Asp.Net

ADO.NET and Database Connectivity in ASP.NET

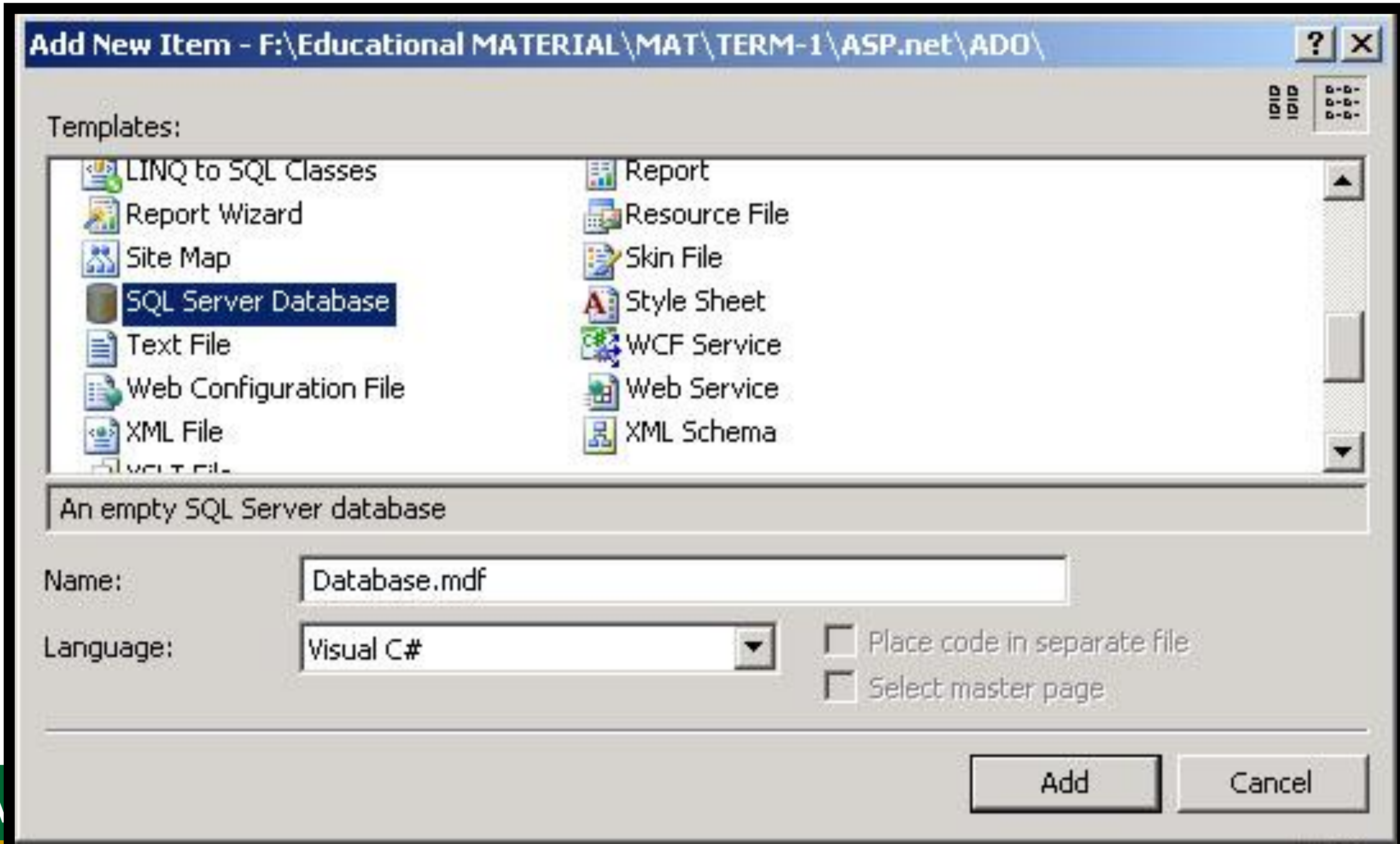
NAME :

- Try to use name with specified rules…

1. No Space
2. No Special Symbols
3. Must starts with ALPHABETS.
4. Name have such meaning.
5. Field name have no space.
6. Field name must follow all the rules of variable naming.
7. Field name must be unique.

How To Add SQL Database in ASP ?

- To add new SQL Database
  - Goto WebSite menu…
    - Select [Add New Item…]



Add New Item - F:\Educational MATERIAL\MAT\TERM-1\ASP.net\ADO\

Templates:

An empty SQL Server database

Name: Database.mdf

Language: Visual C#

□ Place code in separate file

□ Select master page

SQL : Server Explorer

- After creating a new database we can see server explorer as given.

  - We can add new table by RIGHT Click on Tables.

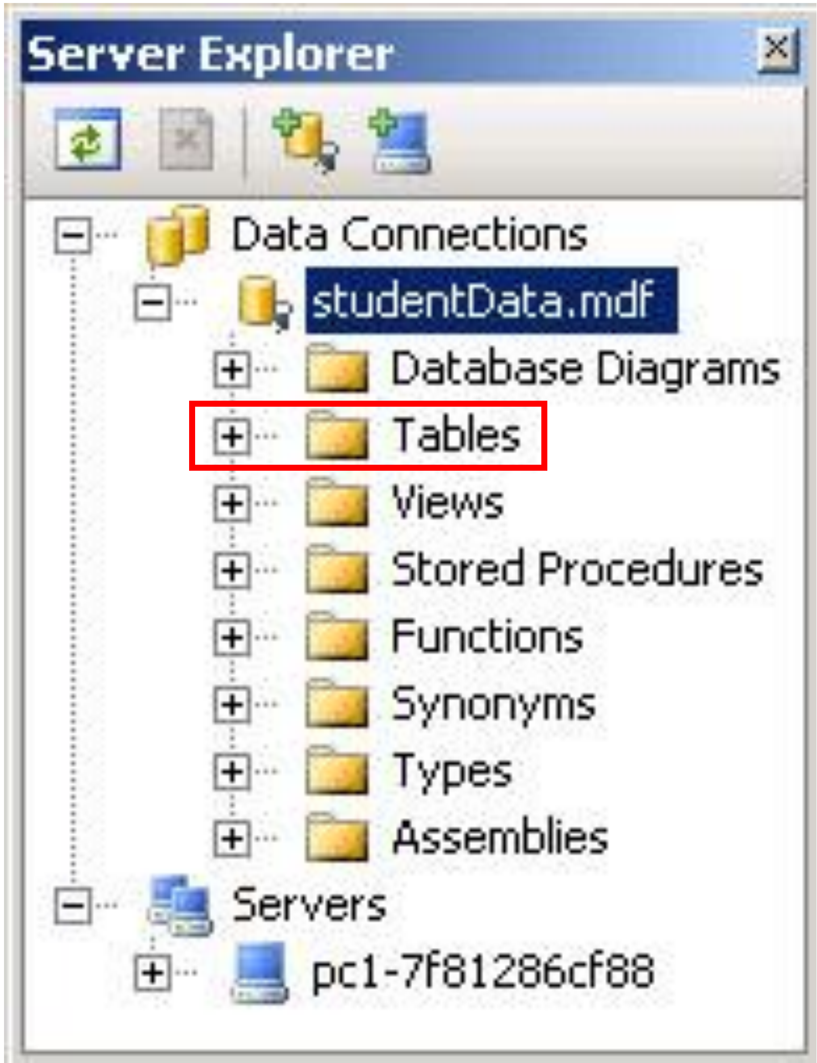
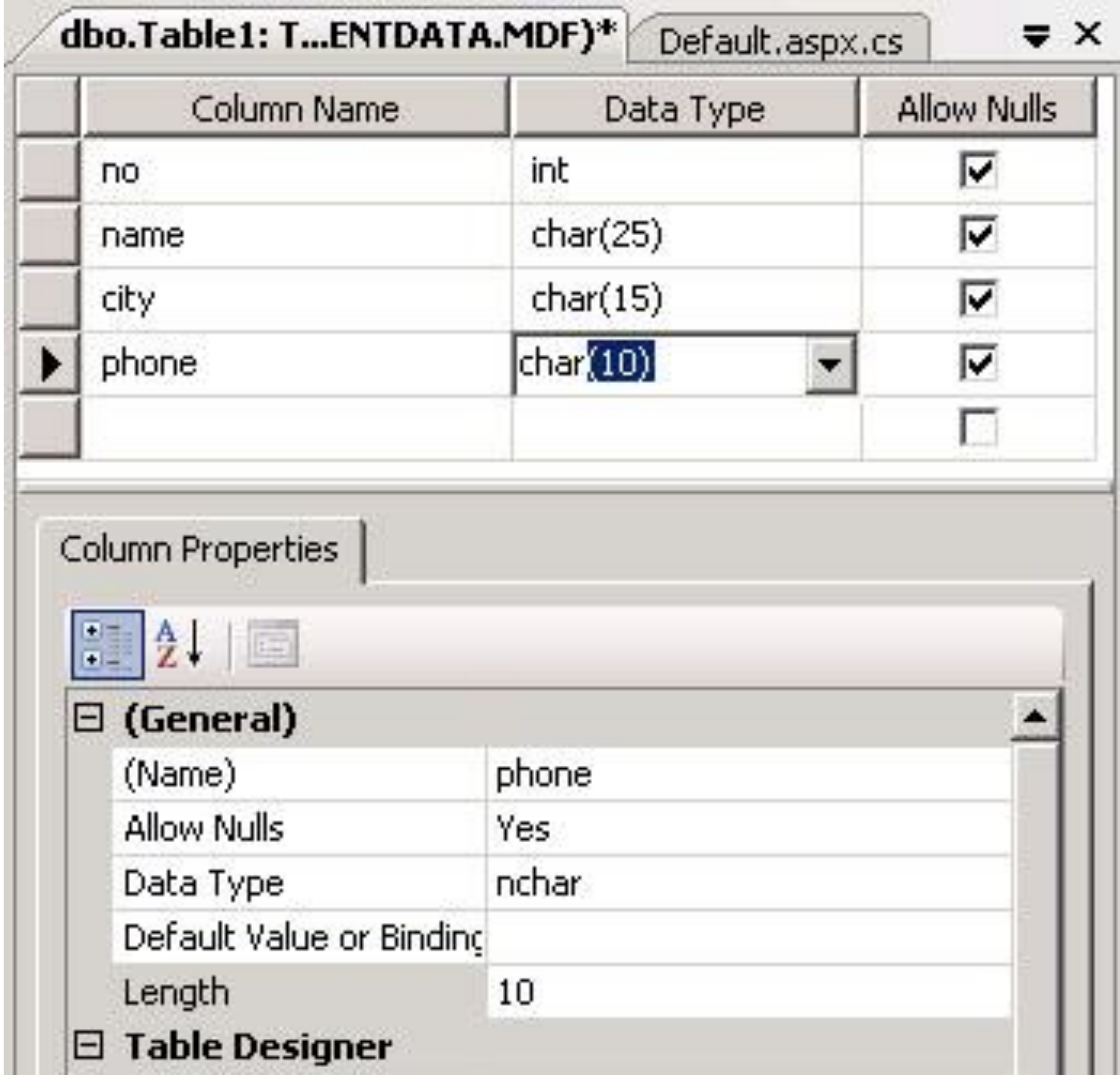


Table Structure :

- We can create a table as given here…



Column Name	Data Type	Allow Nulls
no	int	☑
name	char(25)	☑
city	char(15)	☑
phone	char(10)	☑
		□

Column Properties

⊟ **(General)**	
(Name)	phone
Allow Nulls	Yes
Data Type	nchar
Default Value or Binding	
Length	10
⊟ **Table Designer**	

Def. Make tables as given… and enter 10 Records for Each…

- Create a database table to store information of **PHONEBOOK** with Given Fields…
  - SrNo, Name, Address, City, PhoneNumber
- Create a database table to store information of **STUDENTResult** with Given Fields…
  - Class, Rno, Name, Sub1, Sub2, Sub3, Sub4, Sub5, Sub6

Def. Make changes in table like datatype, fieldname etc and save it.

- ALTER database table to store information of **PHONEBOOK** with Given Fields…
  - SrNo, Name, Address, City, PhoneNumber
- ***NOTE :***
  - If you found ERROR for updation, then
    - Goto Tools Menu -> Options
      - Expand the **Database Tools Node**
        - Goto Table and Database Designers
        - Uncheck ***Prevent saving changes that require table re-creation.***

IMP Note :

If you enter more then two records with SAME data, then it will not update the record NOR delete the record.

- We can also DELETE/UPDATE the records using SQL Queries.
- To fire a query we can ***right click*** on table and select NEW QUERY the write and execute the query.
  - **Example of Delete QUERY :**

    DELETE FROM Phone WHERE srno = 1

PrimaryKEY & AutoNumber

- We can add a filed as PRIMARYKEY by
  - Right Click on filed and select
    - Set Primary KEY
- We can add AUTORecord FIELD in the table.
  - Select Primary KEY Field.
    - Column Property
      - Select Identify Specification
        - Is Identify = Yes
        - Identity Increment = <Value>
        - Identity Seed = <value>

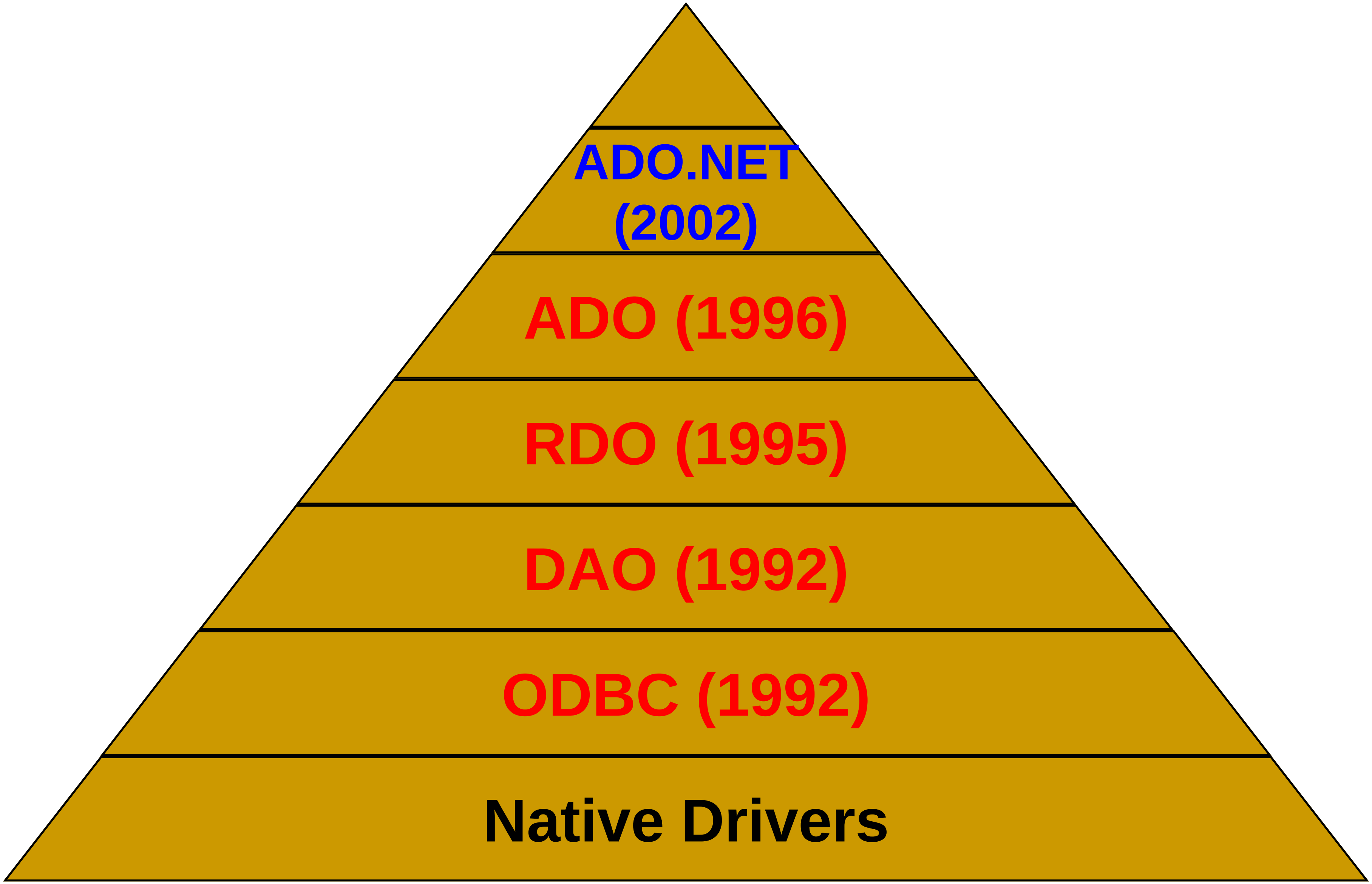
Def. Make tables as given... and enter 10 Records for Each...

- Create a database table to store information of **EmployeeMaster** with Given Fields…
  - EmpID, EmpName, Address, City, Contact, Qulification, JoinDate
- Create a database table to store information of **EmployeeDesignation** with Given Fields
  - EmpID, Dept, Designation, JoinDate, Salary

Def. Make tables as given… and enter 10 Records for Each…

- Create a database table to store information of **EmployeeAttendance** with Given Fields
  - EmpID, Date, TimeIn, TimeOut, Remarks
- Create a database table to store information of **Holidays** with Given Fields…
  - SrNo, Date, NameOfHoliday, Reason

Database Connection technology



ADO.NET (2002)

ADO (1996)

RDO (1995)

DAO (1992)

ODBC (1992)

Native Drivers

Introduction…

- Data access is the main part of any application that you develop.
- More concern is given on how you store the data and manipulate the data.
- Any web based application or window based applications stores the data for accessing the data for different purpose.
- Database programming is important aspect for any language that we learn or use for our development.

Introduction…

- ADO.NET is a group of libraries provided by .NET framework which are used to create powerful databases using various source such as MS SQL, Microsoft Access, Oracle, XML etc.
- It relies (આધાર રાખવો) on various classes. These classes to process requests and performs transition between a database system and the user.

Introduction…

- It provides classes for connecting with a data source, submitting queries, and processing results.
- Thus ADO.NET is a large set of .NET classes which enables us to retrieve and manipulate data, and update data sources in many different ways.

Advantages of ADO.NET

- Connection with any database:
  - ADO.NET can be used to connect with any type of database such as SQL server, Oracle, Access, MySQL etc.
  - It can also connect with any third party database.
- Connected and Disconnected Architecture:
  - It supports both connected and disconnected architecture.

Advantages of ADO.NET...

  - Connected architecture supported by DAO, ADO, RDO or ODBC.
  - Disconnected architecture is new feature provided by ADO.NET.
  - You will study about the disconnected architecture in the future topics.
- XML support:
  - ADO.NET provides the new feature of XML support.
  - Today XML is used widely for transfer data globally.

Advantages of ADO.NET…

  - XML is standard data transfer features for any type of data transfer.
  - Using ADO.NET you can easily convert data from dataset to XML and perform operations in secure and fast way.
- Fast, Scalable and Standard:
  - ADO.NET is quite fast then other applications.
  - It is quite scalable and it can be expanded.

Advantages of ADO.NET…

  - It also provides XML and other features thus it is quite standard used.
- Cross Language Support:
  - ADO.NET provides supports for many languages.
  - You can use VB, C#, or J# for ADO.NET programming.
  - So, it provides cross language support features.

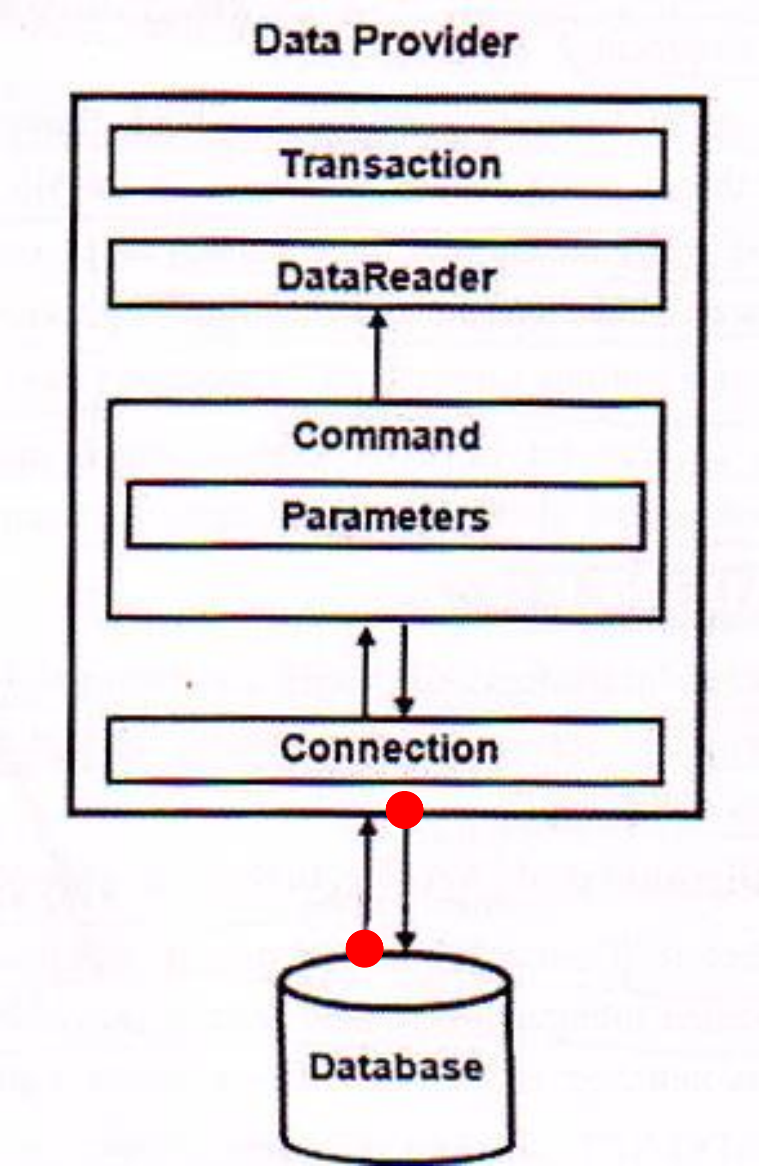
ADO.NET architecture

- The two important components of ADO.NET are:
  - DataSet and
  - Data Providers.
- Before understanding this two terms lets go through the two architecture provided by ADO.NET.
- ADO.NET provides two types of architecture for accessing data via database:
  - Connected Architecture and
  - Disconnected Architecture.

Connected Architecture…

- Connected Architecture is the architecture of ADO.NET
- In connected architecture the connection to the database must be opened to access the data retrieved from database.
- It is built on the classes called connection, command, datareader and transaction.

Connected Architecture...



Components of Connected Architecture

- Connection
  - As the name suggest, its main purpose is to establish a connection to database.
  - This connection should remain open each time any transaction occurs.
- Command
  - This object is used to execute SQL queries against database.
  - Using command object you can execute select, insert, update and delete SQL command.

Components of Connected Architecture

- DataReader
  - It is used to store the data retrieved by command object and make it available for .NET application.
  - Data available in the DataReader is read only one time and it can navigate only in forward direction.
  - It can also read only one record at a time.

Components of Connected Architecture

- Transactions:
  - Enables you to get the list of commands in transactions at the data source.

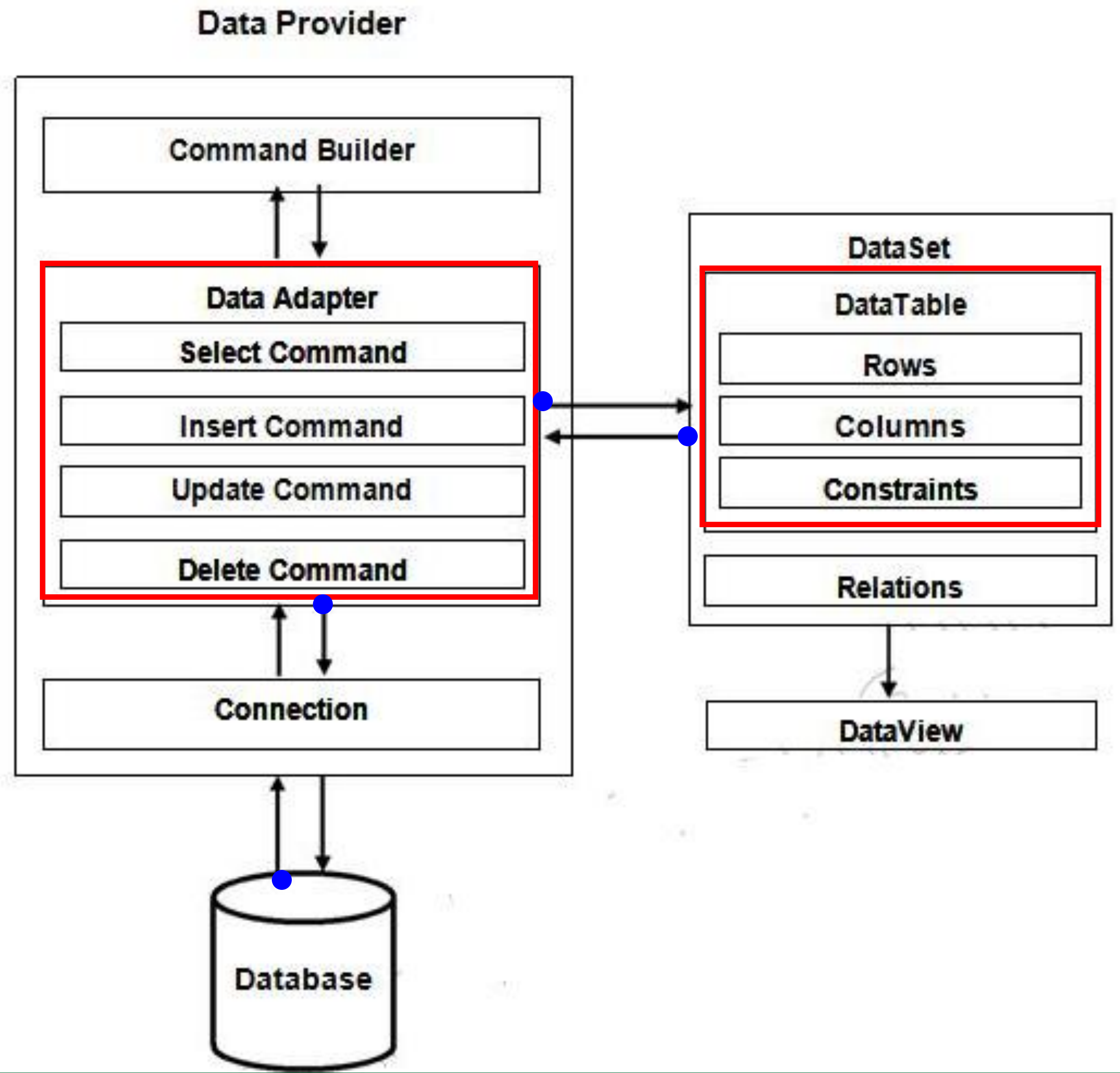
Disconnected Architecture :

- Disconnected architecture of ADO.NET is one in which data retrieved from database can be accessed even when the connection to database is closed.
- It is built on connection, data adapter, command builder and dataset classes.
- In disconnected architecture, the copy of the data is loaded form database and kept in local memory of client.

Disconnected Architecture :

- Database is only connected during the copying of the data.
- After copying the data the connection to the database is lost.
- We can perform insert, update, delete or select operation on the local of the database.
- The updated database is in our local memory only.
- When we connect to the database again then the local copy of the database is copied into the final database.
- In this way data consistency(સુસંગતતા) is maintained.

Disconnected Architecture :



Components of Disconnected Architecture

- Connection:
  - Connection Object is used to establish connection between the data source and database.
- Data Adapter:
  - DataAdapter object is used to transfer the data between database and dataset.
  - It provides commands like select, insert, update and delete.
  - Data Adapter needs an open connection to transfer the data.

Components of Disconnected Architecture

- DataSet:
  - It is collection of table that is used to store the data retrieved from database into local memory.
  - We can load single or multiple tables into DataSet.
- DataTable:
  - It is used when we want to load only one table into local memory.

Components of Disconnected Architecture

- DataRow:
  - It is used to add or remove one or multiple rows into DataSet or DataTable.
- DataColumn:
  - It is used incase if you want to modify any column or add new column into DataSet or Datatable.

Components of Disconnected Architecture

- DataView:
  - It is a view of table available in DataSet.
  - It is used for many purposes such as to find a record, sort the records and filter the records.
  - We can also perform insert, update and delete operations on the table.

Difference between Connected and Disconnected Architecture :

Connected Architecture	Disconnected Architecture
It is connection oriented architecture.	It is connection less architecture.
It uses DataReader for communication between Database and application	It uses DataSet for communication between Database and application
It gives faster performance as the application is always connected with the database.	It gives low in speed and performance as the database at all time.

Difference between Connected and Disconnected Architecture :

Connected Architecture	Disconnected Architecture
Connected Architecture can work with a single table at a time.	Disconnected Architecture can work with multiple table at a time.
DataReader cannot persist(ડેટા હાજર રહેતો નથી) the data.	DataSet can persist (ડેટા હાજર રહે છે) the data.
It is only used Read only mode, you cannot update the data.	In this we can update the data, as the data is copied locally.

Difference between Connected and Disconnected Architecture :

Connected Architecture	Disconnected Architecture
▪ More traffic would be create if the number of users increases on the server side.	▪ In this there is no concern on the traffic as the data is copied locally.
▪ Much memory of the Server is used while processing connected architecture.	▪ Much memory of server is used once only when you copy the data into local memory, after that local memory of each user would be used.

Data Provider :

- Data Providers are the one which does the work of communication between application and database.
- They are responsible for maintaining the connection to the database.
- Different classes are provided this data providers which helps to perform different database operations either in connected or disconnected modes.

Data Provider :

- A data provider is a set of classes provided by ADO.NET.
- Data Provider allows you to access a specific database, execute SQL commands and retrieve data.
- ADO.NET provides a rich set of Data Providers which allows you to do operations on any type of databases.
- For different types of databases, different types of data provides are provided by ADO.NET

Data Provider :

- There are 4 types of data providers provided by Microsoft :
  - ❑ SQL Server provider :
    - This provider is designed specially to work SQL Server Database.
  - ❑ Oracle Provider :
    - This provider deals with Oracle Database

Data Provider :

- OLEDB Provider :
  - This provider works with any database that has OLE DB driver.
  - It can access any data from any type of database using OLEDB driver.
- ODBC Provider :
  - This provider deals with any database which has ODBC driver.
  - Almost all the database comes with ODBC driver inbuilt and we can connect it with any database.

NameSpaces for DataProvider

- In ADO.NET there are namespaces which are specific to these data providers.
- System.Data
  - It is important namespace which is used in disconnected architecture.
  - This namespace give classes such as DataSet, DataTable, DataRelation, DataRow, DataColumn etc.
  - This namespace is automatically added when you add a new Web Form.

NameSpaces for DataProvider

- System.Data.SqlClient
  - This namespace is used when we want to connect with SQL Server Data Provider.
  - It provides special classes to deal with SQL Server.
  - It gives classes such as SQLConnection, SQLCommand, SQLDataReader, SQLDataAdapter etc.

NameSpaces for DataProvider

- System.Data.OracleClient
  - This namespace is used when we want to connect with Oracle Data Provider.
  - It provides special classes to deal with Oracle database.
  - It gives classes such as OracleConnection, OracleCommand, OracleDataReader, OracleDataAdapter etc.

NameSpaces for DataProvider

- System.Data.Oledb
  - This namespace is used when we want to connect to any database which provides Oledb Driver.
  - It gives classes such as OledbConnection, OledbCommand, OledbDataReader, OledbDataAdapter etc.

NameSpaces for DataProvider

- System.Data.ODBC
  - This namespace is used when we want to connect to any database which provides ODBC Driver.
  - It gives classes such as ODBCConnection, ODBCCommand, ODBCDataReader, ODBCDataAdapter etc.

Connection Object :

- In ADO.NET if you are using Connected Architecture or Disconnected Architecture for both first you will need to obtain (મેળવવું) connection.
- To establish a connection we must specify some property first.

Connection Object : SqlConnection Property

Property	Meaning
ConnectionString	It is used to specify the connection string which is used to open SQL Server Connection.
ConnectionTimeout	It is used to get or set the timeout period for the connection till which it would try to establish a connection. After connection timeout, it raises error.

Connection Object : SqlConnection Property

Property	Meaning
Database	It provides the name of database server with which we will be connected or going to be connected after opening the connection.
DataSource	It provides the name of database instance with which we will be connected.

Connection Object : SqlConnection Property

Property	Meaning
ServerVersion	It gives the version details of SQL Server with which you are connected.
WorkStationID	It gives the ID of the client with which we are connected now.
State	It indicates the state of SqlConnection class which you are connected with.

Connection Object : SqlConnection Methods

Method	Meaning
Open	It is used to open the connection with database.
Close	It is used to close the connection with database.
BeginTransaction	It allows us to start a transaction. This is used in case if you are doing a number of transactions at a time.

Connection Object : SqlConnection Methods

Method	Meaning
ChangeDatabase	It allows us to change the name of database. We can change of database runtime by using this method.
CreateCommand	It allows you to create object of SQLCommand which is used in connected architecture.

Connection Object : SqlConnection Methods

- For connected architecture we need to specify the open or close method().
- For disconnected architecture we do not need to specify the open or close method(). This will done automatically by DataAdapter method of the disconnected architecture.

Connection Object : SqlConnection Property

- Connection String :
  - It is a string which holds details like user ID, password, server IP address and the database name to which connection is done.
  - ConnectionString is note case sensitive.
  - It has a specific format where each of the details have certain keywords which need to be given as is and the different parameters would be separated by ‘;’.
  - In C# “\” has a special meaning so we need to write “\\” or append “@” before the connection string.

Connection Object : SqlConnection Property

- Connecting SQL Server using SQLConnection :
    - As we are using SQL Server class for connection, we do not need to specify the provider for SQL Server.

Steps To Connect with DATABASE

Step – 1

- Add required NameSpace

```
Using System.Data;
Using System.Data.SqlClient;
```

Step – 2

- Add required Objects in

```
public partial class   Default:System.Web.UI.Page
SqlConnection <ConName>;
SqlCommand <CommandName>;
SqlDataAdapter <DataAdapter Name>;
DataSet <DataSet Name>;
```

Steps To Connect with DATABASE

Step – 3

- Establish a connection using connection string. (Goto Page Load Event)

```
<ConName>=new SqlConnection
                    (@“<Connection String>”);
```

@ Will read all the symbols as text

Steps To Connect with DATABASE

Step – 3

Connection String :

Data Source=

.\SQLEXPRESS;AttachDbFilename=<Path**&**FileName>;

Integrated Security=True; User Instance=True”);

NOW Open Connection using

<ConnectionName>.Open();

Steps To Connect with DATABASE

Step – 3 [Example]

```
using System.Data;
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{  SqlConnection Con;
   SqlCommand Cmd;
   protected void Page_Load(object sender, EventArgs e)
   { Con = new SqlConnection(@"Data Source=.\
          SQLEXPRESS;AttachDbFilename=
          C:\ADO\App_Data\Database.mdf;
      Integrated Security=True; User Instance=True");
     Con.Open();
   }
}
```

Steps To Connect with DATABASE

Step – 4

- Write Commands as per requirements in Button Events…

```
<command>=new SqlCommand("
                    <SQL Query>",
                    <ConnectionName>);
```

- Execute Specified Command :

```
<Command>.ExecuteNonQuery();
```

Steps To Connect with DATABASE

Step – 4 [To add new Text data in Table]

```
protected void Button1_Click(object sender, EventArgs e)
{Cmd = new SqlCommand("Insert into
              AddressBook Values('Gohil M. A.',
              'Lathi','Sanghavi Street')", Con);
  Cmd.ExecuteNonQuery();
}
```

Note : Text must store in Single Inverted ‘ ’.

Def. * Design a webpage to add data in following table.

Field	Data Type	Size
SrNo	Number	Number
Name	Character	20
City	Character	20
Address	Character	30

- Add 10 records using 10 buttons…

Steps To Connect with DATABASE

Step – 4 [To add new Numeric data in Table]

```
protected void Button1_Click(object sender, EventArgs e)
{Cmd = new SqlCommand("Insert into Fees
               values(1,5000,’11/06/1977’)", Con);
  Cmd.ExecuteNonQuery();
}
```

Note : Numeric Data must store Without inverted commas.

Def. * Design a webpage to add data in following fee table.

Field	Data Type	Size
ReceiptNo	Number	Int
Fees	Number	Int
Date	Varchar	10

- Add 10 records using 10 buttons…

Steps To Connect with DATABASE

Step – 4 [To add new Text data in Table using TEXTBOX Control]

```
protected void Button1_Click(object sender, EventArgs e)
{Cmd = new SqlCommand("Insert into AddressBook
   Values('"+txtName.Text+"','"+txtCity.Text+"',
                 '"+txtAddress.Text+"')", Con);
  Cmd.ExecuteNonQuery();
}
```

Note : Text must store in Single Inverted ‘ ’.

Def. * Design a webpage to add data in following table using textbox. (NOTE: Textbox Must Not BLANK…)

Field	Data Type	Size
Name	Character	20
City	Character	20
Address	Character	30

- Use textbox for multiple records…

Steps To Connect with DATABASE

Step-4[To add Numeric data in Table using textbox]

```
protected void Button1_Click(object sender, EventArgs e)
{Cmd = new SqlCommand("Insert into Fees
        values(" + Int32.Parse(txtGR.Text) + ","
        + Int32.Parse(txtFees.Text) + ")", Con);
  Cmd.ExecuteNonQuery();
}
```

Note : Numeric Data must store Without inverted commas.

Def. * Design a webpage to add data in following table using textbox. . (NOTE: Textbox Must Not BLANK…)

Field	Data Type	Size
EmpId	Number	6
EmpName	Varchar	25
Designation	Varchar	15
Salary	Number	7

- Use textbox for multiple records…

Def. * Design a webpage for student result and add data in following table using textbox with validation.

Field	Data Type	Size	Validation
Seatno	Numeric	Int	2501 to 3000
Name	Varchar	20 Char	Only Alpha + Space
Sub1	Numeric	Int	0 to 100
Sub2	Numeric	Int	0 to 100
Sub3	Numeric	Int	0 to 100
Sub4	Numeric	Int	0 to 100

Display :: Total, Result, Percent, Class

DataAdapter to Fill Data in GridView

DataAdapter

First of All define

- **DataSet DS;**
- **SqlDataAdapter DA;**
  - Display Data In GridView

```
<DataAdapter>=new SqlDataAdapte(
        <Select Query>,<Connection>);
<DataSet>=new DataSet();
<DataAdapter>.Fill(<DataSet>,”Table”);
GridView1.DataSource=<DataSet>;
GridView1.DataBind();
```

Steps To Connect with DATABASE

Gridview EXAMPLE :

```
public partial class _Default : System.Web.UI.Page
{   SqlConnection Con;
    SqlCommand Cmd;
    DataSet DS;
    SqlDataAdapter DA;
    protected void Page_Load(object sender, EventArgs e)
    {   Con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=F:\Educational MATERIAL\MAT\TERM-1\01. ASP.net\Demo\W1\App_Data\Database.mdf;Integrated Security=True; User Instance=True");
        Con.Open();     }
```

Steps To Connect with DATABASE

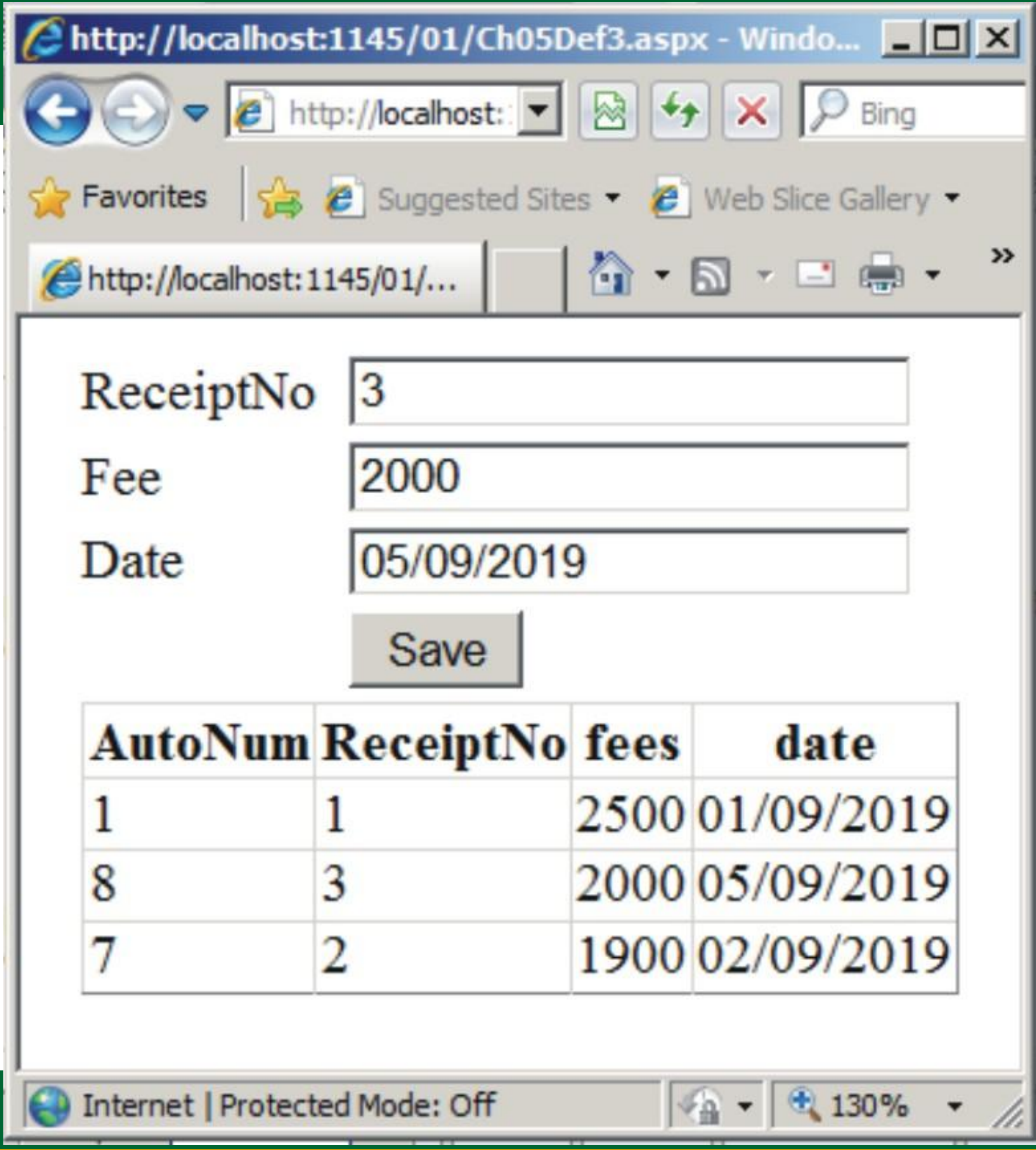
Gridview EXAMPLE :

```
protected void btnSave_Click(object sender, EventArgs e)
{ Cmd = new SqlCommand("Insert into phoneBook
      Values('"+txtSrNo.Text+ "','"+txtName.Text+"','"
      +txtCity.Text+"','"+txtNumber.Text+"')", Con);
    Cmd.ExecuteNonQuery();
    DA=new SqlDataAdapter("Select * from phonebook",Con);
          DS=new DataSet();
          DA.Fill(DS,"Phonebook");
          GridView1.DataSource=DS;
          GridView1.DataBind();
     }
}
```

Def. * Add records in FEE and display all the records in GRID. As Given…

Note :

- After save, data must displayed in grid…
- Before Save check textbox for blank…
- After Save clear all fields…



http://localhost:1145/01/Ch05Def3.aspx - Windo...
http://localhost:
Bing
Favorites
Suggested Sites
Web Slice Gallery
http://localhost:1145/01/...
Internet | Protected Mode: Off
130%

ReceiptNo	3
Fee	2000
Date	05/09/2019

Save

AutoNum	ReceiptNo	fees	date
1	1	2500	01/09/2019
8	3	2000	05/09/2019
7	2	1900	02/09/2019

Data Connectivity Using Dynamic path

- **Required Name Space**

```
using System.Data.SqlClient;
```

- **TO define Default PATH**

```
string appPath = System.AppDomain.
                    CurrentDomain.BaseDirectory;
```

- **To set app path…**

```
Con = new SqlConnection(@"Data
   Source=.\SQLEXPRESS;AttachDbFilename="
   + appPath + @"App_Data\
   database.mdf;Integrated Security=True;
   User Instance=True");
```

Command Object (SqlCommand object)

- Some of properties of SqlCommand can be given as :
  - Connection
    - It provides the information regarding the connection object that you are using for connection with database. That connection should be open before execution command.
  - CommandText
    - It provides the sql queries that will be fired such insert, update, delete or select commands. You can directly specify the sql command constructor.

Command Object (SqlCommand object)

- properties of SqlCommand can be given as :
  - CommandType
    - It is used to set or get the type of command object. It can be either text or stored procedure.
  - CommandTimeout
    - It allows you to set or get the time out period for command object. If time out occurs, then error would be generated.
  - Parameters.Count
    - It gives the total number of parametsrs specified in the command text.

Command Object (SqlCommand object)

- properties of SqlCommand can be given as :
  - ❑ Parameters
    - It used to specify the parameters that would be passed in the parameterized query. It can set by using symbol "@".
    - For Example:

      string query=select * from employee where empId= @empid

      - ❑ Here @empid indicates that there is one parameter passed in the command text. You can specify as many parameter you want.

Command Object (SqlCommand object)

- properties of SqlCommand can be given as :
  - Parameters[0].Value
    - It allows you to get or set value of first parameter in command text. Here 0 indicates the first parameter of command text.
  - Parameters[0].Size
    - It allows you the set the size of the column which is specified as parameter in the command text.

Command Object (SqlCommand object)

- properties of SqlCommand can be given as :
  - Parameters[0].SourceColumn
    - It gives the column name from database tables.
    - For example, in query select * from employee where empId= @empid.
    - Here @empid is connected with empId.
    - In this case it would give empId column from database table.

Command Object (SqlCommand object)

- properties of SqlCommand can be given as :
  - Parameters[0].SqlDbType
    - It gives the column data type from database table.
- **Methods** of SqlCommand can be given as:
  - ExecuteNonQuery
    - This method is used when we are using transactional queries like inser, update or delete.
    - It returns integer values as return value indicating the total number of rows affected by the given query.
    - It return value is>0 then command is successfully executed else it has some error.

Command Object (SqlCommand object)

- **Methods** of SqlCommand can be given as:
  - ExecuteReader
    - This method is used to read data from table using Select query.
    - It helps you to read all record from table.
  - ExecuteScalar
    - This method is used to execute the query, and return the first column of first row in the result set returned by the query.
    - Additional columns or rows will be ignored.
  - ExecuteXmlReader
    - This method is used to read data in XML format when you are using XML file.

Command Object (SqlCommand object)

- **Methods** of SqlCommand can be given as:
  - Cancel
    - This method is used to cancel command text from being executed.
  - CreateParameter
    - This method is used to create parameter.
    - After creation of parameters the parameters would be added to Parameters collections.
  - Parameters.Add()
    - It allows you to add parameter to parameters collection. In this method you need to specify the name, data type and size of parameter. By this we need to add the parameter first and then value can be specified later.

Command Object (SqlCommand object)

- **Methods** of SqlCommand can be given as:
  - Parameters.AddWithValue( )
    - It allows you to add parameter along with its value at the same time. In this data type and size will be automatically taken.
  - Parameters.Clear( )
    - It allows you to remove all parameters values from parameter collection.
  - Parameters.Remove( )
    - It removes all parameters from Parameter Collection.
  - Parameters.RemoveAt( )
    - It removes the specified parameter whose value is given from Parameter Collection

Command Object (SqlCommand object)

- One of the important events of SqlCommand:
  - StatementCompleted
    - This event occurs when the SQL command has finished executing at the database end.

To put a msgbox in ASP.NET

- We can add a msgbox in ASP.NET using script language…
- For Example:

```
Response.Write("<script
                  Language='JavaScript'>
                  alert('Testing');
                  </script>");
```

DataReader Object (SqlDataReader)

- Some Properties of SqlDataReader
  - HasRows
    - It is a Boolean property which is used to find out whether the data reader contains some rows or not.
  - FiledCount
    - It gives you the total number of field count which are loaded under SqlDataReader.
  - IsClosed
    - It is Boolean property which is used to check whether the data reader is closed or not after reading the data.

DataReader Object (SqlDataReader)

- Some Methods of SqlDataReader :
  - Read
    - This method is used to read a reacord from SqlDataReader class.
    - It point to the first record and returns true or false to indicate whether a next record exists or not.
    - When you initialize data reader, you need to called Read() method at first the data reader does not point to first record of the table.
  - Close
    - It is used to close DataReader pointer which is connected to database table.
  - IsDBNull
    - It is a Boolean method. It is used to determine whether the column data is null or not. If it is null it would return true else it would return false.

DataReader Object (SqlDataReader)

DEF. : Design A web Page for LOGIN WINDOW using DetaReader **HasRow** property.

- **Coding :**

```
Cmd = new SqlCommand("select * from login where
                    userId='"+txtUser.Text+"' and
                    pass='"+txtPass.Text+"'",Con);


SqlDataReader Dr;
Dr = Cmd.ExecuteReader();
if (Dr.HasRows)
   Response.Write("Yes, There Are Many Records");
else
   Response.Write("Sorry! There are no Records");
```

Def. Design a WEB page for USER login windows with given data table.

- UserId
- UserName
- Password
- LoginCount

DataReader Object (SqlDataReader)

DEF. : Design A web Page to Enter new data into table and check that the data is duplicate or not.

- **Coding :**

```
Cmd = new SqlCommand("select * from login where
                    userId='"+txtNo.Text+"' and
                    pass='"+txtName.Text+"'",Con);
SqlDataReader Dr;
Dr = Cmd.ExecuteReader();
if (Dr.HasRows)
   Response.Write(“Sorry! It is Duplicate Record");
else
   //Code for Add NEW Record
```

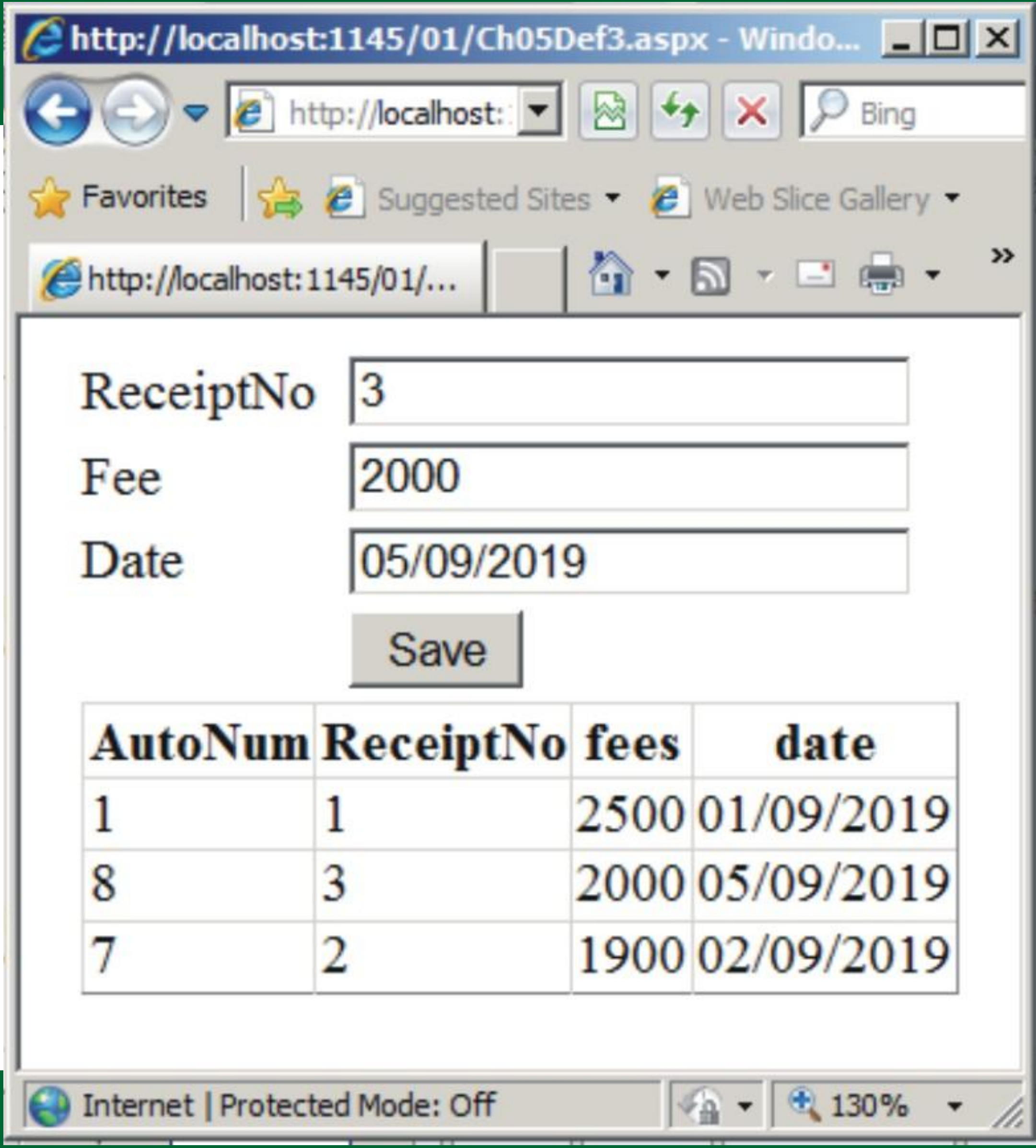
Check DUPLICATE RECORD

- If you want to check that given data is already available in the table.
- It means that the same record is available in the table and if you add the same data AGAIN then it will the **DUPLICATE RECORD**.

Def. * Add records in FEE and display all the records in GRID. As Given…

Note :

- Duplicate RECORD must not ADD in Table…



http://localhost:1145/01/Ch05Def3.aspx - Windo...
http://localhost:
Bing
Favorites
Suggested Sites
Web Slice Gallery
http://localhost:1145/01/...
Internet | Protected Mode: Off
130%

ReceiptNo	3
Fee	2000
Date	05/09/2019

Save

AutoNum	ReceiptNo	fees	date
1	1	2500	01/09/2019
8	3	2000	05/09/2019
7	2	1900	02/09/2019

Delete A record From Table

- We can delete specified record form any table using query…

```
protected void btnDel_Click(object sender, EventArgs e)
{ Con = new SqlConnection(@“ConnectionString");
  Con.Open();
  Cmd = new SqlCommand("delete from
            phonebook where
            no='"+txtNo.Text+"'", Con);
  Cmd.ExecuteNonQuery();
  Con.Close();
}
```

Update A record in a Table

- We can update specified record form any table using query…

```
protected void btnDel_Click(object sender, EventArgs e)
{ Con = new SqlConnection(@“ConnectionString");
  Con.Open();
  Cmd = new SqlCommand(“update from
            phonebook where
            no='"+txtNo.Text+"'", Con);
  Cmd.ExecuteNonQuery();
  Con.Close();
}
```

DataReader Object (SqlDataReader)

DEF. : Design A web to display PhoneBook on Screen Using entry control loop **While**.

- **Coding :**

```
Cmd =  new SqlCommand
                  ("select * from PhoneBook", Con);
SqlDataReader Dr;
Dr = Cmd.ExecuteReader();
if (Dr.HasRows)
   while (Dr.Read())
    {Response.Write(“No :" + Dr["No"].ToString());
     Response.Write("Name :" + Dr[1].ToString());
     Response.Write("City :" + Dr[2].ToString());
     Response.Write("Phone :" + Dr[3].ToString()); }
else
   Response.Write("Sorry! There are no Records");
```

Disconnected Architecture :

- Disconnected Architecture is new concept related to ADO.NET.
- In this data will be loaded into local memory and then all the operation will done on the local copy of the database.
- After doing all operations data will be again copied back to the original database table to maintain consistency. (સુસંગતતા)
- In disconnected Architecture, data is fetched into local memory in form of DataSet or DataTable. At last this DataSet or DataTable is updated back to the actual database.

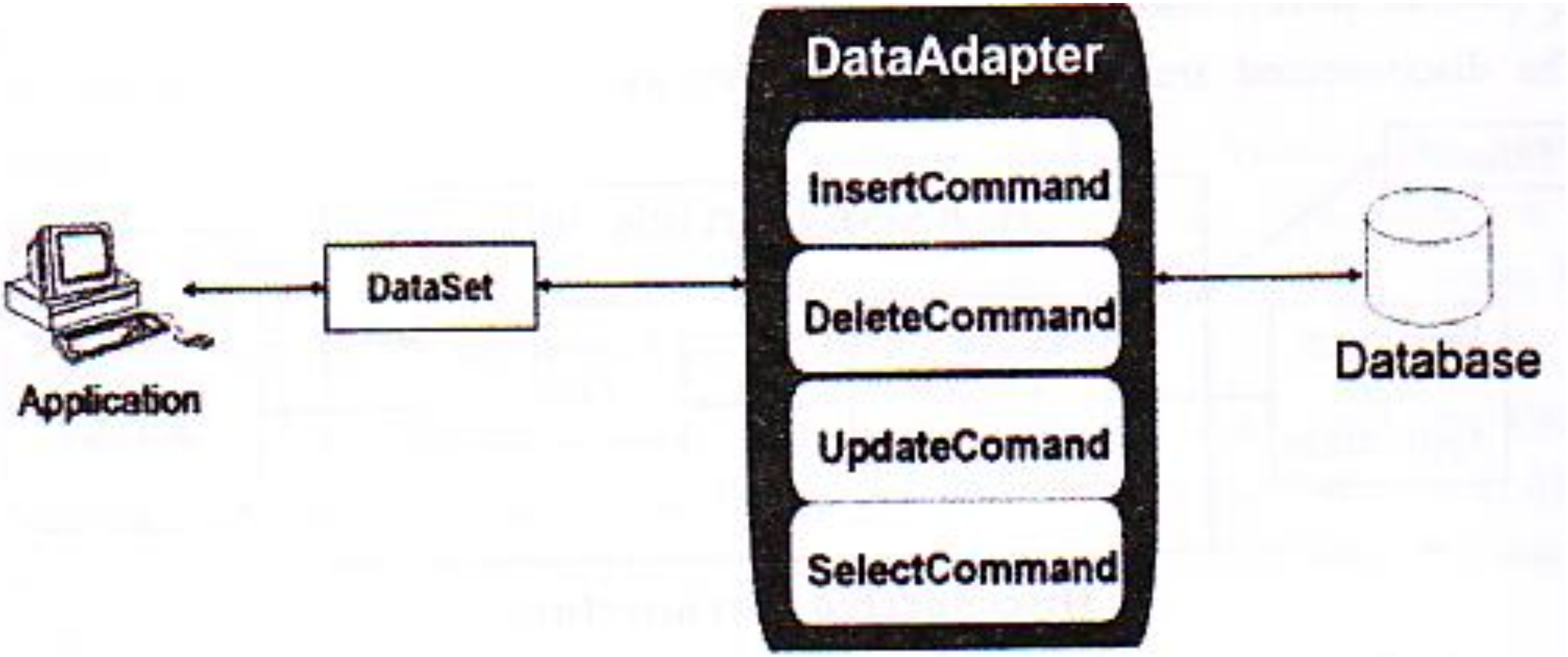
Disconnected Architecture :

- Classes used in Disconnected Architecture:
  - DataAdapter:
    - It is one which fills dataset or datatable and again updates the data back to the database.It is one the important property of the disconnected architecture.
  - DataSet :
    - It is used when we want to load some group of tables into local memory.
    - DataSet is collection of tables.
  - DataTable :
    - It is used when we want to load some group of tables into local memory.

Disconnected Architecture :

- Classes used in Disconnected Architecture:
  - DataRow :
    - It is used when you want to add or remove rows from DataSet or DataTable.
  - DataColumn :
    - It is used in case if you want to modify, add or remove any column from DataSet or DataTable.

DataAdapter Object (SqlDataAdapter):



- DataAdapter is bridge between our database and DataSet.
- It fills and updates the DataSet using its properties and methods.

DataAdapter Object (SqlDataAdapter):

- Properties Of DataAdapter:
  - SelectCommand
    - This property is used to generate the select command.
  - InsertCommand
    - This property is used to generate the insert command automatically. If we have inserted some new rows in the DataSet or DataTable, SqlCommandBuilder automatically generates Insert command and updates the final database.

DataAdapter Object (SqlDataAdapter):

- Properties Of DataAdapter:
  - UpdateCommand
    - This property is used to generate the Update Command automatically.
    - If you have updated some rows in the DataSet or DataTable, SqlCommandBuilder automatically generates Update command and updates the final database.
  - DeleteCommand
    - This property is used to generate the Delete command automatically. If we have delete some rows from the DataSet or DataTable, SqlCommandBuilder automatically generates Delete command and update the final database.

DataAdapter Object (SqlDataAdapter):

- Methods Of DataAdapter:
  - DataAdapter.Fill( ) method:
    - It is one of the most important methods of DataAdapter. It is used to fill the data into DataSet or DataTable.
    - For filling the data into dataset in DataAdpater we need to use the Select command only
  - DataAdapter.Update( )method:
    - DataAdapter is used to update the changes made to DataSet or DataTable into the actual Database tables.
    - Before using Update( ) method we need to use the Insert, update, delete etc commands for this SqlCommandBuilder class would be used.

DataAdapter Object (SqlDataAdapter):

- Events Of DataAdapter:
  - FillError :
    - This event will be raised when there will be any error generated while filling data into DataSet or DataTable.
  - Row**Updating**
    - This event will be raised at the time of updating the content of rows. At the time of call of update( ) method, DataAdapter makes changes to actual table. While updating the rows this event would be fired.
  - Row**Updated**
    - This event will be raised after the changes to the content of rows of table are done. At the time of call of update() method, DataAdapter makes changes to the actual table. After Changes are done, this event will be fired.

DataSet Object :

- DataSet is a collection of tables.
- If we want to do operations on multiple tables at a time DataSet would be used.
- DataSet is only available with Disconnected Architecture.
- Properties of DataSet
  - DataSetName
    - It allows us to set or get the name of DataSet Object.
  - HasErrors
    - It returns a Boolean value to specify whether there are errors while filling or updating data.

DataSet Object : Properties of DataSet

- IsInitialized
  - It returns a Boolean value which specify whether the data is stored in it is initialized with some data or not.
- Relations
  - It returns a set of relations between tables loaded in the DataSet.
- Tables
  - It is the collection of all tables loaded in DataSet object.
- Tables.Count
  - It returns a value specifying how many tables are filled into dataset.

DataSet Object : Properties of DataSet

- Tables[0].TableName
  - It returns the name of first table loaded in the DataSet. Instead of table index you can also specify the name of the table.
- Tables[0].Rows.Count
  - It gives the total number of rows in the first table loaded in the DataSet.
- Tables[0].Column.Count
  - It gives the total number of columns in the first table loaded in the DataSet.

DataSet Object : Properties of DataSet

- Tables[0].PrimaryKey
  - It gives all the primary keys associated with first table in form of any array.
- Tables[0].Constraints
  - It gives all the constraints associated with particular table in form of ConstraintsCollection.

- Methods of DataSet
  - Clear
    - It clears all the data inside DataSet. All tables stored inside the dataset will be clear after this method is called.

DataSet Object : Methods of DataSet

- AcceptChanges
  - After calling this method the changes made to dataset would be reflected back to the database table.
- WriteXml
  - It allows you to create XML file from dataset data.
- ReadXml
  - It allows you to read XML file and copy it into dataset.
- Table[0].NewRow
  - It allows you to create a blank data row in the first table of dataset collection.

DataSet Object : Methods of DataSet

- Table[0].Select
  - It allows you to select group of data from First table of dataset collection.
- Table[0].Rows.Add
  - It allows you to add a new data row for the first table of dataset collection.
- Table[0].Rows.Remove
  - It allows you to remove a row from first table of dataset collection.
- Table[0].Rows.RemoveAt
  - It allows you to remove a row specified by an index number from first table of dataset collection.

DataTable Object:

- DataTable is generalized concept which is not related with any Data Provider.
- It is given under System.Data namespace.
- It can contain only one table data within it.
- So while using DataTable we can work only one table at a time.
- Properties and methods of DataTable are somewhat as the DataSet Tables properties and methods.

DataTable Object : Property of DataSet

- TableName :It gives you the name of table. Here you need to provide just this property with the object of DataTable.
  - It can be given as:

    ```
    string tableName=dt.TableName;
    ```
- Rows.Count : It gives you the total number of rows in DataTable Object.
- Columns.Count : It gives you the total number of columns in DataTable Object.
- PrimaryKey : It gives all the primary keys associated with table as an array.
- Constraints : It gives you all the constraints associated with table in form of ConstraintsCollection.

DataTable Object : Methods of DataSet

- Clear
  - It is used to clear the data inside DataTable.
- AcceptChanges
  - It allows us to accept the changes to DataTable. After calling this method the changes made to data table are reflected to database table.
- WriteXml
  - It allows us to create Xml file from the data given in the datatable.
- ReadXml
  - It allows you to read Xml file and copy it into database.

DataTable Object : Methods of DataSet

- NewRow
  - It allows us to create a new row object.
- Select
  - It allows us to select the data from DataTable.
- Row.Add
  - It allows us to add new row to DataTable.
- Row.Remove
  - It allows you to remove row from DataTable.
- Row.RemoveAt
  - It allows you to remove specific row from DataTable. The row is specified by index number.

Def. DAW to display data from a table to Screen Using DataAdapter and DataTable.

```
Con = new SqlConnection(@“<Connection String>");
Da = new SqlDataAdapter("select * from phonebook", Con);
Dt = new DataTable();
Da.Fill(Dt);
for (int row = 0; row < Dt.Rows.Count; row++)
{
   for (int col = 0; col < Dt.Columns.Count; col++)
    {   string colName = Dt.Columns[col].ColumnName;
        string colValue = Dt.Rows[row][col].ToString();
        Response.Write("<br>" + colName + ":" + colValue);
   }
   Response.Write("<hr>");
}
```

DataRow Object :

- DataRow is generalized concept which is not related to any data provider. It is also available under System.Data namespace.
- DataRow can contain single row of any single row of any table at a time.
- Whenever you store any table into dataTable, it will automatically generate its column as per table.
- For example, if your table has 10 columns, when you copy one row of DataTable into DataRow object, it will automatically crate 10 columns for itself as per DataTable object.

DataRow Object : Properties

- HasError
  - It is a Boolean property which specifies whether DataRow has got some error or not.
- ItemArray
  - It returns the collections of columns in form of object array.
- RowError
  - It allows you to get or set an Error Message if any error occurs
- RowState
  - It gives you the state of row whether rows is being updated, delated, insertd etc.
- Table
  - It gives the name of DataTable with which your DataRow is attached with.

DataRow Object : Methods

- AcceptChanges
  - This method accepts the changes made to the DataTable so far.
- RejectChanges
  - This methods rejects the changes made to the DataTable so far.
- BeginEdit
  - This method allows you to edit the DataRow object data.
- EndEdit
  - This method ends the editing of the DataRow object.
- CancelEdit
  - This method cancel the editing of DataRow object in between.
- Delete
  - This method allows you to delete the specified row from DataTable object.

DataColumn Object :

- DataColumn is the generalized object which is not related to any specific data provider. It also provided by System.Data namespace.
- DataColumn contains a single column of DataTable or DataSet. It has information regarding column like name, data type, default value, maximum length etc.
- Property of DataColumn:
  - AllowDbNull :
    - It is Boolean property which specifies whether you want to allow null values for particular column or not.
  - ColumnName
    - It allows us to set or get the name of DataColumn object’s column.

DataColumn Object : Property

- Caption
  - It gives or set the caption of DataColumn object’s column.
- DataType
  - It allows us to set or get the DataType of particular column object.
- DefaultValue
  - It allows us to set default value of DataColumn object.
- Expression
  - It allows us to set default value of DataColumn object.
- MaxLenght
  - It allows us to set the maximum length of DataColumn object data.
- Table
  - Allows us to get the name of table object with which DataColumn is attached to.

DAW to add new record in Database Table : Using SqlCommandBuilder.

```
Con = new SqlConnection(@“<Connection String>");
Da=new SqlDataAdapter("select * from
   phonebook",Con);
DataTable Dt=new DataTable();
Da.Fill(Dt);
DataRow Dr= Dt.NewRow();
Dr[0]=txtNo.Text;
Dr[1]=txtName.Text;
Dr[2]=txtCity.Text;
Dr[3]=txtPhone.Text;
Dt.Rows.Add(Dr);
SqlCommandBuilder Scb=new SqlCommandBuilder(Da);
Da.Update(Dt);
```

To add a new row to the database table, SqlCommandBuilder is used. It would automatically generate an insert command to reflect the changes done to **table in SQL server.**

Data Binding

- Data Binding is new popular concept.
- Data Binding simply means you are connecting your control with any of the table or table column or data of a particular row.
- Technically, it means data is retrieved from source content and then it is connected with the control to property of visual element which we could see.
- In ASP.Net when we bind our data with any of the control, it gets you data from specified data source. Data Source could be anything such as a DataSet, DataTable or even DataRow. But in most of the cases it is either DataSet or DataTable.

Data Binding

- The control which is binded with some specific DataSet or DataTable is called Bounded control as it bounded with specific column of the table.
- There are basically two types of binding:
  - Simple Data Binding
  - Complex Data Binding
- Simple Data Binding
  - In Simple Data Binding we are connected to any single piece of information, or you can say we are bounded to single column of single row.
  - For Example,
    - If a textbox is bounded with single column of table, it is called Simple Data Binding.

Data Binding

- Complex Data Binding
  - In Complex Data Binding, we are connected to entire table itself, or you can we are bound to the table with which we are connected with.
  - We can also be bounded to single column of single row of table in Complex Data Binding.
  - For Example,
    - If drop down list is bounded to a single column of table or if datagrid is bounded to particular table then it is called **Complex Data Binding.**

Data Binding

- The controls like Data Grid, Grid View or Repeater control are also known as Complex Data Binding controls which are bound to entire table,
- For example

```
GridView1.DataSource=DataTable;
GridView1.DataBind( );
```

- Here, we assigning DataSource property to DataTable. Then the DataBind property is used to bind the data from DataTable to GridView.

GridView and Repeater Control :

- GridView and Repeater control are complex bind controls. They are able to show entire table control at time as data.
- GridView Control:
  - GridView control is a successor to ASP.NET DataGrid Control.
  - It provides more flexibility for displaying and working with data from your database in comparison with any other controls available.
  - The GridView control enables you to connect to datasource and display the data in a tabular format. It also provides options to customize the look and feel.

GridView Control :

- The GridView view mode is used to display a list of data items by binding the data fields to columns and by displaying a column header to identify the field.
- Default GridView Style implements buttons as column headers. This column header is used for interaction capabilities.
- Example
  - For sorting GridView data according to a specified column.

GridView Control : Properties

- AllowPaging
  - It is Boolean property which indicates whether the control support paging or not.
- AllowSorting
  - It is Boolean property which indicates whether the control support sorting or not.
- SortExpression
  - It is used to get the current row on which sorting is done.
- DataSource
  - It gets the DataSource object that contains the data to populate the control.

GridView Control : Behavior Properties

- DataSourceId
  - It indicates the data bound control to be used.
- AutoGenerateEditButton
  - It is Boolean property which provides a separate column that would be added to edit the record.
- AutoGenerateDeleteButton
  - It is Boolean property which provides a separate column that would be added to delete the record.
- AutoGenerateSelectButton
  - It is Boolean property which provides a separate column that would be added to select the record.
- AutoGenerateColumns
  - It is Boolean property which indicates whether columns are automatically created for each field of the data source. Default is true.

GridView Control : Style Properties

- AlternatingRowStyle
  - It defines the style for every alternate row in the GridView.
- EditRowStyle
  - It defines the style or row in EditView.
- RowStyle
  - It defines the style of the rows of the GridView.
- PagerStyle
  - It is used to define the style properties of Pager of the GridView. If AllowPaging=true, the page number row appears in this style.

GridView Control : Style Properties

- EmptyDataRowStyle
  - It defines the style properties of the empty row, which appears if no records are bound to the control.
- HeaderStyle
  - It defines the style for the header of the gridview
- FooterStyle
  - It defines the style for the footer of the gridview.

GridView Control : Appearance Properties

- CellPadding
  - It indicates the space in pixel between the cells and the border of the GridView.
- CellSpacing
  - It inidicates the space in pixel between cells
- GridLines
  - It can have any one of the given value: Both/Horizontals/Vertical/None. It indicates whether GridLines should appear or not.
- HorizontalAlign
  - It indicates the horizontal align of the GridView.
- EmptyDataText
  - It indicates the text to appear when there is no record in the data source.

GridView Control : Appearance Properties

- ShowFooter
  - It indicates whether the footer should appear or not.
- ShowHeader
  - It indicates whether the header should appear or not.
- BackImageUrl
  - It indicates the location of the image that should display as a background of the GridView.
- Caption
  - It is used to get or set the caption of the GridView.
- CaptionAlign
  - It can take any one of the following alignment values: Left/Center/Right.

GridView Control : State Properties

- Columns
  - It is used to get the collection of objects that represent the columns in the GirdView.
- EditIndex
  - It is used to get or set the 0-based index that identifiers the row currently to be edited.
- FooterRow
  - It returns a GridViewRow object represents the footer of the GridView.
- HeaderRow
  - It returns a GridViewRow object that represents the header of the GirdView.

GridView Control : State Properties

- PageCount
  - It get the number of the pages required to display the record of the data source.
- PageIndex
  - It is used to get or set the 0-Based page index.
- PageSize
  - It is used t get or set the number of records to display in one page of GridView.
- Rows
  - It gets a collection of GridViewRow objects that represents the currently displayed rows in the GridView.

GridView Control : State Properties

- DataKeyName
  - It gets a collection of GridViewRow objects that represents the currently displayed rows in the GridView.
- DataKeys
  - It gets a collection of DataKey objects that represent the value of the primary key fields set in DataKeyNames property of the GridView.

GridView : Events Associated :

- PageIndexChanging, PageIndexChanged
  - It provides both events occur when the page link is clicked. They are fired before and after GridView handles the paging operation respectively.
- RowCancelingEdit
  - It is used to fires when cancel button is clicked in Edit mode of GridView.
- RowCommand
  - It is used to files when a button is clicked on any row of GridView.

GridView : Events Associated :

- RowCreated
  - It is used to fires when a new row is created in GridView.
- RowDataBound
  - It is used to fires when row is bound to the data in GridView.
- RowDeleting, Row Deleted
  - It provides both events fires when Delete button of a row is clicked. They are fired before and after GridView handles deleting operation of the row respectively.

GridView : Events Associated :

- RowEditing
  - It is used to fires when a Edit button of a row is clicked but before the GridView handles the Edit operation.
- Row Updating, RowUpdated
  - It is used to fire both events when a update button of a row is clicked. They are fired before and after GridView control update operation respectively.
- Sorting, Sorted
  - It is used to fire both events when column header link is clicked. They are fired before and after the GridView handler the Sort operation respectively.

Connection String Using Web.Config

- What is the requirement of connection string using **Web.Config**?
  - Basically, we use **Connection string on each and every page of the website.**
  - In that case, whenever we need to change database then we must change connection string on each and every webpage of the website.
  - In place of that, we can use connection string on Web.Config file.

Connection String Using Web.Config

- Step to set connection string in Web.Config.
- Step-1
  - Set Connection String in Web.Config

```
<connectionStrings>
  <add name=“<Name of the Connection>”
  connectionString=“<ConnectionString>”
  providerName="System.Data.SqlClient"
  />
  </connectionStrings>
```

Connection String Using Web.Config

- Step-2
  - Set Connection String in **file.aspx.cs** file.

```
String CS = ConfigurationManager.
            ConnectionStrings["ConString"].
            ConnectionString;
SqlConnection Con = new
            SqlConnection(CS);
Con.Open();
```

- Step-3
  - Set required codes and run the program.

Def. Set Connection String using web.config and fill phone data to gridview control.

```
protected void Page_Load( )

{ String CS = ConfigurationManager.

       ConnectionStrings

       ["ConString"].ConnectionString;
```

Continue…

Continue...

```
using (SqlConnection Con=new SqlConnection(CS))
{ SqlCommand Cmd = new
        SqlCommand("select * from phone",Con);
        Con.Open();
        GridView1.DataSource =
                            Cmd.ExecuteReader();
        GridView1.DataBind();
  }
}
```

Repeater Control :

- Repeater is a Data Bind Control.
- Data Bind controls are also called Container control.
- In Data Bind there is a link created between the data source and the presentation UI for displaying the data.
- ASP.NET provides rich and wide variety of controls, for bounding to the data.

Repeater Control :

- The Repeater control is used for displaying a repeated list of items which are bounded to the control.
- The Repeater control can be bounded to a database table or an XML file or any other list of items.

Repeater Control :

- In a Repeater control, data is rendered in form of DataItems which are defined using one or more templates.
- For binding, we can use HTML tags such as<li>, <ul> or <div> tag.
- It also has a DataSource property which is used to set the DataSource of this control.

Repeater Control :

- Once this property is set, the data is bounded with the Repeater control using DataBind( ) method.
- The Repeater control does not support paging or editing of data.
- It is just a display control which displays data bounded. So it is also called a light weight control.

Repeater Control :

- The main advantage is that it displays data quickly and formats the data to be displayed quite easily.
- **Repeater control provides 5 templates for formatting the output data:**

  <HeaderTemplate>

  <ItemTemplate>

  <AlternatingTemplate>

  <SeparatorTemplate>

  <FooterTemplate>

Repeater Control :

<HeaderTemplate>

- It is used to display Header text for the DataSource Collection and they apply different styles for header text.

<FooterTemplate>

- It is used to display the footer text for the DataSource Collection.

<ItemTemplate>

- It is used for elements that are rendered once per row of data from data collection. It is used to display records.

Repeater Control :

<AlternatingTemplate>

- It is used to change the background styles and color for the alternate rows of data from data collection. It only works for even number of rows.

<SeparatorTemplate>

- It is used to determine the separator element which separates each item from item collection, such as line breaks.

Binding Repeater Control with DataSource :

- Repeater control does not have any specific view or any visual effect.
- We need to bind Repeater control with some data source like DataSet or DataTable, but for that we also need to specify which columns should be displayed manually. It will not display columns automatically like girdview.

Binding Repeater Control with DataSource :

- To bind the content of particular column to Repeater control column, you need to use DataBinder class as follows:

```
<%DataBinder.Eval(Container.DataItem.“Name”)%>
```

  - *DataBinder class provides an Eval( ) method.* This method is used to specify the particular column of Data Source with the Repeater control’s column would be bounded.

Binding Repeater Control with DataSource

- Steps To Add Repeater Control…
- Step-1 (Add namespace **System.Configuration;**)
  - Set connection string in **web.config**
  - Above <system.web>

```
<connectionStrings>

    <add name="ConnectionString"

    connectionString="Data Source=.\SQLEXPRESS;

    AttachDbFilename=C:\WebSite1\App_Data\Db.mdf;

    Integrated Security=True;User Instance=True"/>

</connectionStrings>
```

Binding Repeater Control with DataSource

Step-2 Set Connection String

```
protected void Page_Load( )

{ String CS = ConfigurationManager.

      ConnectionStrings

      ["ConString"].ConnectionString;
```

Continue…

Continue...

Step-3 Set Command and DataBinding

```
using (SqlConnection Con=new SqlConnection(CS))
{ SqlCommand Cmd = new
      SqlCommand("select * from phone",Con);
      Con.Open();
      Repeater1.DataSource =
                        Cmd.ExecuteReader();
      Repeater1.DataBind();  }
}
```

Binding Repeater Control with DataSource :

- **Steps-4 Add Repeater Control to webpage**
  - Add Repeater Control in Current Webpage

```
<asp:Repeater ID="Repeater1"
 runat="server‟>

        .... Contain ....

</Repeater>
```

Binding Repeater Control with DataSource :

- **Step-5 Add contents…**
  - Add Header Tmplate

```
<HeaderTemplate>
<table border="1" cellpadding="5"
                              cellspacing="2">
      <tr> //Add Required HEADERS
         <th>No</th>
         <th>Name</th>
         <th>City</th>
         <th>Phone</th>
      </tr>
</HeaderTemplate>
```

Binding Repeater Control with DataSource :

- **Step-6 Add Item Template**

```
<ItemTemplate>
<tr> //Add Required Field Items
    <td><%#DataBinder.Eval(Container.DataItem,"No")%></td>
    <td><%#DataBinder.Eval(Container.DataItem,"Name")%></td>
    <td><%#DataBinder.Eval(Container.DataItem,"City")%></td>
    <td><%#DataBinder.Eval(Container.DataItem,"Phone")%></td>
</tr>
</ItemTemplate>
```

Binding Repeater Control with DataSource :

- **Step-7 Add AlternatingItem (*If required*)**

```
<AlternatingItemTemplate>
<tr> //Add Required Alternating Field Items
   <td><%#DataBinder.Eval(Container.DataItem,"No")%></td>
   <td><%#DataBinder.Eval(Container.DataItem,"Name")%></td>
   <td><%#DataBinder.Eval(Container.DataItem,"City")%></td>
   <td><%#DataBinder.Eval(Container.DataItem,"Phone")%></td>
</tr>
</AlternatingItemTemplate>
```

Binding Repeater Control with DataSource :

- **Step-8 Add Footer Template**

```
<FooterTemplate>
   //Add Clossings
   </table>
</ FooterTemplate>
```

-> PROGRAM Is READY to EXECUTE.

Image :

- We are not stores the image directly in the database as it would take more space and also it quit slow.
- We can store images in local folder inside the web application and retrieving the image from the same folder.

Def : Design a web page to maintain data with given fields.

- Table Fields :

  SrNo, Name, City, Phone, PhotoPath

Modification… In Item Template…

```
<ItemTemplate>
<tr> //Add Required Field Items
   <td><%#DataBinder.Eval(Container.DataItem,"No")%></td>
   <td><%#DataBinder.Eval(Container.DataItem,"Name")%></td>
   <td><%#DataBinder.Eval(Container.DataItem,"City")%></td>
   <td><%#DataBinder.Eval(Container.DataItem,"Phone")%></td>
   <td><asp:Image ID="imgID" Height="50" Width="50"
            runat="Server" ImageUrl=
                  '<%#DataBinder.Eval
                  (Container.DataItem,"photopath") %>'
         />
   </tr>
</ItemTemplate>
```