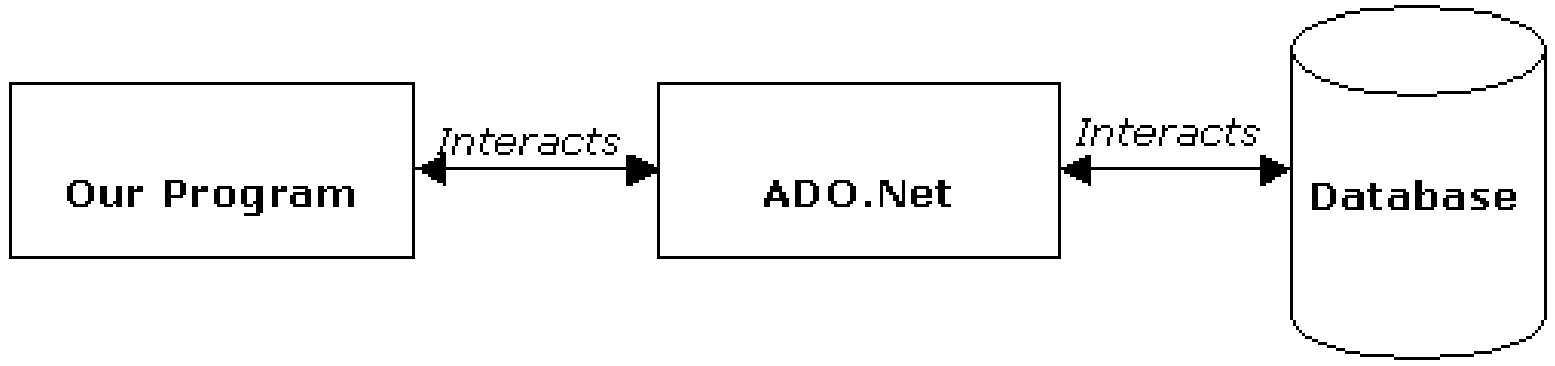
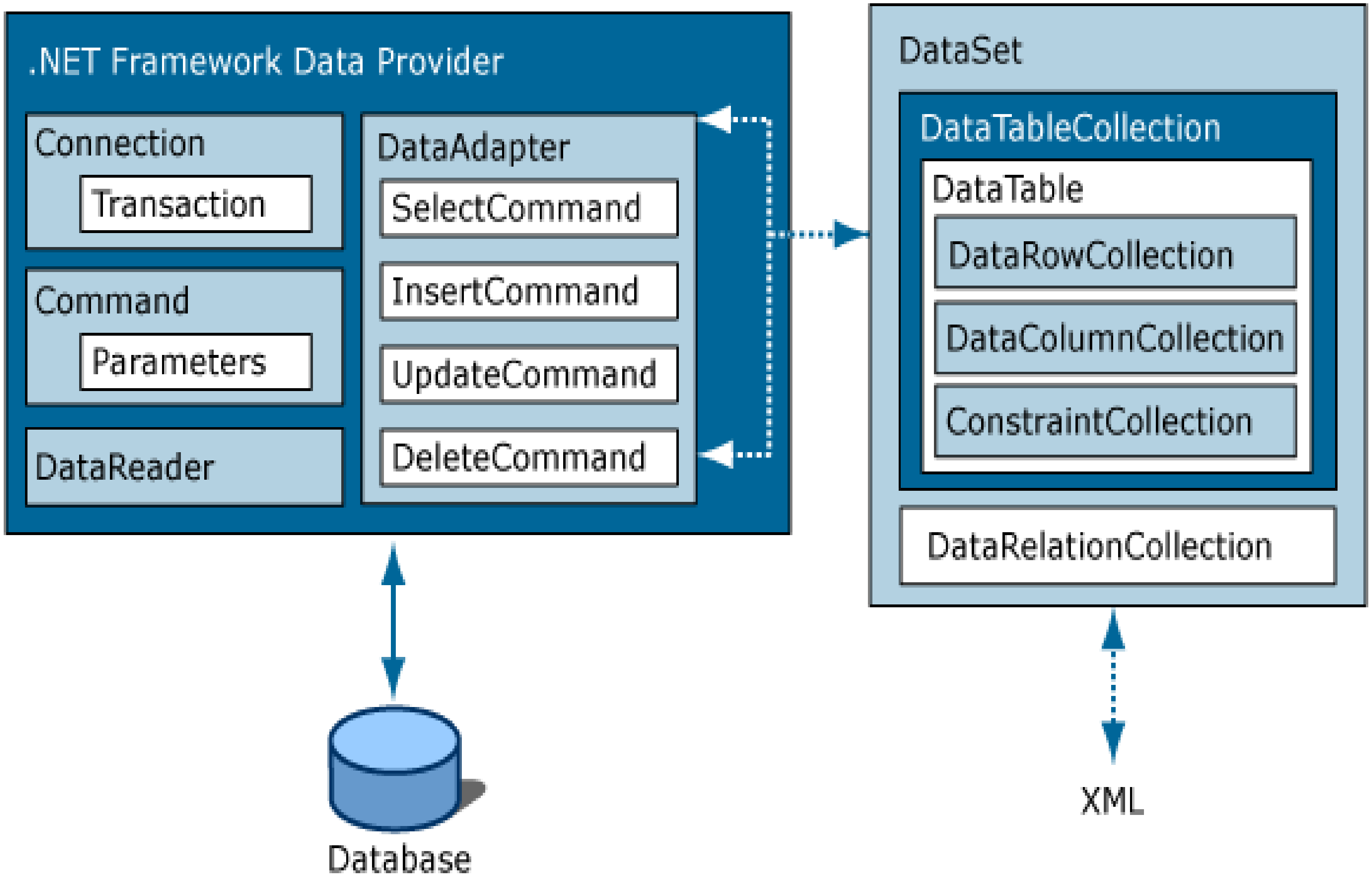
What is ADO.NET?

- A data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways .
- An object oriented framework that allows you to interact with database systems



ADO.NET Architecture



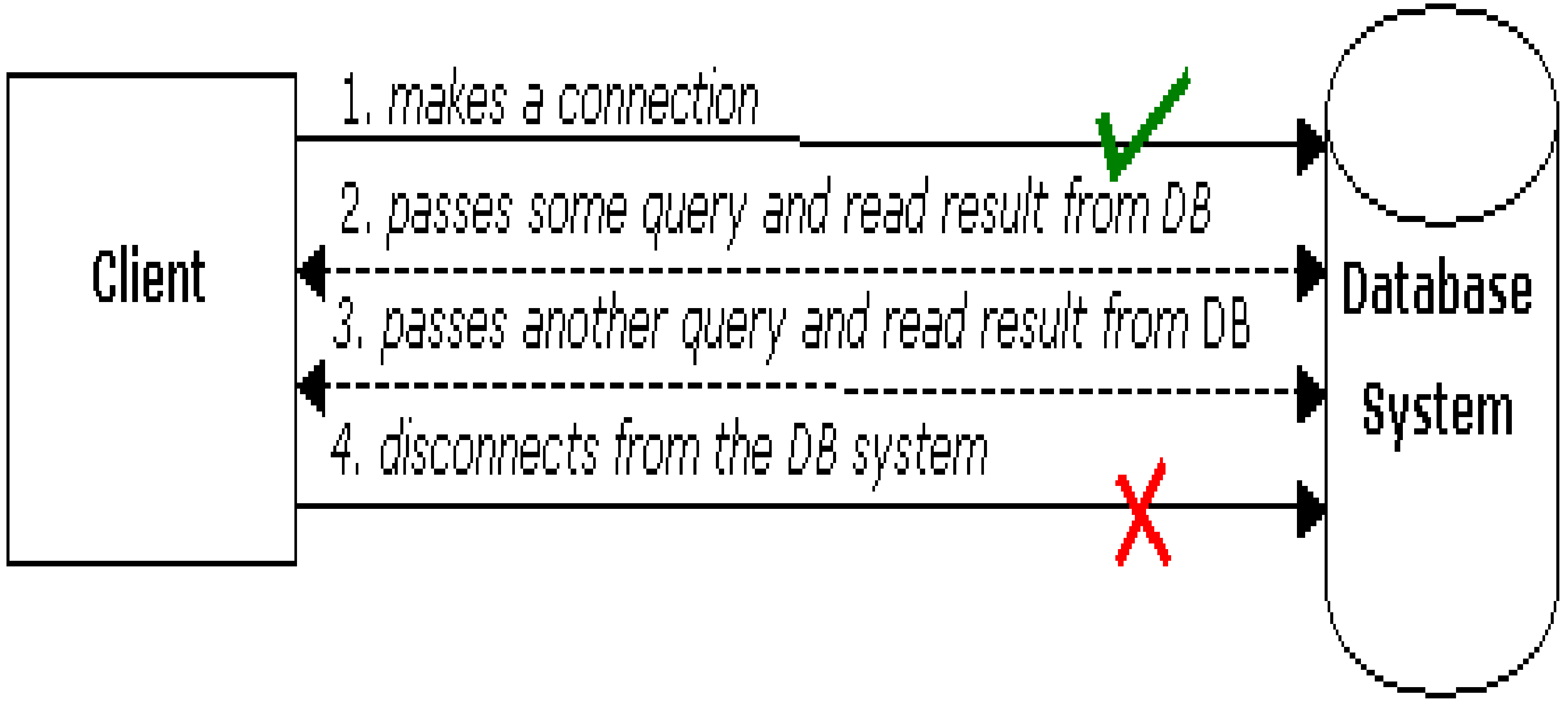
ADO.NET Core Objects

- Core namespace: System.Data
- .NET Framework data providers:

Data Provider	Namespace
SQL Server	`System.Data.SqlClient`
OLE DB	`System.Data.OleDb`
ODBC	`System.Data.Odbc`
Oracle	`System.Data.OracleClient`

ADO.NET Core Objects

Object	Description
Connection	To interact with a database, you must have a connection to it. A connection object is used by command objects so they will know which database to execute the command on
`Command`	You use a command object to send SQL statements to the database. A command object uses a connection object to figure out which database to communicate with.


Client
1. makes a connection
2. passes some query and read result from DB
3. passes another query and read result from DB
4. disconnects from the DB system
Database System

Connection Class

- You need to establish a connection class object for inserting, updating, deleting and retrieving data from a database.

- The Connection class allows you to establish a connection to the data source.

- The Connection class object needs the necessary information to discover the data source and this information is provided by a connection string.

- **Connection Strings:**
- You need to supply a connection string in the Connection class object. The connection string is a series of name/value settings separated by semicolons (;). A connection string requires a few peices of information such as the location of the database, the database name and the database authentication mechanism.

Command

- The Command Class allows performing any data definition tasks such as creating and altering tables and databases, retrieving, updating and deleting of records. The Command object used to execute SQL queries can be inline text or a Stored Procedure.

- **DataReader**
- DataReader is used to read the data from database and it is a read and forward only connection oriented architecture during fetch the data from database.

- DataReader will fetch the data very fast when compared with dataset. Generally we will use ExecuteReader object to bind data to datareader.

-

- Protected void BindGridview()
- {
- using (SqlConnection conn = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test"))
- {
- con.Open();
- SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
- SqlDataReader sdr = cmd.ExecuteReader();
- gvUserInfo.DataSource = sdr;
- gvUserInfo.DataBind();
- conn.Close();
- }
- }
- Holds the connection open until you are finished (don't forget to close it!).
- Can typically only be iterated over once.
- Is not as useful for updating back to the database

- **DataSet**
- DataSet is a disconnected orient architecture that means there is no need of active connections during work with datasets and it is a collection of DataTables and relations between tables. It is used to hold multiple tables with data. You can select data form tables, create views based on table and ask child rows over relations. Also DataSet provides you with rich features like saving data as XML and loading XML data.

- protected void BindGridview()
- {
-     SqlConnection conn = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test");
-     conn.Open();
-     SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
-     SqlDataAdapter sda = new SqlDataAdapter(cmd);
-     DataSet ds = new DataSet();
-     da.Fill(ds);
-     gvUserInfo.DataSource = ds;
-     gvUserInfo.DataBind();
- }

- **DataAdapter**
- DataAdapter will acts as a Bridge between DataSet and database. This dataadapter object is used to read the data from database and bind that data to dataset. Dataadapter is a disconnected oriented architecture. Check below sample code to see how to use DataAdapter in code:

- **ExecuteScalar()** only returns the value from the first column of the first row of your query. **ExecuteReader()** returns an object that can iterate over the entire result set. **ExecuteNonQuery()** does not return data at all: only the number of rows affected by an insert, update, or delete.

- protected void BindGridview()
- {
- SqlConnection con = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test");
- conn.Open();
- SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
- SqlDataAdapter sda = new SqlDataAdapter(cmd);
- DataSet ds = new DataSet();
- da.Fill(ds);
- gvUserInfo.DataSource = ds;
- gvUserInfo.DataBind();
- }
- Lets you close the connection as soon it's done loading data, and may even close it for you automatically
- All of the results are available in memory
- You can iterate over it as many times as you need, or even look up a specific record by index
- Has some built-in faculties for updating back to the database.

- DataTable represents a single table in the database. It has rows and columns. There is no much difference between dataset and datatable, dataset is simply the collection of datatables.
- protected void BindGridview()
- {
-     SqlConnection con = new SqlConnection("Data Source=abc;Integrated Security=true;Initial Catalog=Test");
-     conn.Open();
-     SqlCommand cmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);
-     SqlDataAdapter sda = new SqlDataAdapter(cmd);
-     DataTable dt = new DataTable();
-     da.Fill(dt);
-     gridview1.DataSource = dt;
-     gvidview1.DataBind();
- }