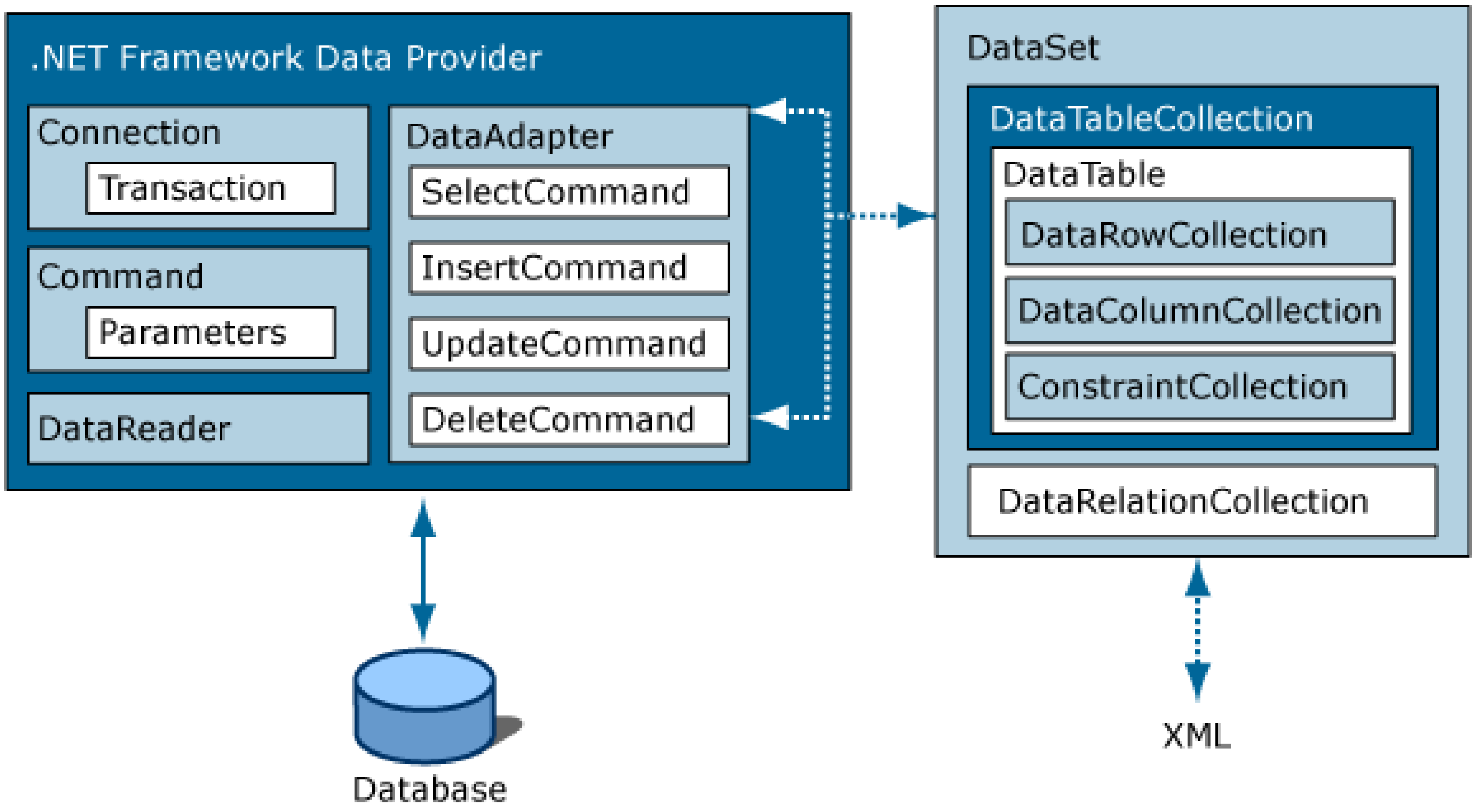
ADO.NET

ADO.NET Architecture



Different features of ADO.Net

- Object oriented data access (Classes to connect, query, and administer data sources)
- Unique data access(Uniform API for accessing data from different sources)
- Differentiate between command-based and disconnected data (Enables data to be accessed through a connection or a data set)
- Data binding (Binds a data source to ASP.NET, Windows Forms, or WPF controls)

Data Providers

- Data Provider for SQL Server
- Data Provider for OLE DB
- Data Provider for ODBC
- Data Provider for Oracle

SqlConnection Class

- How to connect to a Database
- Executing a Query and Retrieving the Results
- Disconnecting from a Database

Connecting Database

Step 1 : Define a connection string for the database

Step 2 : Create an object of the Connection Class

Step 3 : Assign the Connection Class object to the Connection String

Step 4 : Invoke the Open method on the Connection object

Querying and Retrieving the Results

```
string constr1 = @"Data Source=.\sqlexpress;Initial
Catalog=HR;Integrated Security=True";
SqlConnection conn1 = new SqlConnection();
conn1.ConnectionString = constr1;
conn1.Open();
SqlCommand comm1 = new SqlCommand("Select * from Dept",conn1);
SqlDataReader r1 = comm1.ExecuteReader();
DataTable dt1 = new DataTable();
dt1.Load(r1);
dataGridView1.DataSource = dt1;
conn1.Close();
```

Closing Database Connection

To disconnect from the database you need to invoke the Close() or Dispose() method on the Connection object.

To make sure that the connection is closed

- Create the Connection object in within Using statement
- At the end of the Using statement, the Connection object is automatically disposed
- The Dispose() method closes the database connection

Data Adapter

- It is used to exchange data between a data source and a dataset.
- It can take the form of references to SQL statements or stored procedures that are invoked to read or write to a database.
- Following are the available data adapters
  - OleDbDataAdapter
  - SqlDataAdapter
  - OdbcDataAdapter
  - OracleDataAdapter

Creating data adapter

- It can be created by using server explorer. You can drag database elements onto a form or component and generate the adapter automatically.
- It can be created by using Data Adapter Configuration Wizard. It prompts you for all the information required to configure the data adapter (including parameters) and, if necessary, create a connection.
- It can be created manually. Here you need to drag a data adapter from the Toolbox, and then configure it yourself using the Properties window.

Transaction overview

A transaction is an atomic unit of work

- Operations within the transaction succeed or fail atomically

Characteristics of a transaction

- Atomicity
- Consistency
- Isolation
- Durability

How to create and use transactions

- Open a database connection
- Begin the transaction
- Create and run commands within the transaction scope
- Commit or roll back the transaction

Transaction and concurrency

- Data conflicts arise when multiple users attempt to read or modify the same data in a database
- Use one of the following techniques to manage concurrent updates:
- Pessimistic concurrency
- Optimistic concurrency
- "Last in wins‟
- Concurrency errors:

- Dirty reads
- Non-repeatable reads
- Phantom reads

Best practices of implementing transactions

- Minimize the duration of transactions
- Do not perform any user interaction during a transaction
- Do not perform long-running tasks during a transaction
- Specify an isolation level to minimize concurrency errors
- Specify an isolation level to minimize the use of database locks

Sample Code

```
conn1.Open();
//Transaction can start after opening connection
SqlTransaction tr = conn.BeginTransaction();
//Assign the transation to command
comm.Transaction = tr;
iCount = comm.ExecuteNonQuery();
if (txtFName.Text.Contains("1"))
{
    tr.Rollback();
    ShowMSG("Transaction Rolled back");
}
else
{
   tr.Commit();
}
```

Stored Procedure

It is a group of Transact-SQL statements compiled into a single execution plan. It can return data in four ways:

- Output parameters, which can return either data (such as an integer or character value) or a cursor variable (cursors are result sets that can be retrieved one row at a time).
- Return codes, which are always an integer value.
- A result set for each SELECT statement contained in the stored procedure or any other stored procedures called by the stored procedure.
- A global cursor that can be referenced outside the stored procedure.

Example

```
string sConn = @"Data Source=.\sqlexpress;Initial
Catalog=HR;Integrated Security=True";

using (SqlConnection conn = new SqlConnection(sConn))
{
   SqlCommand comm = conn.CreateCommand();
   comm.CommandType = CommandType.StoredProcedure;
   comm.CommandText = "GetEmployee";
   try
   {
      conn.Open();
      DataTable dt = new DataTable();
      dt.Load(comm.ExecuteReader());
      dataGridView1.DataSource = dt;
   }
   catch (Exception ex)
   {
```

Showing data in a Combo box

```
private void LoadComboDept()
{
    using (SqlConnection conn = new SqlConnection(sConn))
    {
         SqlCommand comm = conn.CreateCommand();
         comm.CommandText = "SELECT * FROM Dept";
         try
         {
             conn.Open();
             SqlDataReader dr=  comm.ExecuteReader();

             while(dr.Read())
             {
                 comboDept.Items.Add(dr["DeptName"]);
             }
           }
          catch (Exception ex)
          {
              ShowMSG(ex.Message);
          }
```

Sorting the data

```
public void ShowData()
{
   DataClasses1DataContext db = new DataClasses1DataContext();
   var empname = from fname in db.Emps
                 orderby fname.FirstName descending
                 select fname;
   dataGridView1.DataSource = empname.ToList();
}
```

HCL

Filtering the data

```
public void ShowData()
{
  DataClasses1DataContext db = new DataClasses1DataContext();
  string lname = textBox1.Text;

  var empname = from fname in db.Emps
                where fname.LastName==lname
                select fname;

   dataGridView1.DataSource = empname.ToList();

}
```