommand(insertString, conn);

// 2. Call ExecuteNonQuery to send command

cmd.ExecuteNonQuery();

SqlCommand Methods

1) <u>ExecuteNonQuery (Insert/Update/Delete)</u>

→ <u>To update the a record into a database,</u>

// prepare command string

string updateString = @”update Categories set CategoryName = 'Other‘ where CategoryName = 'Miscellaneous'";

SqlCommand Methods

1) ExecuteNonQuery (Insert/Update/Delete)

→ **To Delete the a record into a database,**

// prepare command string

string deleteString = @"delete from Categories where CategoryName = 'Other'";

SqlCommand Methods

2) <u>ExecuteReader - Querying Data</u>

- Used to retrieve data records for viewing.
- It returns a SqlDataReader object.

- Example:

  // 1. Instantiate a new command with a query and connection

  SqlCommand cmd = new SqlCommand("select CategoryName from Categories", cn);

  // 2. Call Execute reader to get query results

  SqlDataReader rdr = cmd.ExecuteReader();

SqlCommand Methods

3) ExecuteScalar - Getting Single value

→ To **get count, sum, average, or other aggregated value from a database**.

// 1. Instantiate a new command
SqlCommand cmd = new SqlCommand("select count(*) from Categories", cn);

// 2. Call *ExecuteScalar* to send command
int count = (int)cmd.ExecuteScalar();

The query in the SqlCommand constructor obtains the count of all records from the Categories table. This query will only return a single value.

DataReader, DataSet, DataAdapter, and DataTable are four major components of ADO.NET.

DataReader

- DataReader is used to read the data from the database and it is a read and forward only connection oriented architecture during fetch the data from database.
- DataReader will fetch the data very fast when compared with dataset.
- Generally, we will use ExecuteReader object to bind data to datareader.

Program

//To bind DataReader data to GridView:

```
protected void BindGridview()
{
   SqlConnection conn = new SqlConnection("Data
Source=abc;Integrated Security=true;Initial Catalog=Test"))
{
con.Open();
      SqlCommand cmd = new SqlCommand("Select UserName,
First Name,LastName,Location FROM Users", conn);
      SqlDataReader sdr = cmd.ExecuteReader();
      gvUserInfo.DataSource = sdr;
      gvUserInfo.DataBind();
      conn.Close();
   }}
```

DataReader

- Holds the connection open until you are finished (don't forget to close it!).
- Can typically only be iterated over once
- Is not as useful for updating back to the database

DataSet

- DataSet is a disconnected orient architecture that means there is no need of active connections during work with datasets.
- It is a collection of DataTables and relations between tables.
- It is used to hold multiple tables with data.
- You can select data form tables, create views based on table and ask child rows over relations.
- DataSet provides you with rich features like saving data as XML and loading XML data.

Program

```
protected void BindGridview()
{
    SqlConnection conn = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test");
    conn.Open();
    SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
    SqlDataAdapter sda = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    sda.Fill(ds);
    gvUserInfo.DataSource = ds;
    gvUserInfo.DataBind();
}
```

DataAdapter

- DataAdapter will acts as a Bridge between DataSet and database. This dataadapter object is used to read the data from database and bind that data to dataset.
- Dataadapter is a disconnected oriented architecture.

Program

```
protected void BindGridview()
{
    SqlConnection con = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test");
    conn.Open();
    SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
    SqlDataAdapter sda = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    sda.Fill(ds);
    gvUserInfo.DataSource = ds;
    gvUserInfo.DataBind();
}
```

DataAdapter

- Lets you close the connection as soon it's done loading data, and may even close it for you automatically
- All of the results are available in memory
- You can iterate over it as many times as you need, or even look up a specific record by index
- Has some built-in faculties for updating back to the database.

DataTable

- DataTable represents a single table in the database. It has rows and columns. There is no much difference between dataset and datatable, dataset is simply the collection of datatables.

Program

```
protected void BindGridview()
{
    SqlConnection con = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test");
    conn.Open();
    SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
    SqlDataAdapter sda = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    gridview1.DataSource = dt;
    gvidview1.DataBind();
}
```

Any Query????