UNIT – 4

Database Programming with ADO.NET

Programming with C#

Code : CS-23

Introduction of ADO.net :

- ADO stands for ActiveX Data Objects. OR Microsoft ActiveX Data Objects.

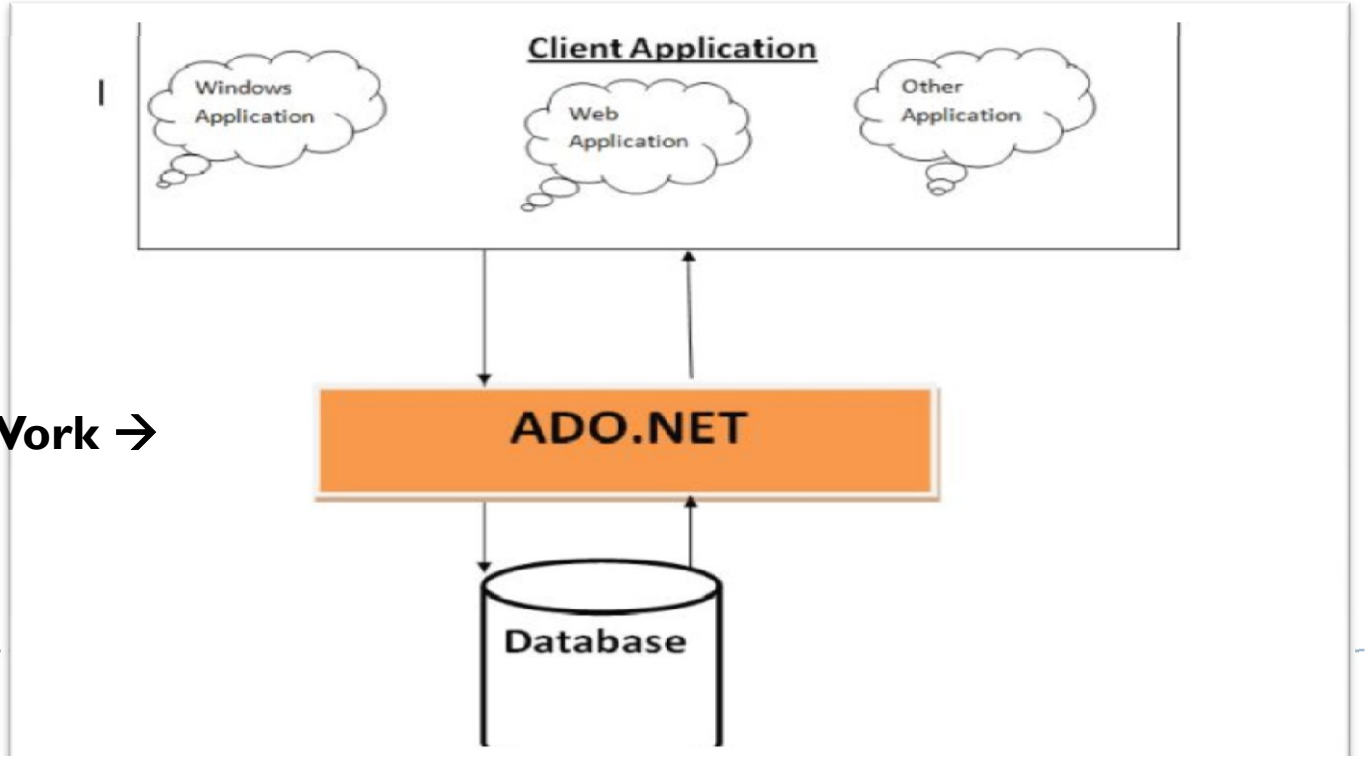
- ADO.NET is a module of .Net Framework which is used to establish connection between application and data sources.
  Or we can also say that :
  ADO.NET provides a bridge between the front end controls and the back end database.

- Data sources can be such as SQL Server and XML.
- ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.
- ADO.NET is a set of classes that expose data access services for .NET Framework programmers.
- ADO.NET provides a rich set of components for creating distributed, data-sharing applications.
- It is an integral part of the .NET Framework, providing access to relational, XML, and application data.

- **We need to add System.Data namespace for work with ADO.NET :**

Namespaces	Description
System.Data	Contains the definition for Columns ,relations, tables, database, rows , views and constraints.
System.Data.SqlClient	Contains the classes to connect to a Microsoft SQL Server database such as SqlCommand, SqlConnection, and SqlDataAdapter.
System.Data.Odbc	Contains classes required to connect to most ODBC drivers. These classes include OdbcCommand and OdbcConnection.
System.Data.OracleClient	Contains classes such as OracleConnection and OracleCommand required to connect to an Oracle database.

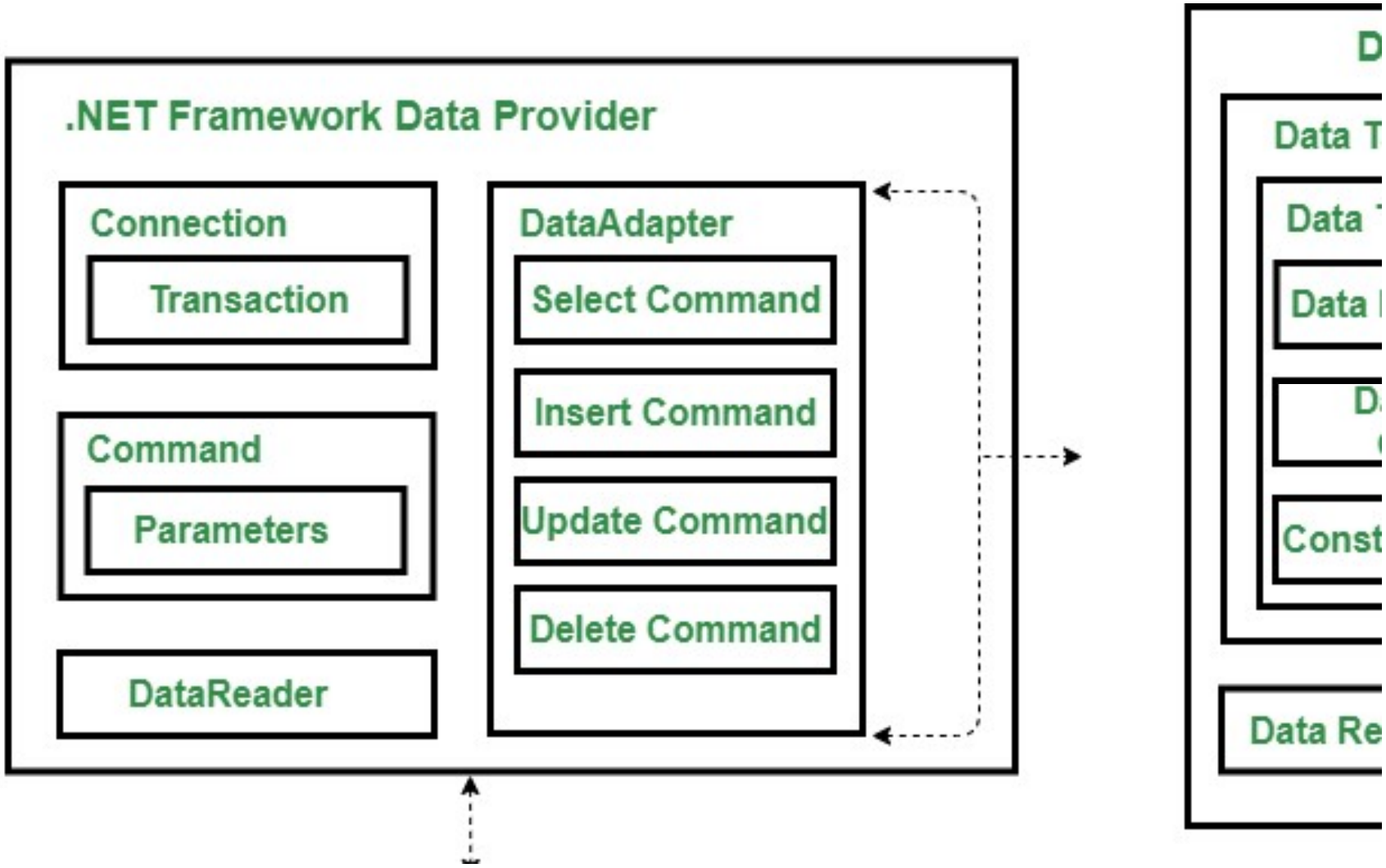
How ADO.net Work →



ADO.NET has several advantages :

- You can use ADO.NET to connect with any type of database.
- ADO.NET supports Connected as well as Disconnected Architecture. Disconnected Architecture is a new invention since ADO.NET is introduced.
- XML is supported by ADO.NET which is widely used for data transformation widely today. This makes your data transformation much more faster.
- You can create Fast, Scalable and Secured applications using ADO.NET
- Cross Language support is provided by ADO.NET as Multilanguage is a prime feature of .NET framework.

Architecture of ADO.net :



- The two main components of ADO.NET for accessing and manipulating data are data provider and dataset.

- **DataSet** represent either an entire database or a subset of database. It can contain tables and relationships between those tables.

- **Data provider** is a collection of components like connection, command, DataReader, DataAdapter objects and handles communication with physical data store and the dataset.

- **There are Two Types of ADO.net Architecture :**

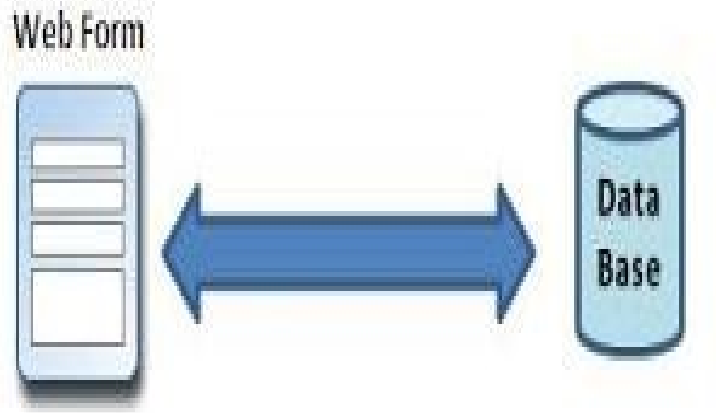
1. Connected Architecture
2. Disconnected Architecture

Connected Architecture :

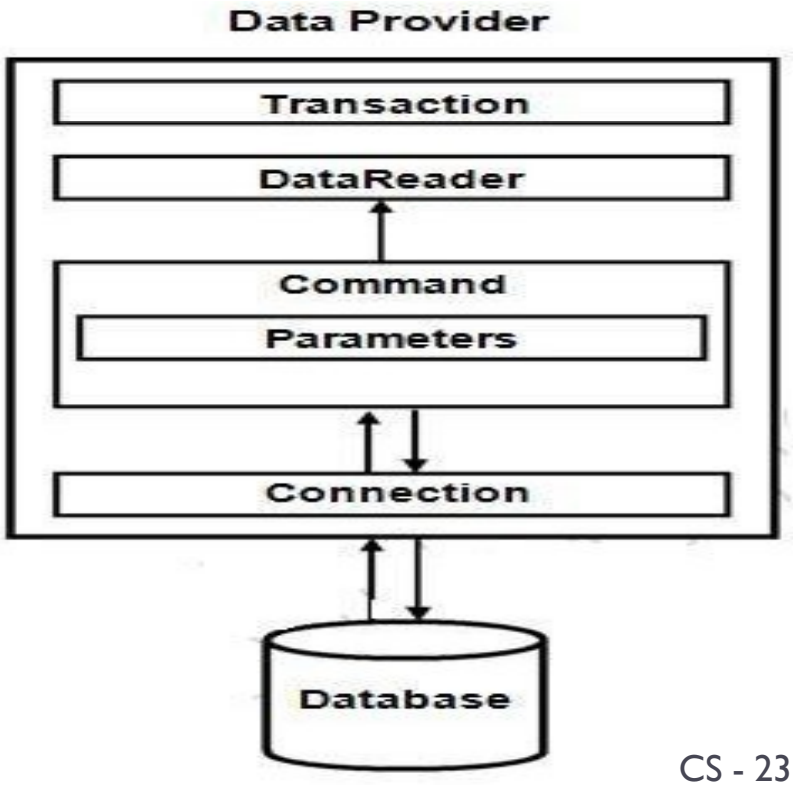
- The architecture of ADO.net, in which connection must be opened to access the data retrieved from database is called as connected architecture.
- We typically employ the DataReader class object for connected architecture.
- In addition to ensuring that the connection is maintained for the entire period of time, DataReader is used to retrieve data from databases.
- Connected Oriented Architecture gives faster performance when dealing with smaller applications, with Select SQL queries and smaller data.
- We can access the data in a **forward-only** and **read-only** manner.
- In connected architecture, the application is **directly linked** to the Database.
- Using **DataReader** we cannot persist the data in the database.

- Classes that have been used in Connected Architecture include :

1. Connection
2. Command
3. DataReader
4. Transaction
5. Parameter



Connected Architecture



- In the connected environment database is not store in a client PC. It means the back up is not on the client PC.

- A data is store directly on server and user wants show a data at a time always we want to connect to the database.

- In connected architecture you have to declare the connection explicitly by using Open(), and close the connection using Close(), and you can execute commands using different methods like.. ExecuteNonQuery, ExecuteScalar, ExecuteReader etc.

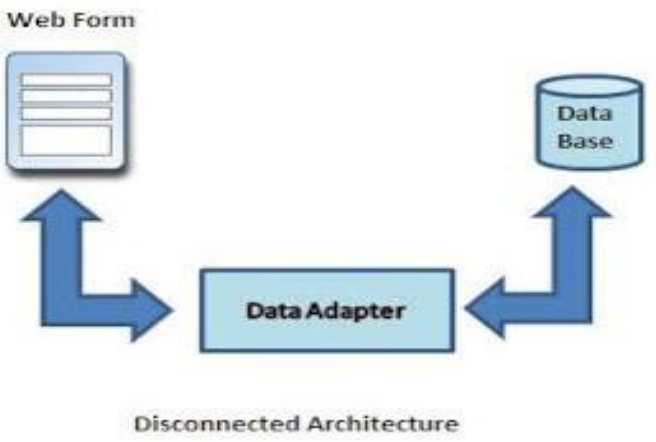
Disconnected Architecture :

- The architecture of ADO.net in which data retrieved from database can be accessed even when connection to database was closed is called as disconnected architecture.
- It is disconnection-oriented data access architecture. It means always an active and open connection is not required.
- Disconnected architecture in Ado.net refers to a style of architecture in which continuous connectivity between the database and application is not upheld.
- In this mode, connectivity is only established to read data from the database and then to update data already present in the database.
- Using DataAdapter and DataSet or DataTable we can implement Dis-Connected Oriented Architecture.
- .NET runtime creates an instance of the DataTable to holds data.
- This indicates that data is fetched from the database and stored in temporary tables because data is required during the processing of the application.

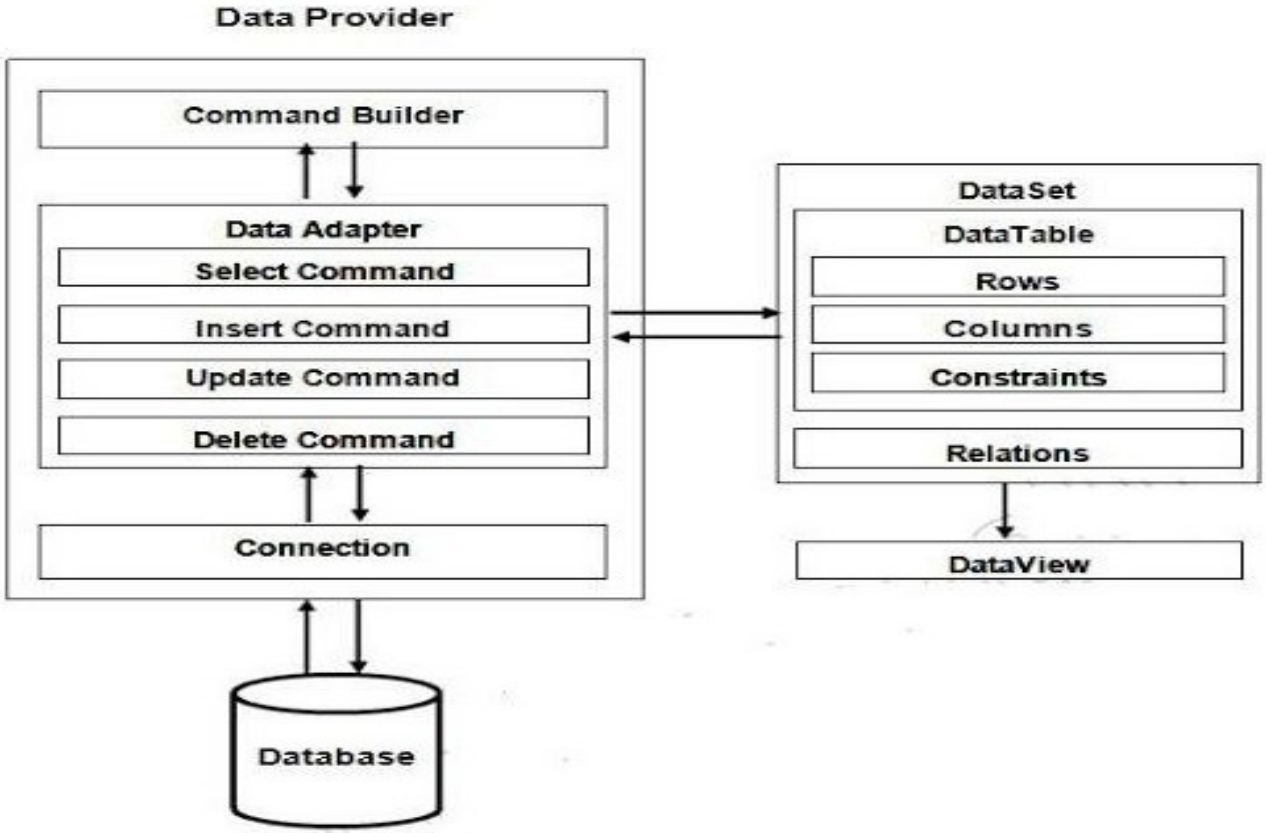
- Data is then fetched from the temporary tables whenever it is needed.
- Finally, after the operations were finished, a connection was made in order to update the database's data using the temporary tables.
- Disconnected Oriented Architecture gives the best performance when dealing with multi-layered applications and loads of data.
- Disconnected Oriented Architecture can hold the data of multiple tables using dataset and single table data using Datatable.
- You can access the data in **forward and backward directions** and you can also **modify the data**.
- Using Data Adapter, we can persist the DataSet or DataTabledata into the database.
- The DataAdapter is an abstraction of the connected classes that simplifies filling the disconnected DataSet or DataTable classes with data from the data source and updating the data source to reflect any changes made to the disconnected data.

- Classes that have been used in Disconnected Architecture include :

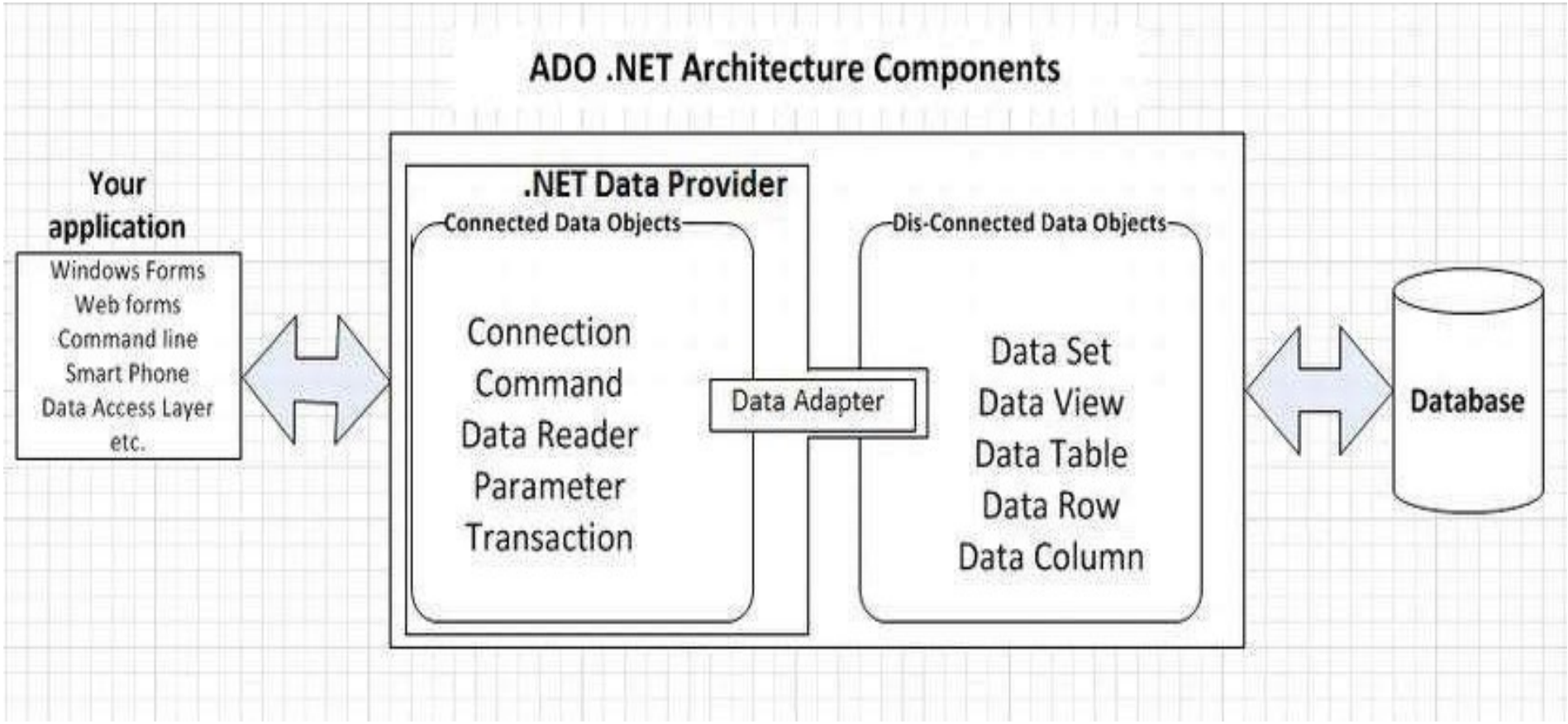
1. DataAdapter
2. DataSet
3. DataTable
4. DataColumn
5. DataRow
6. DataRelationship
7. DataView



Disconnected Architecture



- In disconnected architecture you don’t need to define the connections explicitly.
- **DataAdatpter** itself can open and closes the connection and you can use dataset for storing the info.
- Temporarily and Fill method was used to execute the commands give in adapter.



Points	Connected Architecture	Disconnected Architecture
Connection	The application will establish a connection with database server and it will be kept open until unless the task has been completed.	The required data will be fetched into the dataset and then the connection will be closed.
Traffic	For each and every operation we need to communicate with database server, so that traffic to the database server will be increased.	The required data manipulations will be done on the tables into the dataset and the changes will be updated back to the database at once, so that traffic to the database sever can be reduced.
Transaction	Parallel transactions will be easy.	Parallel transactions will be difficult.
Speed	Fetching the data will be fast.	Fetching the data will be slow.
Memory	No memory management is required.	Memory management is required for dataset.
data read	DataReader for read the data.	Dataset for read the data.
Way of data access	We can access the data in a forward-only	You can access the data in forward and backward directions
Type of Data	Data is read only data	You can modify the data.

Connection Object :

- The Connection object is the first component of ADO.NET that you should be looking at. A connection sets a link between a data source and ADO.NET.
- A Connection object sits between a data source and a DataAdapter (via Command).
- You need to define a data provider and a data source when you create a connection.
- With these two, you can also specify the user ID and password depending on the type of data source.

- **Data provider connection classes :**

DATA PROVIDER	CONNECTION CLASS
OleDb	OleDbConnection
Sql	SqlConnection
ODBC	OdbcConnection

- **Here , We are using SqlClient :**

- First add this two namespaces :

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

- Now create a object of SqlConnection and set path of database :

```
SqlConnection  sql=new SqlConnection(@"Data Source=
(LocalDB)\v11.0;AttachDbFilename=E:\sybca\database_folder\
database_folder\Database1.mdf;Integrated Security=True");
```

- Now open a connection :

```
sql.Open();
```

- After completion on our work then close the connection

```
sql.Close();
```

- **Connected Architecture classes or Objects :**
  1. **Command**
  2. **DataReader**

Command Object:

- The Command Object uses the connection object to execute SQL queries.
- The queries can be in the Form of Inline text, Stored Procedures or direct Table access.
- An important feature of Command object is that it can be used to execute queries and Stored Procedures with Parameters.
- If a select query is issued, the result set it returns is usually stored in either a DataSet or a DataReader object.
- Now let's see some constructors of command object :

Constructor	Description
SqlCommand()	It is used to initialize a new instance of the SqlCommand class.
SqlCommand(String)	It is used to initialize a new instance of the SqlCommand class with a string parameter.
SqlCommand(String, SqlConnection)	It is used to initialize a new instance of the SqlCommand class. It takes two parameters, first is query string and second is connection string.
SqlCommand(String, SqlConnection, SqlTransaction)	It is used to initialize a new instance of the SqlCommand class. It takes three parameters query, connection and transaction string respectively.
SqlCommand(String,SqlConnection,SqlTransaction,SqlCommand ColumnEncryptionSetting)	It Initializes a new instance of the SqlCommand class with specified command text, connection, transaction, and encryption setting.

SqlCommand() Because of we are using here SqlClient

Example of SqlCommand Object:

```
using System;
using System.Data;
using System.Data.SqlClient;
namespace database
{
    public partial class Form1 : Form
    {
            SqlConnection con = new SqlConnection(@"Data
            Source=(LocalDB)\v11.0;AttachDbFilename=E:\sybca\unit_5_demo\unit_5_demo\
            db_file.mdf;Integrated Security=True");
          private void button1_Click(object sender, EventArgs e)
          {
                  con.Open();
                  SqlCommand cmd=new SqlCommand("insert into demo_tbl values
                                                        (2,'sybca')",con);
                   cmd.ExecuteNonQuery();
                   con.Close();
            }
    }
}
```

- The Command class provides methods for storing and executing SQL statements and Stored Procedures. The following are the various commands that are executed by the Command Class.
- **ExecuteReader:** Returns data to the client as rows. This would typically be an SQL select statement or a Stored Procedure that contains one or more select statements. This method returns a DataReader object that can be used to fill a DataTable object or used directly for printing reports and so forth.

- **ExecuteNonQuery:** Executes a command that changes the data in the database, such as an update, delete, or insert statement, or a Stored Procedure that contains one or more of these statements. This method returns an integer that is the number of rows affected by the query.

- **ExecuteScalar:** This method only returns a single value. This kind of query returns a count of rows or a calculated value.

- **ExecuteXMLReader:** (SqlClient classes only) Obtains data from an SQL Server 2000 database using an XML stream. Returns an XML Reader object.

DataReader :

- The DataReader object is an alternative to the DataSet and DataAdapter combination.
- This object provides a connection oriented access to the data records in the database.
- The ADO.NET SqlDataReader class in C# is used to read data from the SQL Server database in the most efficient manner.
- **SqlDataReader is Connection-Oriented.** It means it requires an open or active connection to the data source while reading the data. The data is available as long as the connection with the database exists.
- **SqlDataReader is Read-Only.** It means it is also not possible to change the data using SqlDataReader. You also need to open and close the connection explicitly.
- **Forward-Only:** The SqlDataReader works forwardly, meaning it can only read data in one direction – from the first to the last.

ADO.NET SqlDataReader Class Properties in C#:

Property	Description
Connection	It is used to get the SqlConnection associated with the SqlDataReader.
Depth	It is used to get a value that indicates the depth of nesting for the current row.
FieldCount	It is used to get the number of columns in the current row.
HasRows	It is used to get a value that indicates whether the SqlDataReader contains one or more rows.
IsClosed	It is used to retrieve a boolean value that indicates whether the specified SqlDataReader instance has been closed.
Item[String]	It is used to get the value of the specified column in its native format given the column name.
Item[Int32]	It is used to get the value of the specified column in its native format given the column ordinal.
RecordsAffected	It is used to get the number of rows changed, inserted or deleted by execution of the Transact-SQL statement.
VisibleFieldCount	It is used to get the number of fields in the SqlDataReader that are not hidden.

ADO.NET SqlDataReader Class Methods in C#:

Method	Description
Close()	It is used to closes the SqlDataReader object.
GetName(Int32)	It is used to get the name of the specified column.
GetSchemaTable()	It is used to get a DataTable that describes the column metadata of the SqlDataReader.
GetValue(Int32)	It is used to get the value of the specified column in its native format.
GetValues(Object[])	It is used to populate an array of objects with the column values of the current row.
NextResult()	It is used to get the next result, when reading the results of SQL statements.
Read()	It is used to read record from the SQL Server database.

Example :

```
SqlConnection con = new SqlConnection(@"Data
    Source=(LocalDB)\v11.0;AttachDbFilename=E:\sybca\unit_5_demo\unit_5_demo\db_file.mdf;
    Integrated Security=True");


private void show_data_Click(object sender, EventArgs e)
{
        con.Open();
        SqlCommand cmd = new SqlCommand("select *from demo_tbl",con);
        SqlDataReader rd = cmd.ExecuteReader();
        if (rd.HasRows)
        {
            DataTable dt = new DataTable();
            dt.Load(rd);
            dataGridView1.DataSource = dt;
        }
        else
        {
            MessageBox.Show("Data Not Found");
        }
}
```

Disconnected Architecture classes or Objects :

1. DataAdapter
2. DataSet
3. DataTable
4. DataRow
5. DataColumn
6. DataRelation
7. DataView

DataAdapter :

- The DataAdapter works as a bridge between a DataSet and a data source to retrieve data.
- DataAdapter is a class that represents a set of SQL commands and a database connection.
- It can be used to fill the DataSet and update the data source.

Constructors	Description
DataAdapter()	It is used to initialize a new instance of a DataAdapter class.
DataAdapter(DataAdapter)	It is used to initializes a new instance of a DataAdapter class from an existing object of the same type.

Method	Description
CloneInternals()	It is used to create a copy of this instance of DataAdapter.
Dispose(Boolean)	It is used to release the unmanaged resources used by the DataAdapter.
Fill(DataSet)	It is used to add rows in the DataSet to match those in the data source.
FillSchema(DataSet, SchemaType,String, IDataReader)	It is used to add a DataTable to the specified DataSet.
GetFillParameters()	It is used with SQL SELECT Command.
ResetFillLoadOption()	It is used to reset FillLoadOption to its default state.
ShouldSerializeAcceptChangesDuringFill()	It determines whether the AcceptChangesDuringFill property should be persisted or not.
ShouldSerializeFillLoadOption()	It determines whether the FillLoadOption property should be persisted or not.
ShouldSerializeTableMappings()	It determines whether one or more DataTableMapping objects exist or not.
Update(DataSet)	It is used to call the respective INSERT, UPDATE, or DELETE statements.

Example :

```
SqlConnection con = new SqlConnection(@"Data
   Source=(LocalDB)\v11.0;AttachDbFilename=E:\sybca\unit_5_de
   mo\unit_5_demo\db_file.mdf;Integrated Security=True");


private void show_data_Click(object sender, EventArgs e)
{
        SqlDataAdapter s_adapter = new
   SqlDataAdapter("select *from demo_tbl",con);
          DataSet ds = new DataSet();
          s_adapter.Fill(ds);
          dataGridView1.DataSource = ds.Tables[0];
}
```

- You can also pass value in command like this :
- **SqlDataAdapter** adptr = new SqlDataAdapter("select *from demo_tbl;select * from second_tbl", con);

- When you want to access data of table 1 then write data_set.Table[0]
- dataGridView1.DataSource = ds.Tables[0];

- When you want to access data of table 2 then write data_set.Table[1]
- dataGridView1.DataSource = ds.Tables[1];

DataSet :

- The DataSet represents a subset of the database in memory.
- DataSet is tabular representation of data.
- Tabular representation means it represents data into row and column format.
- This class is counted in a disconnected architecture in .NET Framework.
- This is the reason why it is used to fetch the data without interacting with any data source.
- Which means it is found in the "**System.Data**" namespace.
- The Dataset can hold records of more than one Database tables or DataTables (Work with multiple table at a time with its relatonship).
- The dataset is for all kinds of data providers,like sql, odbc, oledb, oracle.

Follow Example of Slide number 29

DataTable :

- In the ADO.NET library, C# DataTable is a central object.
- It represents the database tables that provide a collection of rows and columns in grid form.
- DataTable represents relational data into tabular form. ADO.NET provides a DataTable class to create and use data table independently. It can also be used with DataSet also. Initially, when we create DataTable, it does not have table schema. We can create table schema by adding columns and constraints to the table. After defining table schema, we can add rows to the table.
- We must include **System.Data** namespace before creating DataTable.
- DataTable represents a single table.

Constructors	Description
DataTable()	It is used to initialize a new instance of the DataTable class with no arguments.
DataTable(String)	It is used to initialize a new instance of the DataTable class with the specified table name.
DataTable(SerializationInfo, StreamingContext)	It is used to initialize a new instance of the DataTable class with the SerializationInfo and the StreamingContext.
DataTable(String, String)	It is used to initialize a new instance of the DataTable class using the specified table name and namespace.

METHOD	DESCRIPTION
AcceptChanges	Commits all the changes made since last AcceptChanges was called
Clear	Deletes all data table data
Clone	Creates a clone of a DataTable including its schema
Copy	Copies a data table including its schema
NewRow	Creates a new row, which is later added by calling the Rows.Add method
RejectChanges	Reject all changed made after last AcceptChanges was called
Reset	Resets a data table's original state
Select	Gets an array of rows based on the criteria

Property	Description
Columns	It is used to get the collection of columns that belong to this table.
Constraints	It is used to get the collection of constraints maintained by this table.
DataSet	It is used to get the DataSet to which this table belongs.
DefaultView	It is used to get a customized view of the table that may include a filtered view.
HasErrors	It is used to get a value indicating whether there are errors in any of the rows in the table of the DataSet.
MinimumCapacity	It is used to get or set the initial starting size for this table.
PrimaryKey	It is used to get or set an array of columns that function as primary keys for the data table.
Rows	It is used to get the collection of rows that belong to this table.
TableName	It is used to get or set the name of the DataTable.

Follow Example of Slide number 25

DataColumn :

- The DataColumnCollection type returns a collection of columns that can be accessed through the Columns property of the DataTable.
- The DataColumnCollection object represents a collection of columns attached to a data table.
- You add a data column to the DataColumnCollection using its Add method.
- The DataColumn object represents a column of a DataTable.
- For example, say you want to create a customer table that consists of three columns: ID, Address, and Name. You create three DataColumn objects and these columns to the DataColumnCollection using the DataTable.Column.Add method.

PROPERTY	DESCRIPTION
AllowDBNull	Both read and write, represent if the column can store null values or not
AutoIncrement	Represent if the column's value is auto increment or not
AutoIncrementSeed	Starting value of auto increment, applicable when AutoIncrement is true
AutoIncrementStep	Indicates the increment value
Caption	Caption of the column
ColumnMapping	Represent the MappingType of the column
ColumnName	Name of the column
DataType	Data type stored by the column
DefaultValue	Default value of the column
Expression	Represents the expression used to filter rows, calculate values, and so on
MaxLength	Represents maximum length of a text column
ReadOnly	Represents if a column is read-only or not
Unique	Indicates whether the values in a column must be unique or not

DataRow :

- A DataRow represent a row of data in data table.
- You add data to the data table using DataRow object.
- A DataRowCollection object represents a collection of data rows of a data table.
- You use DataTable's NewRow method to return a DataRow object of data table, add values to the data row and add a row to the data Table again by using DataRowCollection's Add method.

- For Properties and methods :
- https://www.c-sharpcorner.com/uploadfile/mahesh/datarow-in-ado-net/

- **Example of DataTable, DataRow, DataColumn**

```
DataTable d_tbl = new DataTable("student");
DataColumn id = new DataColumn("id");
id.DataType = typeof(int);
id.AllowDBNull = false;
d_tbl.Columns.Add(id);

DataColumn nm = new DataColumn("nm");
id.Caption = "s_id";
nm.DataType = typeof(string);
nm.AllowDBNull = false;
nm.MaxLength = 10;
d_tbl.Columns.Add(nm);

d_tbl.Rows.Add(1, "ab");

DataRow dr = d_tbl.NewRow();

dr["id"] = 2;
dr["nm"] = "xyz";

d_tbl.Rows.Add(dr);
dataGridView1.DataSource = d_tbl;
```

DataRelation :

- To provide data integrity and consistency, you should use relationships between two tables.
- You achieve this relationship by defining a primary key in one table and using a foreign key in the other table.
- Say a customer has multiple orders; the Customers table stores the customer details, and the Orders table stores all the order details.
- To avoid the redundancy of data, you define the primary key of the Customers table as a foreign key in the Orders table.
- **Note:** In general this relationship is called the customer/order relationship parent/child, or sometimes master/ details.

Example :

```
DataRelation dtRelation;
DataColumn CustCol = dtSet.Tables["Customers"].Columns["id"];
DataColumn orderCol = dtSet.Tables["Orders"].Columns["CustId"];
dtRelation = new DataRelation("CustOrderRelation", CustCol, orderCol);
dtSet.Tables["Orders"].ParentRelations.Add(dtRelation);
dataGrid1.SetDataBinding(dtSet, "Customers");
```

DataView :

- A DataView provides various views of the data stored in a DataTable.
- Using a DataView, you can expose the data in a table with different sort orders, and you can filter the data by row state or based on a filter expression.
- That is we can customize the views of data from a DataTable.
- **Example :**

```
SqlDataAdapter dt = new SqlDataAdapter("select * from UserDetail",
    con);
DataSet ds = new DataSet();
dt.Fill(ds, "UserDetail");
con.Close();
DataView dv = new DataView();
GridView1.DataSource = ds.Tables[0].DefaultView;
GridView1.DataBind();
```

DataBinding :

- Simple data binding involves attaching any collection (item collection) which implements the IEnumerable interface, or the DataSet and DataTable classes to the DataSource property of the control.
- On the other hand, some controls can bind records, lists, or columns of data into their structure through a DataSource control. These controls derive from the BaseDataBoundControl class. This is called **declarative data binding**.
- The data source controls help the data-bound controls implement functionalities such as, sorting, paging, and editing data collections.

- **For More :**
- https://www.c-sharpcorner.com/article/data-binding-in-net/

GridView :

- **GridView** is a powerful and versatile data presentation control in C# that allows developers to display data in a *tabular format with sorting, paging, and editing capabilities.*
- Data from a database or other information sources can be shown using this method frequently in online applications.
- The namespace ***Web.UI.WebControls*** is frequently used in web applications.
- GridView can be bound to various data sources like databases, XML, and other data sources.
- **For More Detail :**
- https://www.javatpoint.com/gridview-c-sharp

- ADO : ActiveX Data Objects
- .net : Network Enable Technology
- OLEDB : Object Linking and Embedding, Database
- ODBC : Open Database Connectivity
- XML : Extensible Markup Language
- **Fore Difference between difference between**

  ExecuteNonQuery() and ExecuteScalar() and ExecuteReader()
- https://aravindchakkarapani.medium.com/difference-between-executenonquery-vs-executereader-vs-executescalar-5a53e19f2184#:~:text=2)%20ExecuteNonQuery%20%3A%20Used%20for%20executing,aggregate%20function%20or%20a%20subquery.

  DataReader & DataSet
- https://reintech.io/blog/ado-net-dataset-vs-datareader
- DataSet & DataTable
- https://www.educba.com/dataset-vs-datatable/

- **Link For ONE MARK Question of ADO.net definitions :**
- https://www.educative.io/answers/what-is-adonetrt
- *[2 mark]*
- Explian Command Object.
- Short Note on DataGridView.
- Explain Data Table
- Explain DataRow
- Explain DataColumn
- Explain ExecuteNonQuery()
- Explain ExecuteScalar()
- Explain ExecuteReader()
- *[3 mark]*
- Explain DataReader
- Difference between ExecuteNonQuery() & ExecuteNonQuery()
- Difference between Connected & Dis-connected architecture.
- Difference between DataSet & DataTable
- Difference between DataReader & DataSet
- Explain Connected Architecture.
- *[5 mark]*
- Explain architecture of ADO.net
- Explain Connection and DataAdapter objects in detail

DataReader	DataSet / DataAdapter
Database Connecting Mode: Connected mode	**Database Connecting Mo** Disconnected mode
Data Navigation: Unidirectional i.e., Forward Only	**Data Navigation:** Bidirectional i.e., data can n and forth
Read / Write: Only Read operation can be carried out.	**Read / Write:** Both Read and Write operat possible
Data Handling: Handles Database table	**Data Handling:** Handles text file, XML file ar table
Storage Capacity : No storage	**Storage Capacity :** Temporary Storage Capacity memory cache of data

Difference between DataSet and DataTable

	DataSet	DataTable
1	DataSet can fetch multiple TableRows at a time.	DataTable fetches onl TableRow at a time.
2	A DataSet is like structure which has collection of DataTables.	A DataTable is a sing table which has collec and columns.
3	DataSet, DataTable objects can be related to each other with DataRelation objects.	DataTable is a single table, so there is no D object in it.
4	DataSet is serialized DataSource .	DataTable, DataSourc serialized.
5	DataSet, data integrity is enforced by using the Unique Constraint and ForeignKeyConstraint objects.	DataTable, there is n Constraint and Foreig objects available.

Using DataSets vs. DataReaders

DataSet	DataReader
Read/write access to data	Read-only
Includes multiple tables from different databases	Based on one SQL statement from one database
Disconnected	Connected
Bind to multiple controls	Bind to one control only
Forward and backward scanning of data	Forward-only
Slower access	Faster access
Supported by Visual Studio .NET tools	Manually coded

1) What are the differences between DataReader and DataAdapter?

S.No	DataReader	DataAdapter
1	Works in Connected Mode	Works in Disconnected Mod
2	Can have only one record at a time	Can have more than 1 reco
3	Is ForwardOnly and Readonly	Can navigate front and bac editable
4	Faster	Slower

2) What are the differences between DataSet and DataReade

S.No	DataSet	DataReader
1	Works in Disconnected Mode	Works in Connected Mode
2	Can navigate back and forth	Can navigate forward only
3	Data is editable	Data is Readonly
4	Can contain more than one table and relationships	Can contain only one row a time
5	Slower as having more overhead	Faster when compared with DataSet

3) What is the difference between DataSet.Copy() and DataSet.Clone()?

S.No	DataSet.Copy()	DataSet.Clone()
1	DataSet.Copy() copies both the structure and data	DataSet.Clone() copies the structure of the DataSet, including all DataTable sch relations, and constraints a does not copy any data

4) What are the differences between RecordSet and DataSet?

S.No	RecordSet	DataSet
1	RecordSet provides data of	DataSet is a data structure

onnected	Disconnected
It is connection oriented.	It is dis_connection oriented.
Datareader	DataSet
Connected methods gives faster performance	Disconnected get low in speed
connected can hold the data of single table	disconnected can hold multipl
connected you need to use a read only forward only data reader	disconnected you cannot

SqlCommand Methods

- **1) ExecuteReader -Querying Data**
- 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", conn);

// 2. Call Execute reader to get query results
SqlDataReader rdr = cmd.ExecuteReader();
```

- **2) ExecuteNonQuery (Insert/Update/Delete)**
- To insert data into a database, use the ExecuteNonQuery method of the SqlCommand object.

```
// 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();
```

- **Similarly ExecuteNonQuery can be used to update a record using below query string**

```
// prepare command string
 string updateString = @"
    update Categories
    set CategoryName = 'Other'
    where CategoryName = 'Miscellaneous'";
```

- **Similarly ExecuteNonQuery can be used to delete a record using below query string**

```
// prepare command string
 string deleteString = @"
    delete from Categories
    where CategoryName = 'Other'";
```

3) ExecuteScalar - Getting Single values

- Used 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", conn);

 // 2. Call ExecuteNonQuery 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.