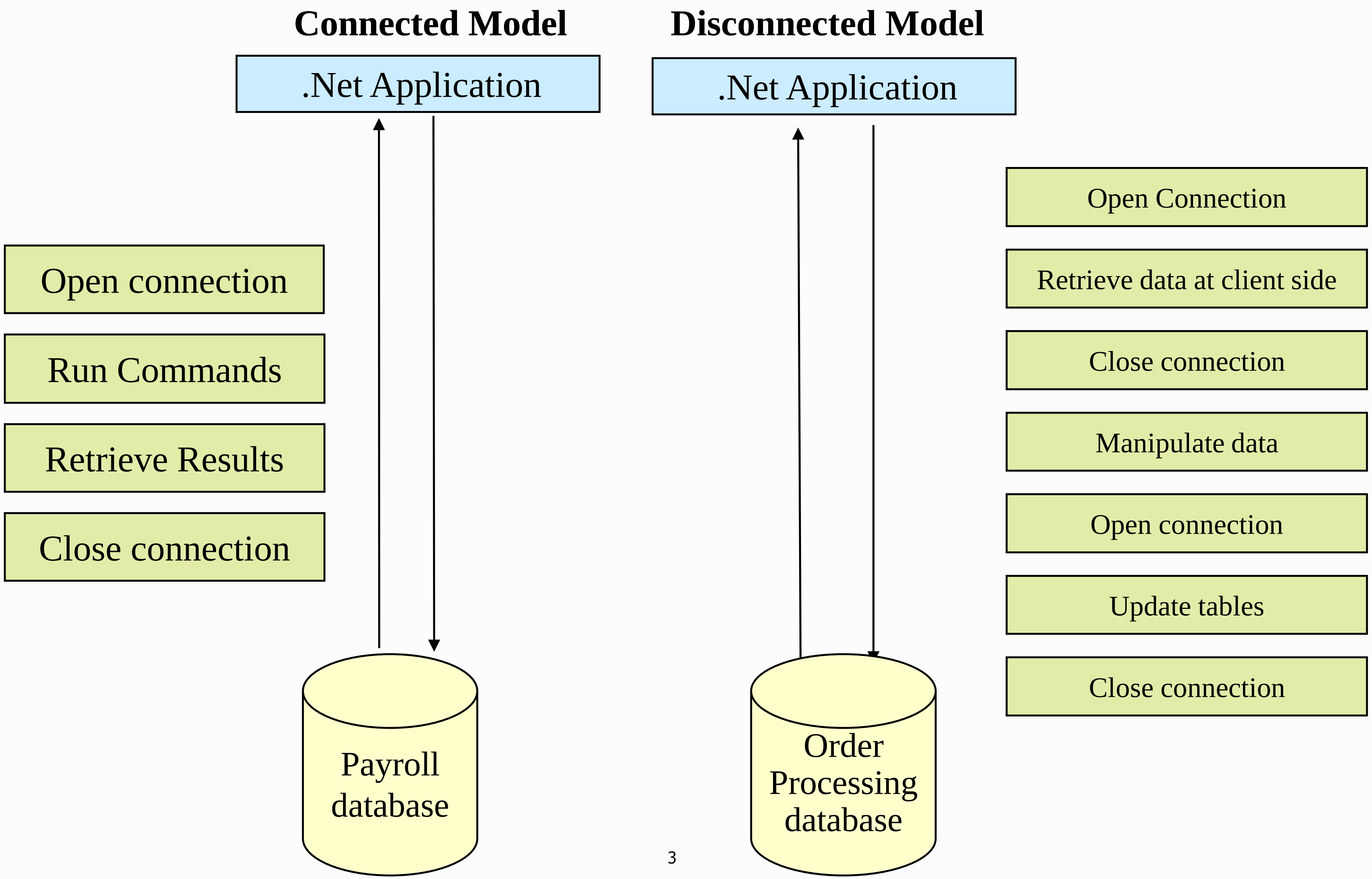
ADO.NET - Disconnected

Objectives

- On completion of this session you will be able to
  - Define disconnected architecture
  - List the objects required to achieve disconnected scenario.
  - Use DataAdapter object to fetch the data at client side.
  - Use dataset to store data at client side.
  - Navigate through the records using BindingContext.
  - Create a master detail application using DataRelation class.
  - List the XML classes and methods supported by DataSet object

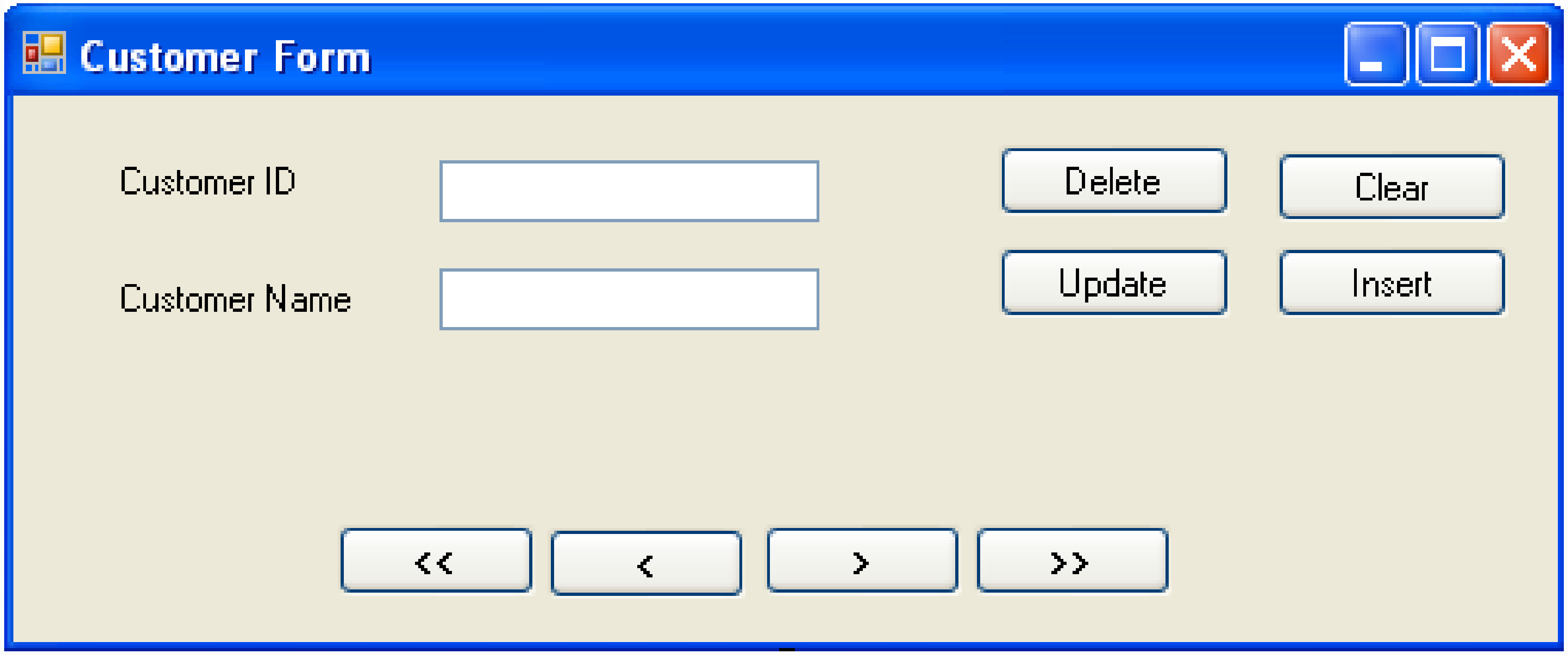
What is Disconnected Architecture?



Objects supporting the Disconnected Model

- `DataAdapter`
  - Represents a set of data commands & a database connection that are used to fill the dataset and update a SQL Server database.
- `DataSet`
  - In-memory representation of data.
- `CommandBuilder`
  - Automatically generates single-table commands that are used to make changes made to a Dataset with the associated SQL Server database.

User Interface



`DataAdapter` Object

- Forms a bridge between a disconnected ADO.NET objects and a data source
- Supports methods
  - `Fill()`
  - `Update()`

```
string SqlStr = "SELECT * FROM Orders";
SqlDataAdapter da = new SqlDataAdapter(SqlStr,con);
```

`DataAdapter` Properties

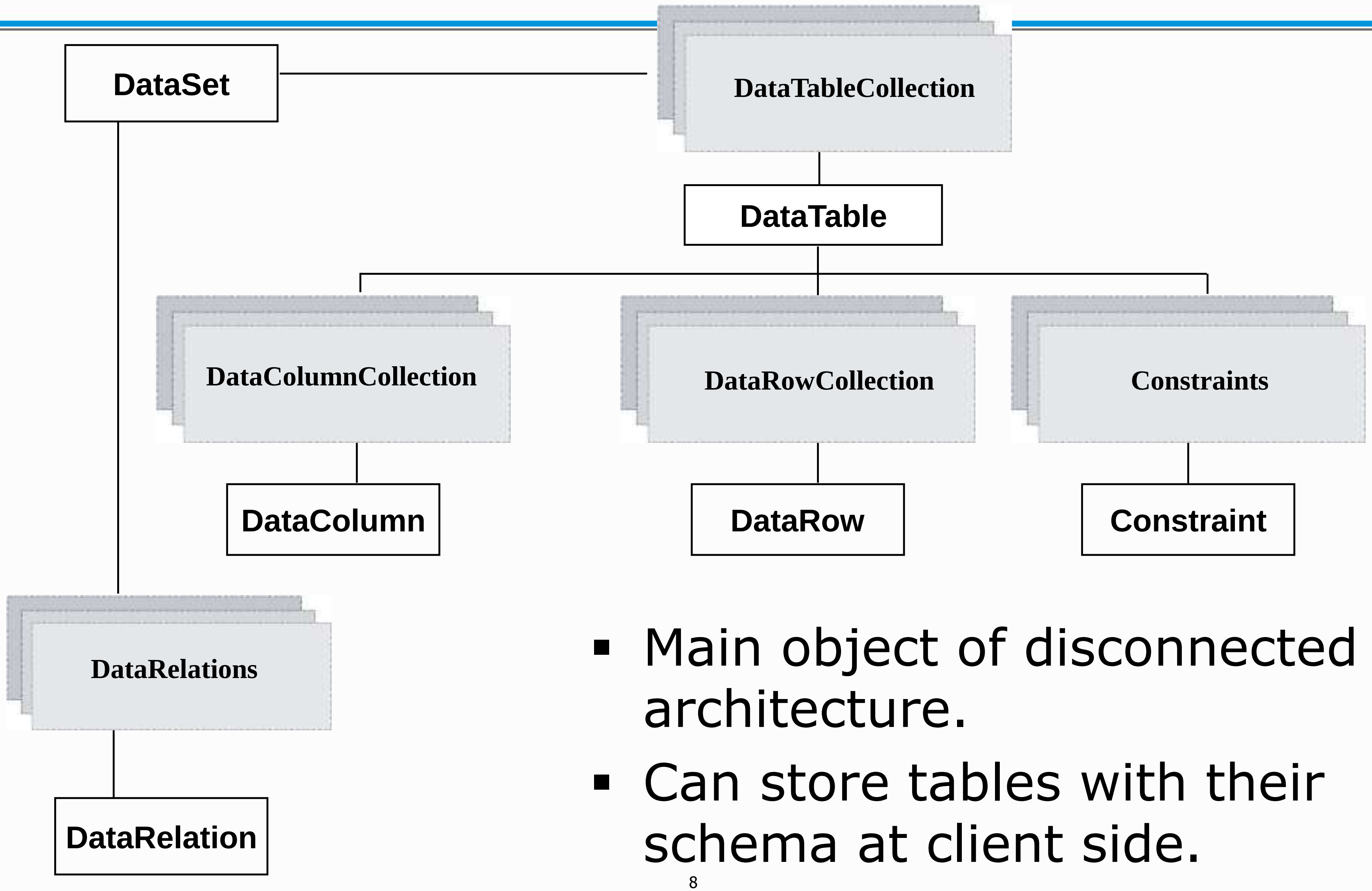
- SelectCommand
- InsertCommand
- DeleteCommand
- UpdateCommand

```
da.SelectCommand.CommandText="SELECT CustomerID,
                              ContactName FROM CustomersTab";
```

It can be Select, Update or Delete statement

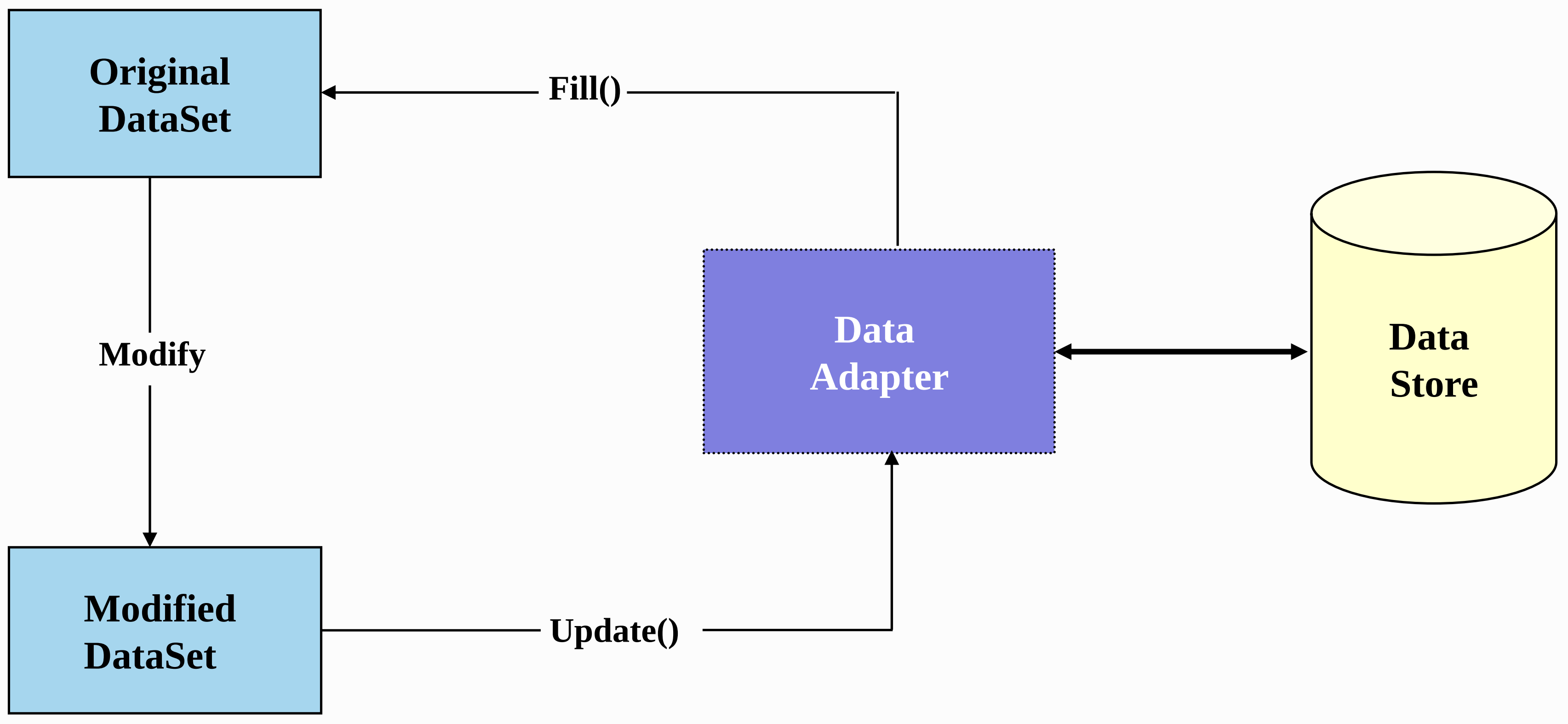
```
SqlCommand command=new SqlCommand("INSERT into Customers(
CustomerID,CompanyName) VALUES('ANATR','SunRise'");
da.InsertCommand = command;
```

DataSet object



- Main object of disconnected architecture.
- Can store tables with their schema at client side.

Retrieving & updating data using DataAdapter



Role of `CommandBuilder` Object

- Automatically generates Insert, Update, Delete queries by using the `SelectCommand` property of `DataAdaper`

```
SqlConnection con = new Sqlconnection("server=veena;
      Initial Catalog =Northwind;userid=sa;password=sa“);

SqlDataAdapter SqlDA=new SqlDataAdapter(
                           "Select * from Customers",con);

SqlCommandBuilder cmdBuilder = new SqlCommandBuilder(SqlDA);

DataSet ds = New DataSet();

SqlDA.Fill(ds, “Customers");
```

Constraints and DataViews

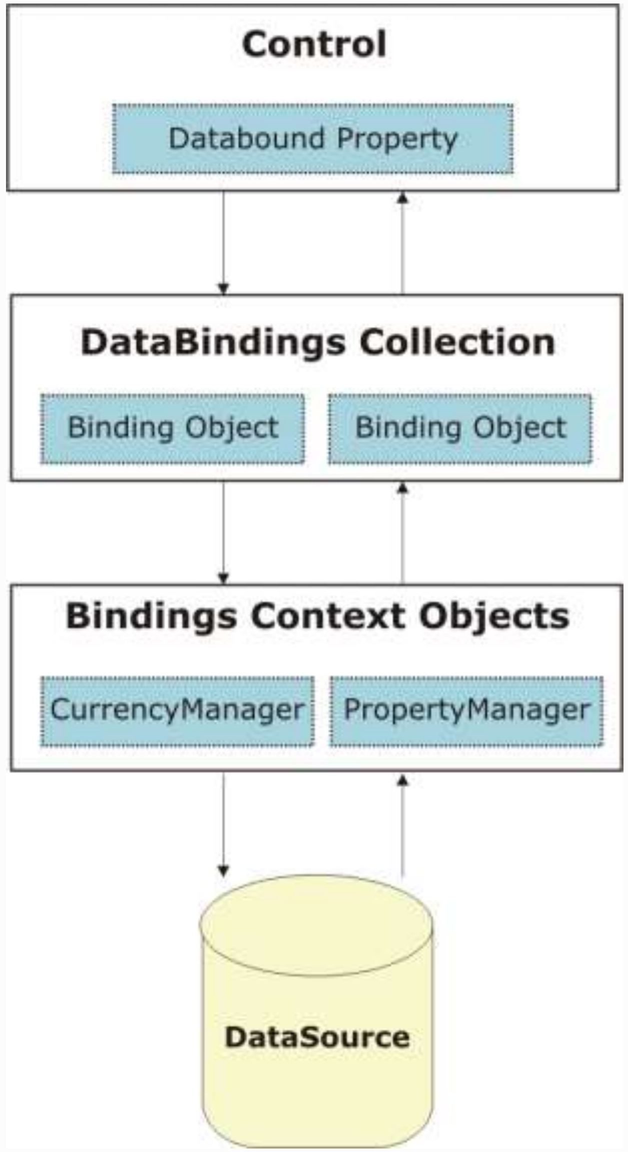
- Constraints restrict the data allowed in a data column or set of data columns.
  - Constraint classes in the `System.Data` namespace
    - `UniqueConstraint`
    - `ForeignKeyConstraint`
  - Using existing primary key constraint

```
daL.FillSchema(ds, schematype.Source,"Customers");
Or
da.MissingSchemaAction  = AddWithKey;
da.Fill(ds, “Customers")
```

- DataViews defines a model of the data, and different views that provide different representations of the data.

Binding the data

- Data Binding enables visual elements such as `TextBox`, `Datagrid` to connect to a data source such as DataSets, DataViews, Arrays etc.



Data Binding and Navigation of records

Data Binding

```
TextBox1.DataBindings.Add("text",ds.Tables(0),“deptId‟);
TextBox2.DataBindings.Add("text",ds.Tables(0),“dName‟);
```

Navigation of Records

Navigate to previous record

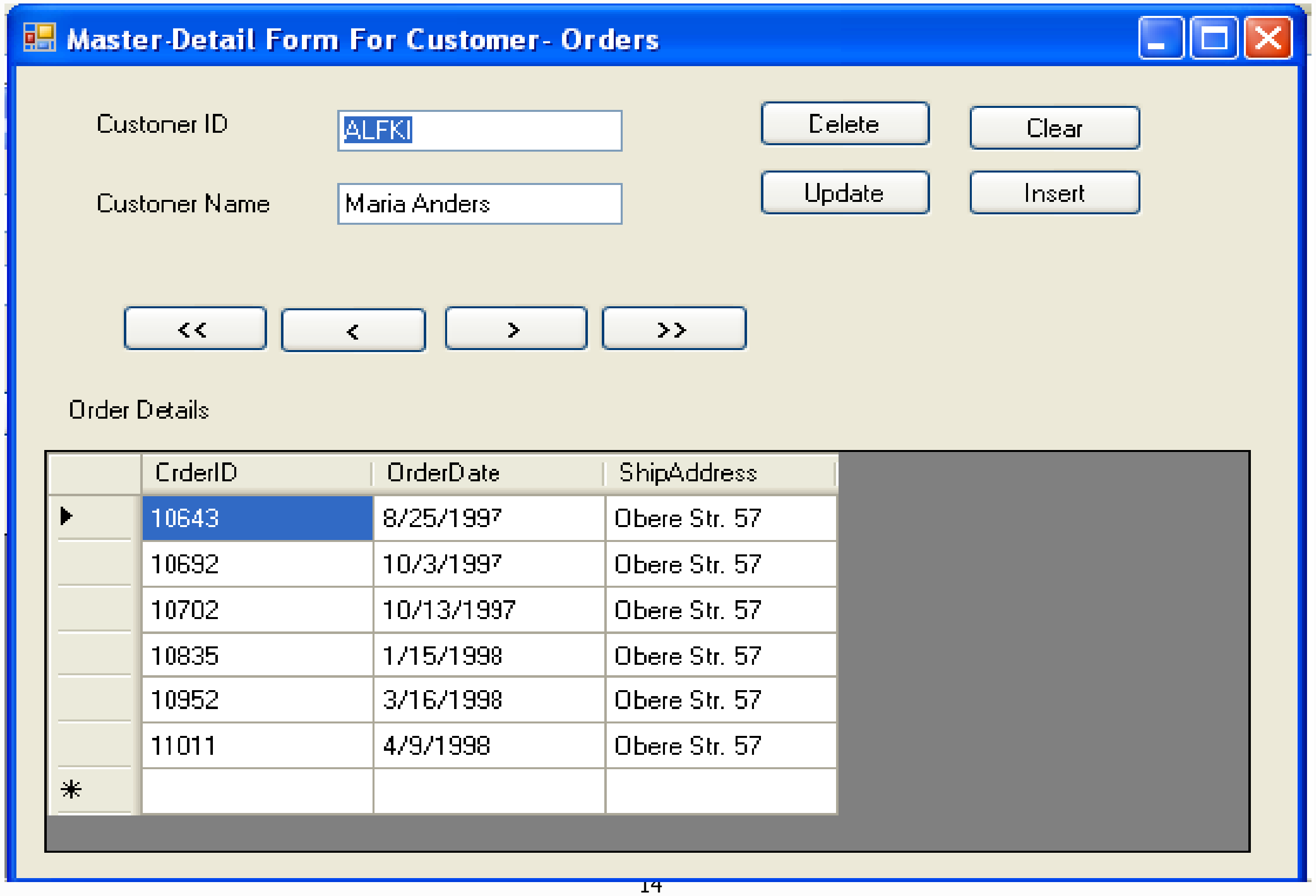
```
if(BindingContext[ds.Tables[0]].Position >0)
   BindingContext[ds.Tables[0]].Position =
       BindingContext[ds.Tables[0]].Position - 1;
```

Navigate to last record

```
BindingContext(ds.Tables(0)).Position =
  ds.Tables(0).Rows.Count - 1;
```

Master Detail Relationship

- Relates two Data Tables via Data Columns
- Data Type value of both Data Columns must be identical



Customer ID: ALFKI

Customer Name: Maria Anders

Order Details

	OrderID	OrderDate	ShipAddress
▶	10643	8/25/1997	Obere Str. 57
	10692	10/3/1997	Obere Str. 57
	10702	10/13/1997	Obere Str. 57
	10835	1/15/1998	Obere Str. 57
	10952	3/16/1998	Obere Str. 57
	11011	4/9/1998	Obere Str. 57
*			

DataRelation class

```
SqlDataAdapter CustomerDA = new  SqlDataAdapter
("SELECT customerId,  ContactName FROM Customers", conn);
SqlDataAdapter OrdersDA = new SqlDataAdapter
("SELECT OrderID,CustomerID,OrderDate,ShipAddress from Orders",
  conn);
CustomerDA.Fill(ds,"Customers");
OrdersDA.Fill(ds, "Orders");
DataRelation dRelation = new DataRelation
                ("Customer-Orders", ds.Tables[0].Columns[0],
 ds.Tables[1].Columns[1],true);

ds.Relations.Add(dRelation);
dataGridView1.DataSource = ds.Tables[0];
dataGridView1.DataMember = "Customer-Orders";
```

ADO.NET and XML

- With ADO.NET it is easy to
  - convert data into XML.
  - generate a matching XSD schema.
  - perform an `XPath` search on a result set.
  - interact with an ordinary XML document through the ADO.NET data objects.
- XML Schema Definition XSD
  - It is a dialect of XML for describing data structures.
  - Strongly Typed DataSets are made possible through inheritance and an XML Schema Definition (`XSD`).
- XPath is a language for extracting information from XML files.

DataSet XML Methods

GetXml()
GetXmlSchema()
ReadXml()
ReadXmlSchema()
WriteXml()
WriteXmlSchema()
InferXmlSchema()

Concurrency and Disconnected Architecture

- Disadvantage of disconnected architecture
  - Conflict can occur when two or more users retrieve and then try to update data in the same row of a table.
  - The second user's changes could overwrite the changes made by the first user.
- Solutions
  - Optimistic concurrency
    - Retrieves and updates just one row a time.

Quick Recap . . .

- Application does not stay connected to the database in disconnected scenario.
- `DataAdapter` object forms a bridge between the `DataSet` object and data source.
- `DataSet` object is in-memory representation of data and could contain data from any `DataSet`.
- `DataSet` is a collection of different objects like `DataTable`, `DataRelations`, `DataRow`, `DataColumn` and Constraints.
- `CommandBuilder` is used to generate `INSERT`,`UPDATE` and `DELETE` commands for the `DataAdapter`.
- `BindingContext` class manages the binding of control to the database field and also helps in navigation of records.
- Master Detail relationships could be easily used in the application using the `DataRelation` class.
- `DataSets` store their internal structure in a standardized format called XML Schema Definition (XSD)