Inbuilt Functions: Mathematical Functions

- Math functions are very useful function when we need to perform mathematical operations such as find a square root, absolute number, exponential value, and Log value so for all these we have in-built functions in VB.Net.
- To use these math functions, import the **System.Math** namespace.
- The common mathematical functions in Visual Basic are **Abs, Exp, Fix, Int, Log, Rnd, Round and Sqrt**.

Method	Description	Example
Abs	Returns the absolute value of a specified number.	Dim no1 As Double = Math.Abs(50.3) 'Output is 50.3. Dim no2 As Double = Math.Abs(-50.3) 'Output is 50.3.
Exp	Returns a Double value containing e (the base of natural logarithms) raised to the specified power.	Dim no1 As Double = Math.Exp(2) 'Output is 7.39.
Fix	It remove the fractional part of Number and return the resulting integer value.	Dim no1 As Double = Math.Fix(-8.4) 'Output is -8
Int	It remove the fractional part of Number and return the resulting integer value. It return smallest integer value for negative numbers.	Dim no As Double = Math.Int(-8.4) 'Output is -9
Log	Returns a Double value containing the logarithm of a specified number. This method is overloaded and can return either the natural (base e) logarithm of a specified number or the logarithm of a specified number in a specified base.	Dim no1 As Double = Math.Log(2) 'Output is 0.6931
Rnd	It generates a random number which returns a value between 0 and 1	Randomize() Dim no1 As Integer = Int((6 * Rnd()) + 1) 'Output is 4 (random no between 1 to 6)
Round	It is used to return whole number nearest to value for the specified number of decimal places.	Dim no1 As Double = Math.Round(4.4) 'Output is 4
Sqrt	Returns a Double value specifying the square root of a number.	Dim no1 As Double = Math.Sqrt(4) 'Output is 2

- Example of all the above function is given below:

```
Module Module1
    Sub Main()
        Console.WriteLine("Square root of the number is : " & Math.Sqrt(16))
        Console.WriteLine("Absolute value of the number is : " & Math.Abs(-45.67))
        Console.WriteLine("Expontential values is : " & Math.Exp(2))
        Console.WriteLine("Fix number is : " & Fix(-4.5))
        Console.WriteLine("Integer number for a +Ve Number is " & Int(7.8))
```

```
        Console.WriteLine("Integer number for a -Ve Number is " & Int(-7.8))
        Console.WriteLine("Log value of the number is : " & Math.Log(2))
        Console.WriteLine("Random Number is : " & Rnd(8))
        Console.WriteLine("Round Number is : " & Math.Round(67.83999, 2))
        Console.ReadKey()
    End Sub
End Module
```

String manipulation

- In VB.Net, you can use strings as array of characters.
- The **String** class of the .NET framework provides many built-in methods to facilitate the comparison and manipulation of strings.
- The string keyword is an alias for the **System.String** class.

Method	Description	Example
Mid	Returns a string containing a specified number of characters from a string.	Dim MidExample As String = "Mid Function Demo" Dim no As String = Mid(MidExample, 5, 8) 'Output is "Function".
Right	Returns a string containing a specified number of characters from the right side of a string.	Dim RightExample As String = "Hello World!" Dim no As String = Right(RightExample, 6) 'Output is "World!".
Left	Returns a string containing a specified number of characters from the left side of a string.	Dim LeftExample As String = "Hello World!" Dim no As String = Left(LeftExample, 5) 'Output is "Hello".
Trim	It removes leading and trailing whitespace. It returns a string containing a copy of a specified string with no leading or trailing spaces.	Dim TrimExample As String = " Apple " Console.WriteLine(TrimExample.Trim()) 'Output is Apple
Ltrim	It returns a string containing a copy of a specified string with no leading spaces.	Dim LtrimData As String = " Apple " Console.WriteLine(LtrimData.Ltrim()) ' Output is “Apple “
Rtrim	It returns a string containing a copy of a specified string with no trailing spaces.	Dim RtrimData As String = " Apple " Console.WriteLine(RtrimData.Rtrim()) ' Output is “ Apple”
InStr	It returns an integer specifying the start position of the first occurrence of one string within another.	Dim InstrExample As String = "Hello World!" Dim no as Integer= InStr(InstrExample, "e") ' Output is 2
Ucase	It returns a string after converting to uppercase.	Dim no As String = UCase("Apple") 'Output is APPLE

Lcase	It returns a string after converting to lowercase.	Dim no As String = LCase("Apple") 'Output is apple
Chr	Returns the character associated with the specified character code (ASCII Value).	Dim no As String = Chr(65) 'Output is A
Asc	Returns an Integer value representing the character code corresponding to a character.	Dim no As Integer = Asc("A") 'Output is 65
Format	Returns a string formatted according to instructions contained in a format String expression.	Dim TestDateTime As Date = "1/27/2001 5:04:23 PM" Dim no As String = Format(TestDateTime, "hh:mm:ss tt") 'Output is 05:04:23 PM
ToString	Converts the value of the instance to a String.	Dim theDate As Date = #12/25/2016# Console.WriteLine(theDate.ToString("MMMM d, yyyy")) 'Output is December 25, 2016

- Example of all the above function is given below:

```
Module Module1
    Sub Main()
        Dim Engg_Clg As String = "Darshan Institute of Engineering & Technology"
        Console.WriteLine("Engineering and Technology= " & Mid(Engg_Clg, 1, 20))
        Console.Write("Rajkot = " & Right(Engg_Clg, 10))
        Console.WriteLine()
        Console.WriteLine("College Name= " & Left(Engg_Clg, 7))
        Dim Dip_Clg As String = " DIETDS-CE "
        Console.WriteLine(Trim(Dip_Clg))
        Console.WriteLine(LTrim(Dip_Clg))
        Console.WriteLine(RTrim(Dip_Clg))
        Console.WriteLine("Alphabet Find at position: " & InStr(1, "Darshan", "r",
        CompareMethod.Text))
        Console.WriteLine("Lower Case: " & LCase(Engg_Clg))
        Console.WriteLine("Upper Case: " & UCase(Engg_Clg))
        Console.WriteLine("Character At: " & Chr("65"))
        Console.WriteLine("Asscii values is: " & Asc("DIET"))
        Console.ReadKey()
    End Sub
End Module
```

What is ADO.NET?

- ADO stands for **ActiveX Data Objects.**
- ADO.NET is a database technology of .NET Framework **used to connect application system and database server.**
- ADO.NET is a part of the **.NET Framework** providing access to relational data, XML documents, and application data.
- ADO.NET consists of a **set of classes used to handle data access.**
- ADO.NET uses XML to store and transfer data among applications, which is not only an industry standard but also provide fast access of data for desktop and distributed applications.
- ADO.NET is **scalable and interoperable**.

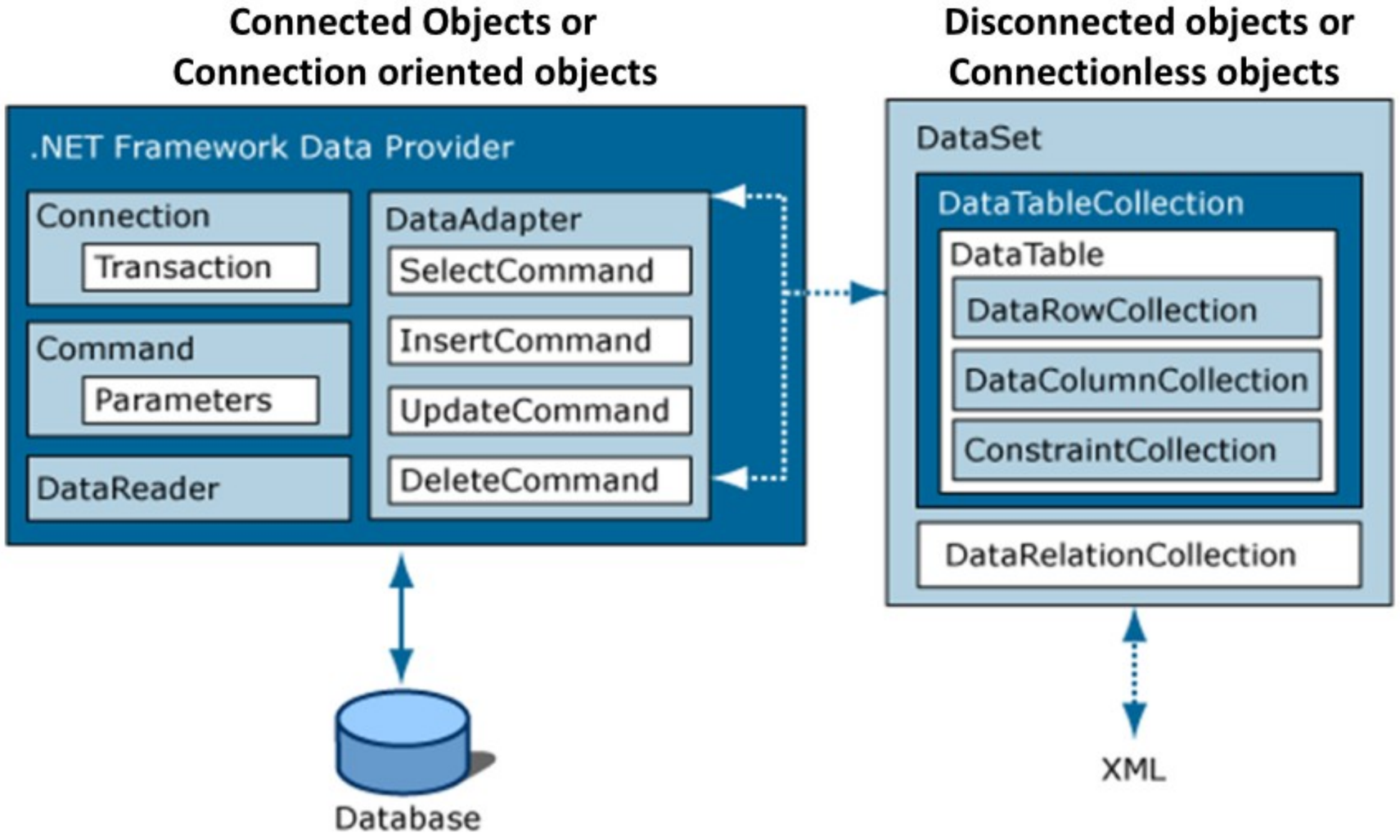

Fig 5.1 ADO.NET Architecture

- The ADO.NET architecture has two main parts:

1. **Data Provider** (Connected Objects or Connection oriented objects)
2. **Data Set** (Disconnected objects or connectionless objects)

Data Provider (Connection Oriented Objects)

- The .NET framework Data Provider is component that has been explicitly designed for data manipulation and fast, forward-only, read-only access to data.
- .NET Framework data provider is used for connecting to a database, executing commands, and retrieving results.
- The Data Provider has four core objects:
  - **Connection:** The Connection objects provide connectivity to a data source.
  - **Command:** The Command object enables access to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information.

- **Data Reader:** The Data Reader provides a high-performance stream of data from the data source.
- **Data Adapter:** The Data Adapter provides the bridge between the Data Set object and the data source. It uses command object to execute SQL commands at the data source to both load the Data Set with data and reconcile changes that were made to the data in the dataset back to the data source.

- The following lists the data providers that are included in the .NET framework.

Data Provider	Description
SQL Server	Provides data access for Microsoft SQL server. It uses the System.Data.SqlClient namespace.
OLEDB	For data sources exposed by using OLEDB. It uses the System.Data.OleDb namespace.
ODBC	For data sources exposed by using ODBC. It uses the System.Data.Odbc namespace.
Oracle	For Oracle data sources. It uses the System.Data.OracleClient namespace.

Data Set (Disconnected objects)

- DataSet is an in-memory representation of data.
- It is a **disconnected**, cached set of records that are retrieved from a database.
- When a connection is established with the database, the data adapter creates a dataset and stores data in it.
- After the data is retrieved and stored in a dataset, the connection with the database is closed. This is called the **'disconnected architecture'**.
- The dataset works as a virtual database containing tables, rows, and columns.
- The DataSet class is present in the **System.Data** namespace.
- The dataset has two major objects:
  - **The Data Table Collection**
    - ✓ The Data table collection contains all the data table objects in a dataset.
    - ✓ A Data table is defined in the **System.Data** namespace and represents a single table of memory-resident data.
    - ✓ It contains a collection of columns represented by a data column collection, and constraints represented by a constraint collection, which together define the schema of the table.
  - **The Data Relation Collection:**
    - ✓ A relationship represented by the Data relation object, associated rows in one Data table with rows in another Data table.
    - ✓ A relationship is analogous to a join path that might exist between primary and foreign key columns in a relational database.
    - ✓ The essential element of a data relation are: **The name of the relationship, the name of the tables being related, and the related column in each table.**

Properties of DataSet Object

- **CaseSensitive:** Gets or sets a value indicating whether string comparisons within DataTable. objects are case-sensitive.
- **DataSetName:** Gets or sets the name of the current DataSet.
- **Namespace:** Gets or sets the namespace of the DataSet.
- **Tables:** Gets the collection of tables contained in the DataSet.

Methods of DataSet Object

- **Clear()**: Clears data.
- **Clone()**: Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.
- **Copy()**: Copies both the structure and data.
- **Finalize()**: Free resources and perform other cleanups.
- **RejectChanges()**: Rolls back all changes made since the last call to AcceptChanges.
- **WriteXML()**: Writes an XML schema and data from the DataSet. This method has different overloaded forms.

Connection Object

- The Connection object is the first component of ADO.NET.
- The Connection objects provider connectivity to a data source.
- It establishes a connection to a specific data source.
- Connection object helps in accessing and manipulating a database.
- The base class for all Connection objects is the **DbConnection** class.
- For creating the connection we have to do as per follows, before that we have to use a namespace for connecting to database, here we had used Access database , so we had include the following namespace and code of line for creating connection:

```
Imports System.Data.OleDb;
Dim ocon as OleDbConnection= new OleDbConnection();
```

Properties of Connection Object

- **ConnectionString**: Connection String is collection of name/value pairs separated by semicolon which gives information of data source with which connection needs to be established.

  Syntax of connection string:

  Data Source=ComputerName\SQLInstance;Initial Catalog=DatabaseName; Integrated Security=False; User ID=username; Password=password;

  Example of connection string:

  Data Source=ANSPC\SQLEXPRESS;Initial Catalog= Seminar; Integrated Security=False; User ID=admin; Password=123;

- **Data Source**: It specifies name of computer and SQL server instance with which connection is required.
- **Provider:** It represents the name of the data provider used to establish a connection with the data source.

- **Initial Catalog**: It specifies name of database with which connection is required.
- **User ID**: It specifies username to connect with database.
- **Password**: It specifies password to connect with database.
- **Integrated Security**: It specifies that connection will be established using windows authentication or SQL Server username/password.
- **ConnectionTimeout**: It represents the time (in seconds) after which the attempt to connect is terminated and an error is generated.

Methods of Connection Object

- **Open():** This method opens connection using information provided by connection string.
- **Close():** This method closes already opened connection.
- **ChangeDatabase():** Changes the current database on an open connection.

Command Object

- A command is a SQL statement or a stored procedure used to retrieve, insert, delete or modify data in a data source.
- The Command object enables access to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information.
- It executes a command against a data source. Exposes Parameters and can execute in the scope of a Transaction from a Connection.
- You can execute SQL queries to return data in a DataSet or a DataReader object.
- Command object performs the standard Select, Insert, Delete and Update SQL operations.
- The base class for all Command objects is the Command class.

Properties of Command Object

- **Connection** : It specifies that on which connection command executes.
- **CommandType**: It specifies type of Command to execute:
  - ✓ **Text**: SQL Statement as command type.
  - ✓ **StoredProcedure**: Stored Procedure as command type.
  - ✓ **TableDirect**: Table Name as command type.
- **CommandText**: Either SQL Statement or name of Stored procedure or name of Database table.

Methods of Command Object

- **ExecuteNonQuery():** This method executes the command specifies and returns the number of rows affected.
- **ExecuteScalar():** This method executes the command specified and returns the first column of first row of the result set. The remaining rows and column are ignored.
- **ExecuteReader():** The ExecuteReader method executes the command specified and returns an instance of SqlDataReader class.
- **ExecuteXmlReader():** This method executes the command specified and returns an instance of XmlReader class. This method can be used to return the result set in the form of an XML document.
- **CreateCommand():** This method created new command object on given connection object.

Data Readers

- The Data Reader provides a high-performance stream of data from the data source.
- It reads a forward-only, read-only stream of data from a data source.
- DataReader object works in connected model.
- The base class for all DataReader objects is the DbDataReader class.
- When we started to read from a DataReader it should always be open and positioned prior to the first record.
- We can not use a DataReader to modify rows in the database.
- The **Read()** method in the DataReader is used to read the rows from DataReader and it always moves forward to a new valid row, if any row exist .

Properties of DataReader Object

- **Depth**: Indicates the depth of nesting for row.
- **FieldCount**: Returns number of columns in a row.
- **IsClosed**: Indicates whether a data reader is closed.
- **Item**: Gets the value of a column in native format.
- **RecordsAffected**: Number of row affected after a transaction.

Methods of DataReader Object

- **Close()**: Closes a DataReader object.
- **Read()**: Reads next record in the data reader.
- **NextResult()**: Advances the data reader to the next result during batch transactions.
- **GetDataTypeName()**: Returns the name of the data type for a column based on its ordinal.
- **GetName()**: Returns the name of a column based on its ordinal.
- **GetValue()**: Returns the value of a column based on its ordinal.

- Following below give the demonstration of **binding GridView control using Connection Oriented Objects** of ADO.Net

```
Dim objConnection As New SqlConnection()
objConnection.ConnectionString = " Data Source=ANSPC\SQLEXPRESS;Initial Catalog= Seminar;
Integrated Security=False; User ID=admin; Password=123;"

objConnection.Open()
Dim objCommand As New SqlCommand()
objCommand.Connection = objConnection
objCommand.CommandType = CommandType.Text
objCommand.CommandText = "SELECT CountryID, CountryName FROM Country ORDER BY
CountryName"

Dim objSDR As SqlDataReader = objCommand.ExecuteReader()
```

```
gvCountry.DataSource = objSDR
gvCountry.DataBind()        // gvCountry is the name of DataGridView
objConnection.Close()
```

DataAdapter

- DataAdapter provides the **communication between the Dataset and the DataSource**.
- The DataAdapter can perform Select, Insert, Update and Delete SQL operations in the Data Source.
- This is integral to the working of ADO.Net since data is transferred to and from a database through a data adapter.
- It retrieves data from a database into a dataset and updates the database.
- When changes are made to the dataset, the changes in the database are actually done by the data adapter.
- DataAdapter retrieves data into dataset or datatable from a data source using the Fill() method.

  Properties of DataAdapter Object
  - **DeleteCommand**: Represents a DELETE statement or stored procedure for deleting records from the data source.
  - **InsertCommand**: Represents an INSERT statement or stored procedure for inserting a new record to The data source.
  - **UpdateCommand**: Represents an UPDATE statement or stored procedure for Updating recording in a data source.
  - **TableMappings**: Represents a collection of mappings between actual data source table and a DataTable object.

  Methods of DataAdapter Object
  - **Fill()**: This method fills data records from a DataAdapter to a DataSet object.
  - **FillSchema()**: This method adds a DataTable to a DataSet.
  - **GetFillParameters()**: This method retrieves parameters that are used when a SELECT statement is executed.
  - **Update()**: This method stores data from a data set to the data source.
- Following code demonstrate code of disconnected objects

```
Imports System.Data
Imports System.Data.SqlClient
Namespace DataTable_Datarow_DataColumn_Example
    Class Program
        Private Shared Sub Main(ByVal args As String())

            'Step 1: Prepare Connection
            Dim objConnection As New SqlConnection()
            objConnection.ConnectionString = "Data
```

```
Source = ANSPC\SQLEXPRESS;Initial Catalog= Seminar; Integrated
Security=False; User ID=admin; Password=123;"
objConnection.Open()

'Step 2: Prepare & Execute Command
Dim objCommand As New SqlCommand()
objCommand.Connection = objConnection
objCommand.CommandType = CommandType.Text
objCommand.CommandText = "SELECT CountryID, CountryName FROM
Country  ORDER BY CountryName"

        Dim sda As New SqlDataAdapter(objCommand)
        Dim dt As New DataTable()
        sda.Fill(dt)
        objConnection.Close()
    End Sub
End Class
End Namespace
```

DataTable

- A DataTable is an in-memory representation of a single database table which has collection of rows and columns whereas a DataSet is an in-memory representation of a database-like structure which has collection of DataTables.
- DataTable class is a member of the **System.Data** namespace within .Net Framework class library.
- DataTable fetches only one **TableRow** at a time.
- In DataTable, there is no Unique Constraint and Foreign Key Constraint objects available.
- To add rows to a DataTable, you must first use the NewRow() method to return a new DataRow object.

Properties of DataTable Object

- **Columns**: Represents all table's columns.
- **DataSet**: Returns the dataset for the table.
- **DefaultView**: Customized view of the data table.
- **PrimaryKey**: Represents an array of columns that function as primary key for the table.
- **Rows**: All rows of the data table.
- **TableName**: Name of the table.

Methods of DataTable Object

- **Clear():** Deletes all data table data
- **Copy():** Copies a data table including its schema
- **NewRow():** Creates a new row, which is later added by calling the Rows.Add method.

- **AcceptChanges():** Commits all the changes made since last AcceptChanges was called.
- **RejectChanges():** Reject all changed made after last AcceptChanges was called.
- **GetErrors():** Returns an array hat contains the DataRow objects that contain errors.
- **Reset():** Reset your DataTable to its original, uninitialized state.
- **Select():** Returns an array of DataRow objects based on the specified search criteria.

DataColumn

- DataColumn designates columns in DataTables.
- A DataTable is represented by several types.
- It specifies the name and type of certain columns in the table. This is called a DataColumn.
- With the DataColumn class, we can add and loop over columns.

Properties of DataColumn Object

- **AllowDBNull**: Gets or sets a value that indicates whether null values are allowed in this column for rows that belong to the table.
- **AutoIncrement**: Gets or sets a value that indicates whether the column automatically increments the value of the column
- **ColumnName**: Gets or sets the name of the column in the DataColumnCollection.
- **DataType**: Gets or sets the type of data stored in the column.
- **MaxLength**: Gets or sets the maximum length of a text column.
- **ReadOnly**: Gets or sets a value that indicates whether the column allows for changes as soon as a row has been added to the table.
- **Table**: Gets the DataTable to which the column belongs to.

DataRow

- The DataRow object lets you examine and modify the content of a row in your DataTable.
- It represents a row in the DataTable.
- The DataRow object and its properties and methods are used to retrieve, evaluate, insert, delete, and update values in the DataTable.
- The **NewRow** method is used to create a new row and the Add method adds a row to the table.
- To assign a DataRow object to a particular row in the DataTable.

Properties of DataRow Object

- **HasErrors**: Gets a value that indicates whether there are errors in a row.
- **Item[Int32]:** Gets or sets the data stored in the column specified by index.
- **Item[String]:** Gets or sets the data stored in the column specified by name.
- **RowError:** Gets or sets the custom error description for a row.
- **Table**: Gets the DataTable for which this row has a schema.

Methods of DataRow Object

- **AcceptChanges()**: Commits all the changes made to this row since the last time AcceptChanges was called.
- **BeginEdit()**: Starts an edit operation on a DataRow object.
- **CancelEdit()**: Cancels the current edit on the row.
- **Delete()**: Deletes the DataRow.
- **ToString()**: Returns a string that represents the current object.

Create a Database Application

- First of all in SQL Server create one database. So for that click on the SQL Server→connect→Right click on Database→New Database→Enter Database name "DIETDS"→ In that hierarchy Tables Right click on table→Add new table→Add name of columns with its specified datatype→Now press CTRL + S → Enter the name of the table
- Now database connection can be created with types:
  - **Bound Connection**
  - **Unbound Connection**

Bound Connection

- Bound Connection means in this connectivity we don't have to write any code, we have to use inbuilt functionality of the .NET framework.
- To perform a bound connection, do the following steps:
  - Open the Visual Studio→Click on new project→select language VB.NET and select windows forms application→Give proper title and proper path where you want to save this application.
  - On the form, Drag and drop **DataGridView** control and **BindingNavigator** control.
  - Click on the smart tag of DataGridView control→ then, Choose a data source→In it, Click on the **Add Project Data Source**→It will open one dialog box. After that, Select **Database** option from the opened dialog box and click **NEXT**→Select **Dataset** and click **NEXT**→Now Click on the **NEW CONNECTION** button.
  - It will one **Add Connection** dialog box. In the opened dialog box, Select a proper **data source** as per your requirement, Here, we are using **SQL Server** so click on the **CHANGE** button and select the Microsoft SQL Server option from opened dialog box and click **OK** button.
  - After that enter the name of the server or select name of server from the dropdown list.
  - If you don't kept any username and password for database then select **Windows Authentication** mode
  - If you want to keep username and password then select **SQL Server Authentication**.
  - Now select the proper database name from the dropdown of database name and click on TEST CONNECTION button. If test successful, then click on **OK** button.

- After that click **Next** button and Now select the proper table from the opened Data Source Configuration wizard. If we want to give a proper dataset name then change it, otherwise framework will give it itself. After that click on the **FINISH** button.
- Now select the binding navigator control and go to the properties of this control.
- Now select the **BindingSource** property and in that select the binding source which we have to connect with this DataGrid control.
- Save and run the application.

Unbound Connection

- Unbound Connection means in this connectivity we have to write code as per our requirement of the application and in industrial project this type of the connection usable.
- Open the Visual Studio→Click on new project→select language VB.NET and select windows forms application→Give proper title and proper path where you want to save this application.
- Design the form as shown in Fig 5.2 below:

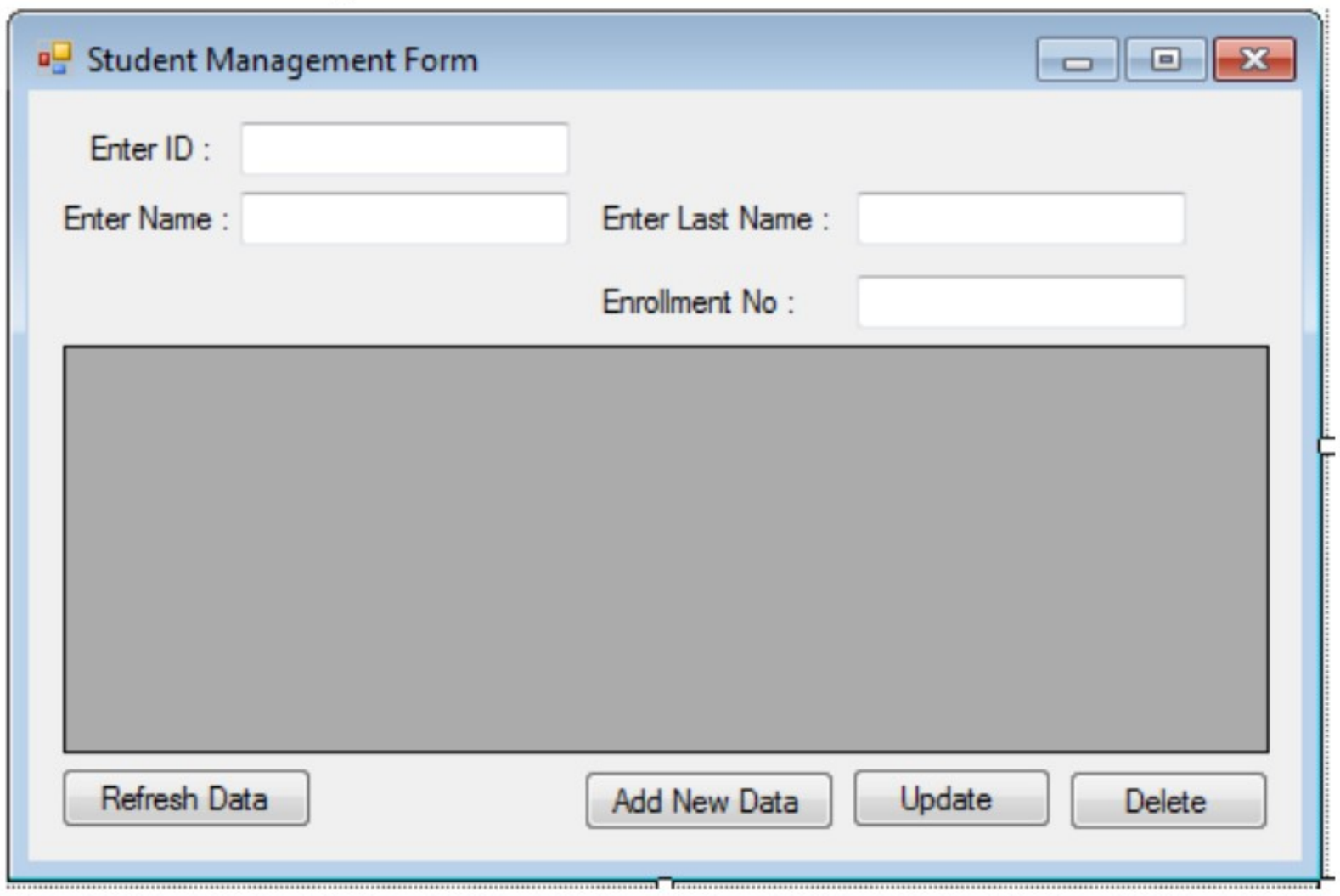

Fig 5.2 Form Design

- Now set text and name property of the control as given below:

Control	Text Property	Name Property
Label	Enter ID:	lblEnterID
	Enter Name:	lblName
	Enter Last Name:	lblLastName
	Enrollment No:	lblEnrollmentNo
Textbox		txtEnterID
		txtName
		txtLastName

		txtEnrollmentNo
DataGrid Control		dgStudentInfo
Button	Refresh Data	btnRefresh
	Add New Data	btnAdd
	Update	BtnUpdate
	Delete	btnDelete
Form	Student Management Form	frmStudentInfo

- Before Perform Unbound connection import following two namespaces in header:

```
Imports System.Data.Sql
Imports System.Data.SqlClient
```

- Write the following code on the **load event** of the form

```
Private Sub frmStudentInfo_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

        Dim con = New SqlConnection("Data Source=ANSPC\SQLEXPRESS;Initial
        Catalog=Seminar;Integrated Security=True")
         con.Open()

        Dim selectquery As String = "Select * from Student"
        Dim sda As SqlDataAdapter = New SqlDataAdapter(selectquery, con)
        Dim dt As DataTable = New DataTable()
        sda.Fill(dt)
        dgStudent.DataSource = dt
End Sub
```

- Write the following code on the click event of **Add Button.**

```
Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnAdd.Click
        If (txtName.Text = "") Then
                MsgBox("Enter the First Name")
        ElseIf (txtLastName.Text = "") Then
                MsgBox("Enter the Last Name")
        ElseIf (txtEnrollmentNo.Text = "") Then
                MsgBox("Enter the Enrollment Number")
        Else
                Dim con = New SqlConnection("Data Source=ANSPC\SQLEXPRESS;Initial
                Catalog=Seminar;Integrated Security=True")
```

```
        con.Open()

        Dim insertquery As String = "Insert into Student (Student_Name,
        Student_LastName, EnrollementNo) values ('" + txtName.Text + "','" +
        txtLastName.Text + "','" + txtEnrollmentNo.Text + "')"

        Dim cmd As SqlCommand = New SqlCommand(insertquery, con)
        cmd.ExecuteNonQuery()
        MsgBox("Inserted Successfully Data")
    End If
End Sub
```

- Write the following code on the click event of **Update Button.**

```
Private Sub btnUpdate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    If (txtEnterID.Text = "") Then
        MsgBox("Please Enter the ID First")
    Else
        Dim con = New SqlConnection("Data Source=ANSPC\SQLEXPRESS;Initial
        Catalog=Seminar;Integrated Security=True")
        con.Open()

        Dim updatequery As String = "update Student SET Student_Name='" +
        txtName.Text + "',Student_LastName='" + txtLastName.Text + "' ,
        EnrollementNo='" + txtEnrollmentNo.Text + "' where Student_ID='" +
        txtEnterID.Text + "' "

        Dim cmd As SqlCommand = New SqlCommand(updatequery, con)
        cmd.ExecuteNonQuery()
        MsgBox("Updated Successfully Data")
    End If
End Sub
```

- Write the following code on the click event of **Delete Button.**

```
Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
    If (txtEnterID.Text = "") Then
        MsgBox("Please Enter the ID Firstly")
    Else
```

```
        Dim con = New SqlConnection("Data Source=ANSPC\SQLEXPRESS;Initial
        Catalog=Seminar;Integrated Security=True")
        con.Open()

        Dim deletequery As String = "delete from Student where Student_ID='" +
        txtEnterID.Text + "' "

        Dim cmd As SqlCommand = New SqlCommand(deletequery, con)
        cmd.ExecuteNonQuery()
        MsgBox("Deleted Successfully Data")
    End If
    End Sub
```

- Write the following code on the click event of **Refresh Data Button.**

```
Private Sub btnRefresh_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnRefresh.Click
        Dim con = New SqlConnection("Data Source=ANSPC\SQLEXPRESS;Initial
        Catalog=Seminar;Integrated Security=True")
        con.Open()

        Dim selectquery As String = "Select * from Student"

        Dim sda As SqlDataAdapter = New SqlDataAdapter(selectquery, con)
        Dim dt As DataTable = New DataTable()
        sda.Fill(dt)
        dgStudentInfo.DataSource = dt
End Sub
```

- By writing the above code, It will open one form in which can perform insert, update and delete operation using unbound connection.