Chapter 12

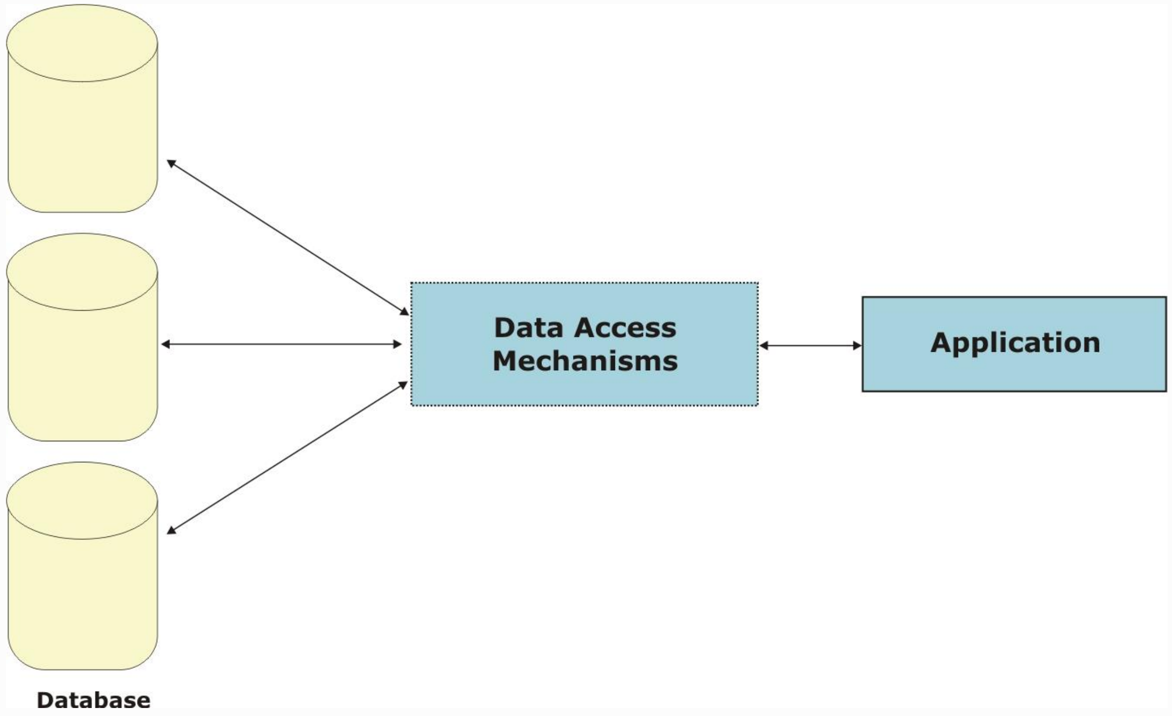
ADO.NET - Connected

Objectives

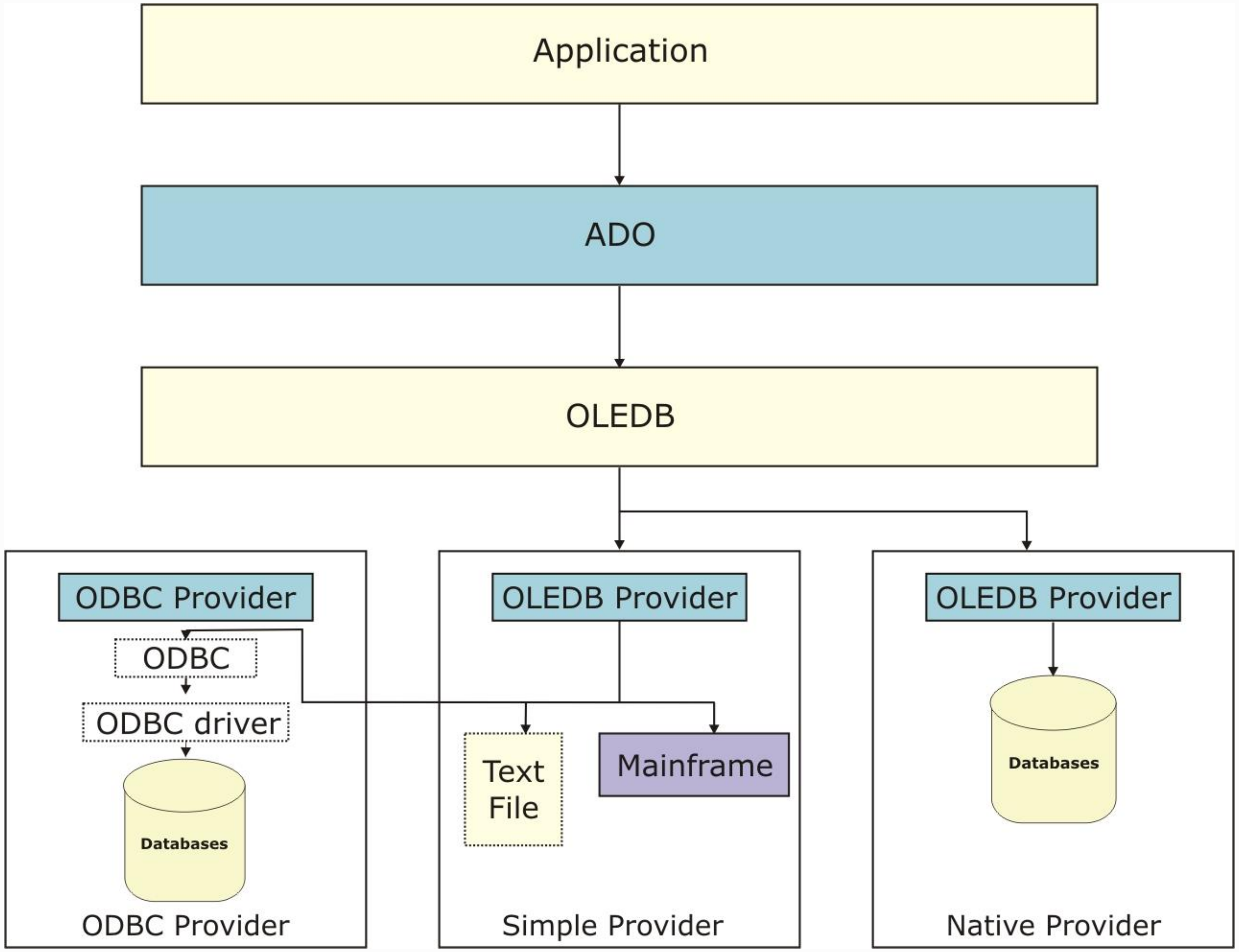
- On completion of this Session you will be able to :
  - Name the features of ADO.NET
  - List different .NET data providers
  - Differentiate between Connected and Disconnected environment
  - Connect to SQLServer database using connection object
  - Create and use command object to query the database.
  - Use a DataReader object to read the data fetched from the database.
  - Call a stored procedure created in SQLServer database to perform insert, update or delete operations.

Ways to access Data…

- ODBC (Open Database Connectivity)
- DAO (Data Access Objects)
- RDO (Remote Data Objects)
- ADO (ActiveX Data Objects)



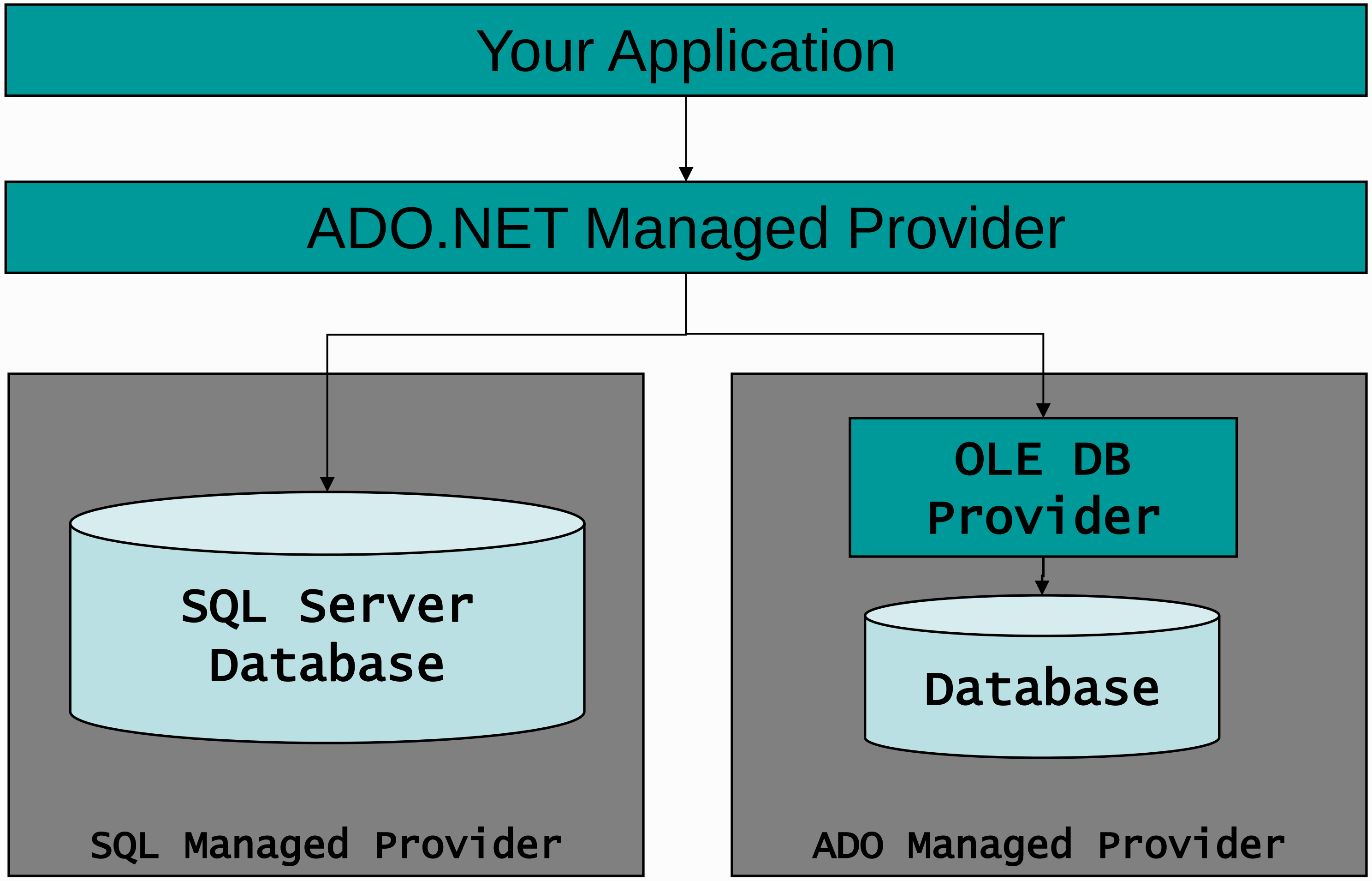
Accessing Data using ADO



What is ADO.NET ?

- A rich set of classes, interfaces, structures and enumerated types that manage data access from different types of data stores
- Features
  - A robust disconnected model.
  - Integrate XML support.
  - Data from varied Data Sources
  - Familiarity to ADO programming model.
  - Enhanced performance

Managed Provider



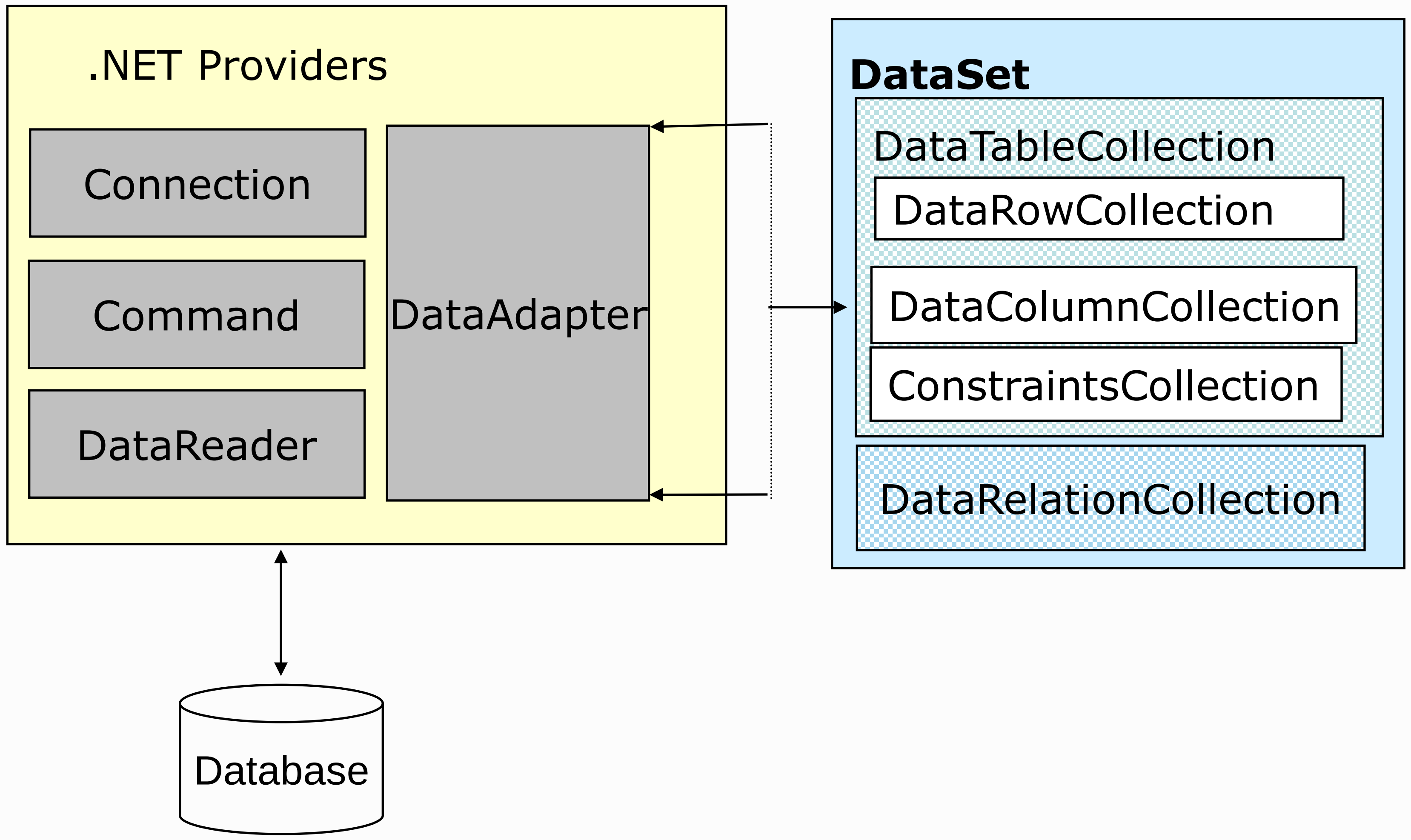
Connected vs Disconnected architecture

	Connected	Disconnected
State of connection	Constantly kept open	Closed once data is fetched in cache at client side
Scalability	limited	more
Current data	Always available	Not up to date

ADO.NET Components

- .NET Data Providers
  - Allow users to interact with different type of data sources.
    - ODBC Data Provider
    - OLEDB Data Provider
    - SQL Data Provider(for SQL Server 7.0 and above)
    - Oracle Data Provider
- DataSet
  - Explicitly designed for disconnected architecture.

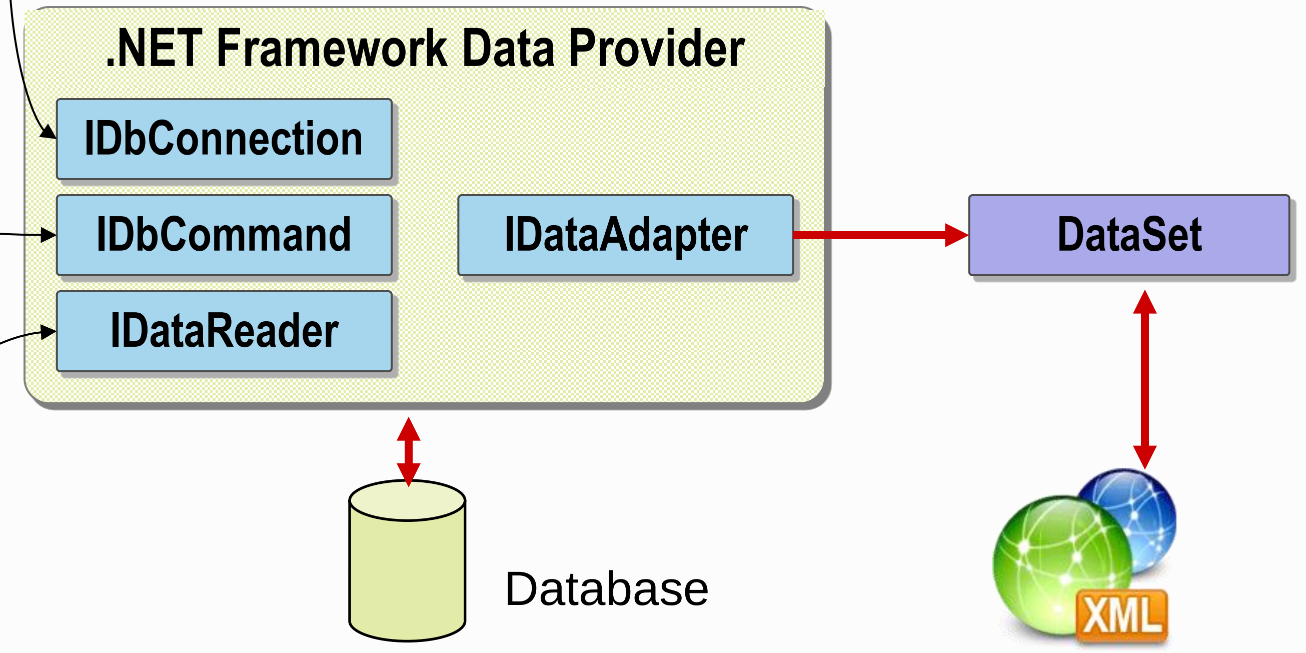
ADO.NET Objects



ADO.NET Interfaces

Represents an open connection to a data source

Represents an SQL statement that is executed while connected to a data source



Provides a means of reading one or more forward-only streams of result sets obtained by executing a command at a data source

Connection Object

- Has the responsibility of establishing connection with the data store.
- Connection has to be explicitly closed in finally block.

```
 SqlConnection conSQL = new SqlConnection();
conSQL.ConnectionString = "server=DatabaseServer;Initial
Catalog =northwind;user id=sa;password=sa”;
conSQL.Open( );
```

Command Object

- Used to specify the type of interaction to perform with the database like select, insert, update and delete.
- Exposes properties like :
  - `CommandText`
  - `CommandType`
  - `Connection`
- Exposes several Execute methods like
  - `ExecuteScalar()`
  - `ExecuteReader()`
  - `ExecuteNonQuery()`

All methods require open connection

Inserting Data

```
string insertString  = "insert into dept(deptId, deptName,
loc) values(10,’Mktg’, ‘Mumbai’)";
```

```
string updateString = "update dept set deptName=‘Marketing’
                        where deptName=‘Mktg’ ”;
```

```
string deleteString = "delete from dept where
                      deptName = ‘ABC’;
```

String could be updateString or deleteString

```
conSQL.Open();
SqlCommand cmd = new SqlCommand(insertString,conSQL);

cmd.ExecuteNonQuery() ;
```

Getting Single value

```
SqlCommand cmdSQL = new SqlCommand( );
cmdSQL.Connection = conSQL;
cmdSQL.CommandText = "Select Count(*) from emp";
int cnt  = (int)cmdSQL.ExecuteScalar();
MessageBox.Show(cnt.ToString());
```

Returns a single object, cast it to integer

The **DataReader** Object

- Used to only read data in forward - only sequential manner.

```
string queryStr ="Select deptName,loc from dept";
conSQL.Open();
SqlCommand cmdSQL = new SqlCommand(queryStr,conSQL);
SqlDataReader dataRead = cmdSQL.ExecuteReader();
while(datRead.Read())
{
  MessageBox.Show ("Last Name is" +
                   dataRead[0].ToString());
  MessageBox.Show ("First Name is" +
                   dataRead[1].ToString());
}
datRead.Close();
```

returns a reference to DataReader object.

Adding Parameters to Command

```
conSQL.Open();
SqlCommand cmd = new SqlCommand(“select * from
                  emp where empNo = @eno",conSQL);
SqlParameter param  = new SqlParameter();
param.ParameterName = "@eno";
param.Value = 100;
cmd.Parameters.Add(param);
SqlDataReader rdr;
rdr = cmd.ExecuteReader();
while(rdr.Read())
{
      //display the data
}
```

Calling A Stored Procedure

```
CREATE PROCEDURE DeleteEmpRecord
(
  @eno int
 )
AS
delete from emp where empno = @eno;
RETURN
```

Stored procedure in SQLServer

Stored procedure name

```
SqlCommand cmd  = new SqlCommand(“DeleteEmpRecord", conSQL);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add (new SqlParameter ("@eno", 100));
```

Multiple Queries

- Multiple queries can be executed using a single command object.

```
cmd = new SqlCommand("select * from
justdept;select * from justEmp", conSQL);
dr = cmd.ExecuteReader();
. . .// code to access first resultset
bool result = dr.NextResult();
. . .// code to access next resultset
```

Quick Recap…

- ADO.NET is an object oriented set of libraries that allows to interact with data sources.
- Different .NET providers are available in ADO.NET library to support both connected as well as disconnected architecture.
- Command object uses open connection to query the database.
- DataReader object is used to only read the data in forward only direction.
- Stored procedure enhances the performance of the application.