leren. durven. doen.

SYNTRA WEST

C# Advanced

C# - ADO.NET

Inhoud

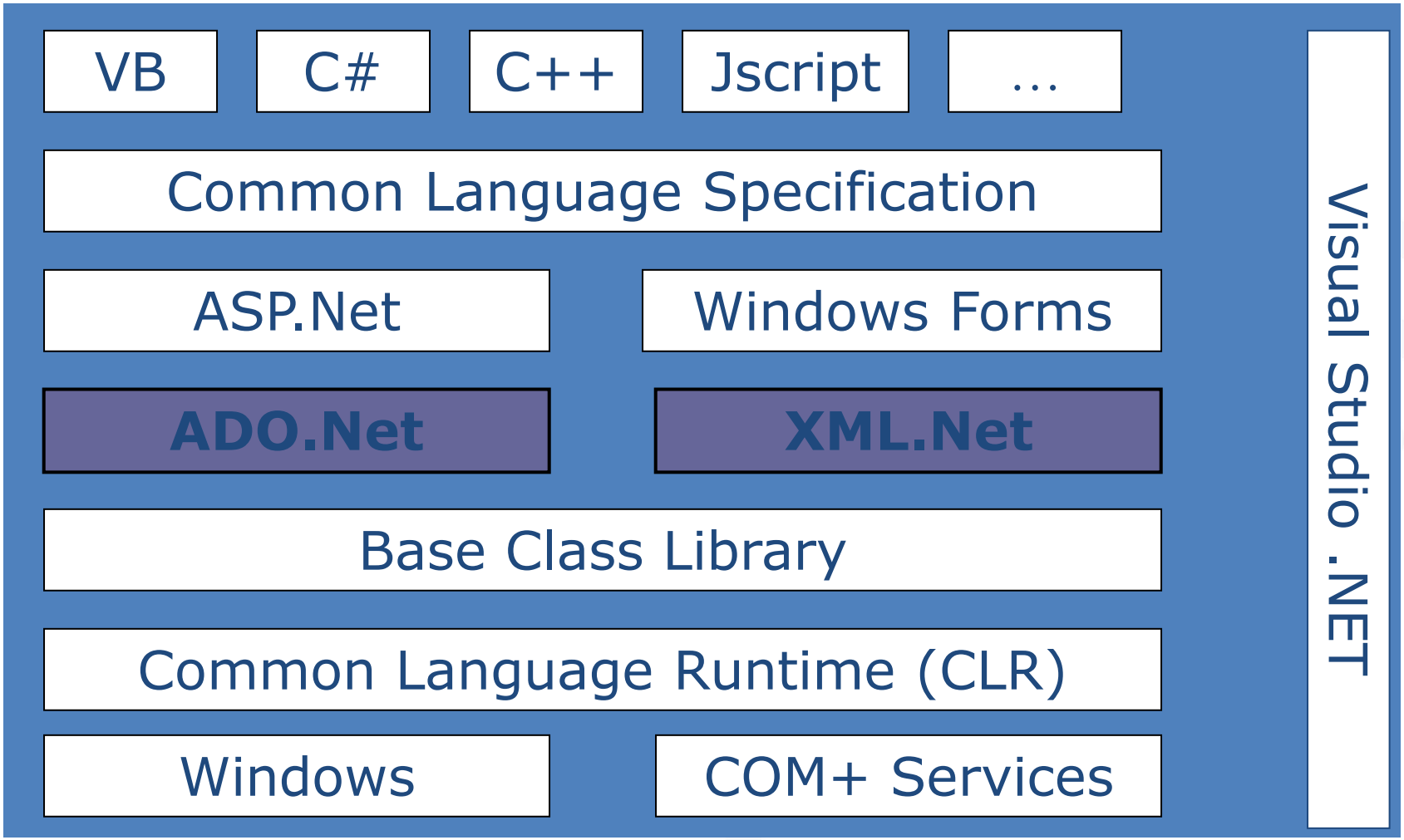
- Wat is ADO.Net?
- ADO.Net object structuur
- Connecteren op database
- Commands (queries uitvoeren)
- Readers en DataSets

Wat is ADO.Net?

= Active Data Objects . NET

- data access klassen in .Net
- Ontworpen voor efficiënte datatoegang
- ondersteuning voor XML en gedisconnecteerde record sets

ADO.Net in .Net Framework



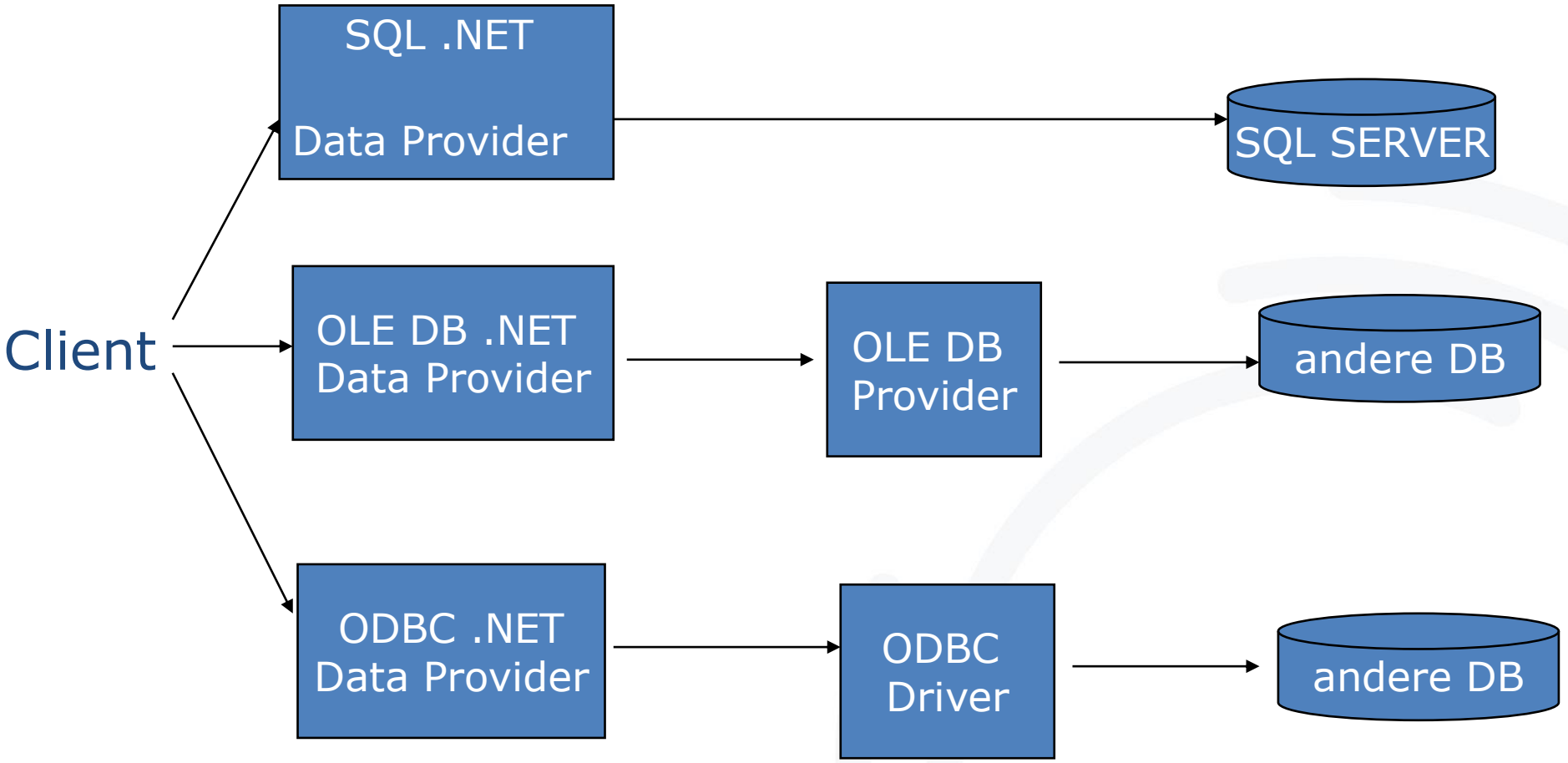
ADO.Net belangrijkste Functionaliteiten

Functionaliteit	ADO.Net
In memory - data bijhouden	Dataset bevat DataTables
Data Lezen/wijzigen/ver wijderen	Lezen kan sequentieel of niet-sequentieel
gegevens-bronnen (Data Sources)	Verschillende soorten data providers

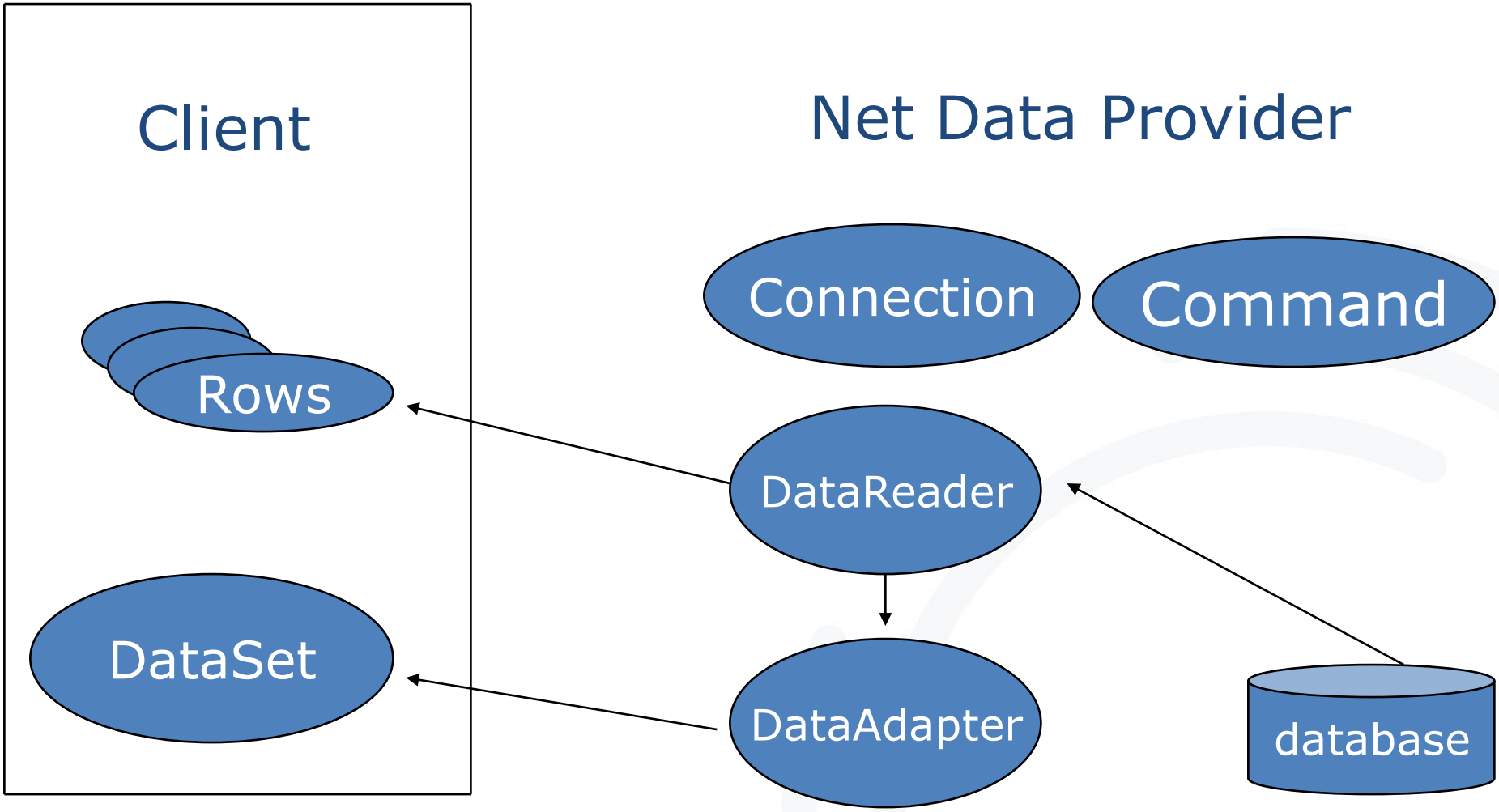
ADO.Net belangrijkste Functionaliteiten

FunctionaliteitéQ1A	ADO.Net
Gedisconnecteerde data	Zowel lezen als schrijven van gegevens
Datasets	DataSet ondersteuning voor XML

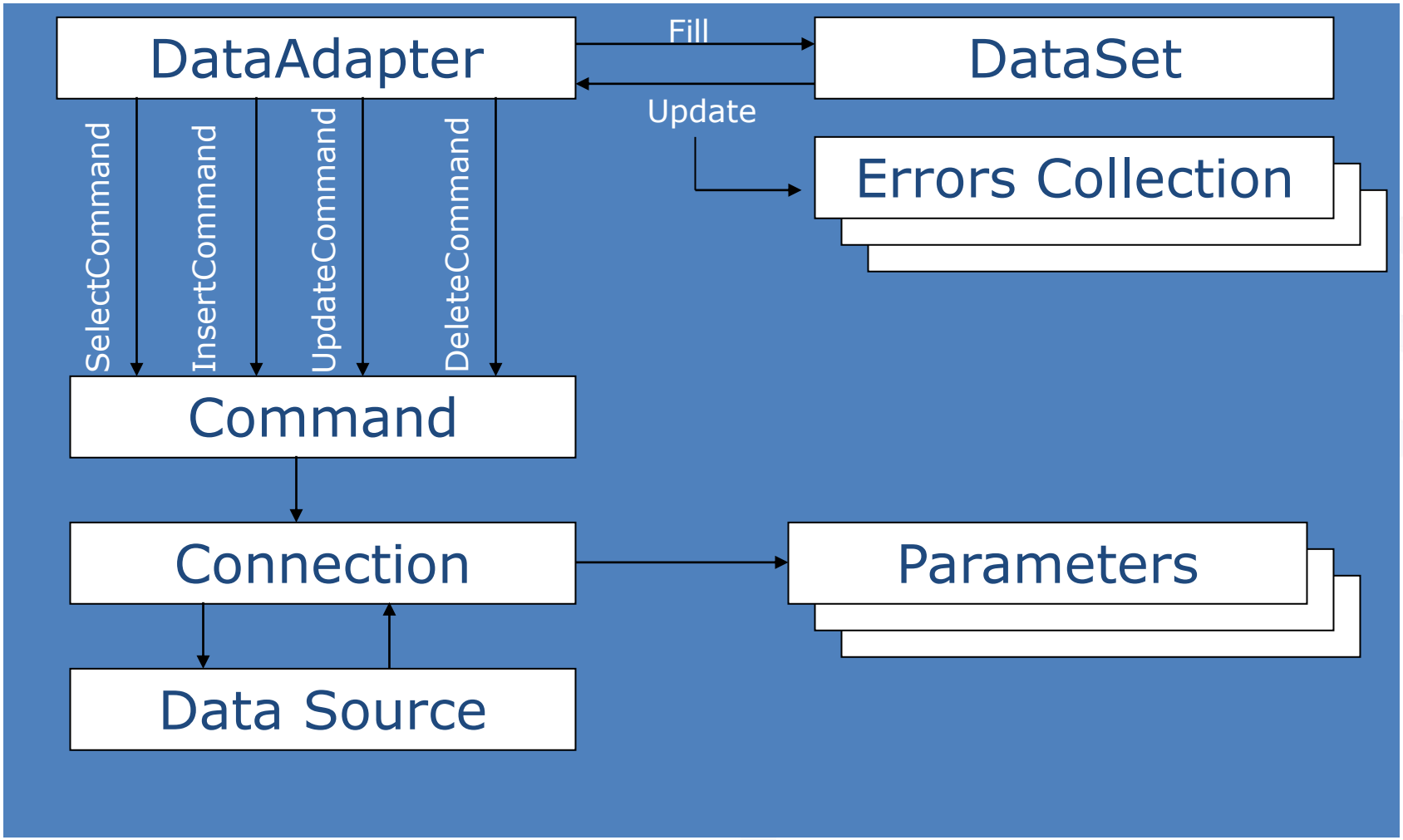
.NET Data Providers



Functionaliteit van Data Provider



ADO.Net object model



Namespaces

- **System.Data & System.Data.Common**
- **System.Data.SqlClient & System.Data.OleDB**
- **System.Data.SqlTypes**
- **System.XML & System.XML.Schema**

- **C#**

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

SQL Namespace Objecten

- `using System.Data.SqlClient;`
- **SqlConnection**
- **SqlCommand**
- **SqlDataReader**
- **SqlDataAdapter**
- **SqlParameter**
- **SqlParameterCollection**
- **SqlError**
- **SqlErrorCollection**
- **SqlException**
- **SqlTransaction**
- **SqlDbType**

Connectie leggen naar SQL database

```
using System.Data.SqlClient;

string _connectieString =
  "Data Source=.;Initial
Catalog=BierenDb;Integrated
Security=True";

SqlConnection conn = new SqlConnection();
conn.ConnectionString = connectionString;
conn.Open();
//…Acties gegevens ophalen en/schrijven
//van/naar Sql server database
conn.Close();
```

Connectie Pooling

- **ADO.Net houdt een verzamelingen (pools) van connecties bij. Wanneer een connective geopend wordt, wordt deze uit de 'pool' gehaald. Wanneer een connectie gesloten wordt, wordt deze terug in de 'pool' geplaatst**
- ```
  string _connectieString =
    "Data Source=.;Initial Catalog=BierenDb;Integrated Security=True";
  ```
- ```
  SqlConnection conn = new SqlConnection();
  conn.ConnectionString = _connectieString;

  conn.Open();  //connectie is aangemaakt in Pool A.
  ```
- ```
  SqlConnection conn = new SqlConnection();
  conn.ConnectionString =
    "Integrated Security=SSPI;Initial Catalog=Northwind";
  conn.Open();
  // connectie Pool B wordt gecreëerd want de connectie strings is verschillend.
  ```
- ```
  SqlConnection conn = new SqlConnection();
  conn.ConnectionString = _connectieString;
  conn.Open(); // connectie uit Pool A want de connection string komt overeen met uit pool A
  ```

Gegevens Lezen/Schrijven

- SqlCommand
  - ExecuteReader
  - ExecuteNonQuery
  - ExecuteScalar
  - ExecuteXMLReader
- SqlDataAdapter
  - DataSet

SqlCommand object aanmaken

- **SqlCommand**
  meerdere constructors:
- `new SqlCommand()`
- `new SqlCommand(cmdText)`
- `new SqlCommand(cmdText, connection)`
- `new SqlCommand(cmdText, connection, transaction)`

SqlCommand object gebruiken

```
string sSelectQuery =
  "SELECT * FROM Bieren ORDER BY BierNr";
string sConnectionString =
  "Initial Catalog=BierenDb;
   Data Source=localhost;
   Integrated Security=SSPI;";
SqlConnection objConnect = new SqlConnection(sConnectString);
SqlCommand objCommand = new SqlCommand(sSelectQuery,
                                       objConnect);
/*
objCommand.CommandTimeout = 15;
objCommand.CommandType = CommandType.Text;
*/

objConnect.Open();

SqlDataReader drResults;
drResults = objCommand.ExecuteReader()

drResults.Close();
objConnect.Dispose();
```

SqlCommand Methoden

- **ExecuteReader() - Geeft DataReader object terug**
- **ExecuteNonQuery() - Geeft aantal gewijzigde rijen terug**
- **ExecuteXMLReader() - Geeft XMLReader object terug**
- **ExecuteScalar() – Geeft één enkele waarde terug (voor SQL queries die één enkele waarde teruggeven bv SUM(), Count(),AVG(),MAX(), MIN(),…).**

DataReader object

- **DataReader objecten zijn geoptimaliseerd voor snel, 'forward only' – afhalen van gegevens via een Command**
- **Een DataReader is niet gedisconnecteerd**

- **Toegang tot gegevens is per record (bv lijn uit database-tabel of view)**
- **Is 'Forward only'**
- **Is 'Read only'**
- **Meerdere recordsets zijn mogelijk**

Aanmaken van data reader object

```
SqlDataReader sqlReader;
sqlReader =
  sqlCommand.ExecuteReader();
while (sqlReader.Read())
{
   // rij per rij aflopen van resultaat,
  waarde uit kolom opvragen via
  sqlReader["kolomnaam"]
}
sqlReader.Close();
```

Andere Methoden

- GetString(), GetInt(),GetFloat….
- GetValues()
- …

DataSets

- In-memory presentatie van gegevens uit een database/XML
- Operaties worden uitgevoerd op de DataSet, niet rechtstreeks op de data source
- Kan worden aangemaakt via een DataAdapter of XML schema en/of document

Namespace System.Data

- namespace `System.Data` ➔ bevat volgende klassen:

  `DataColumn, DataRow, DataTable, DataGridView, DataSet`

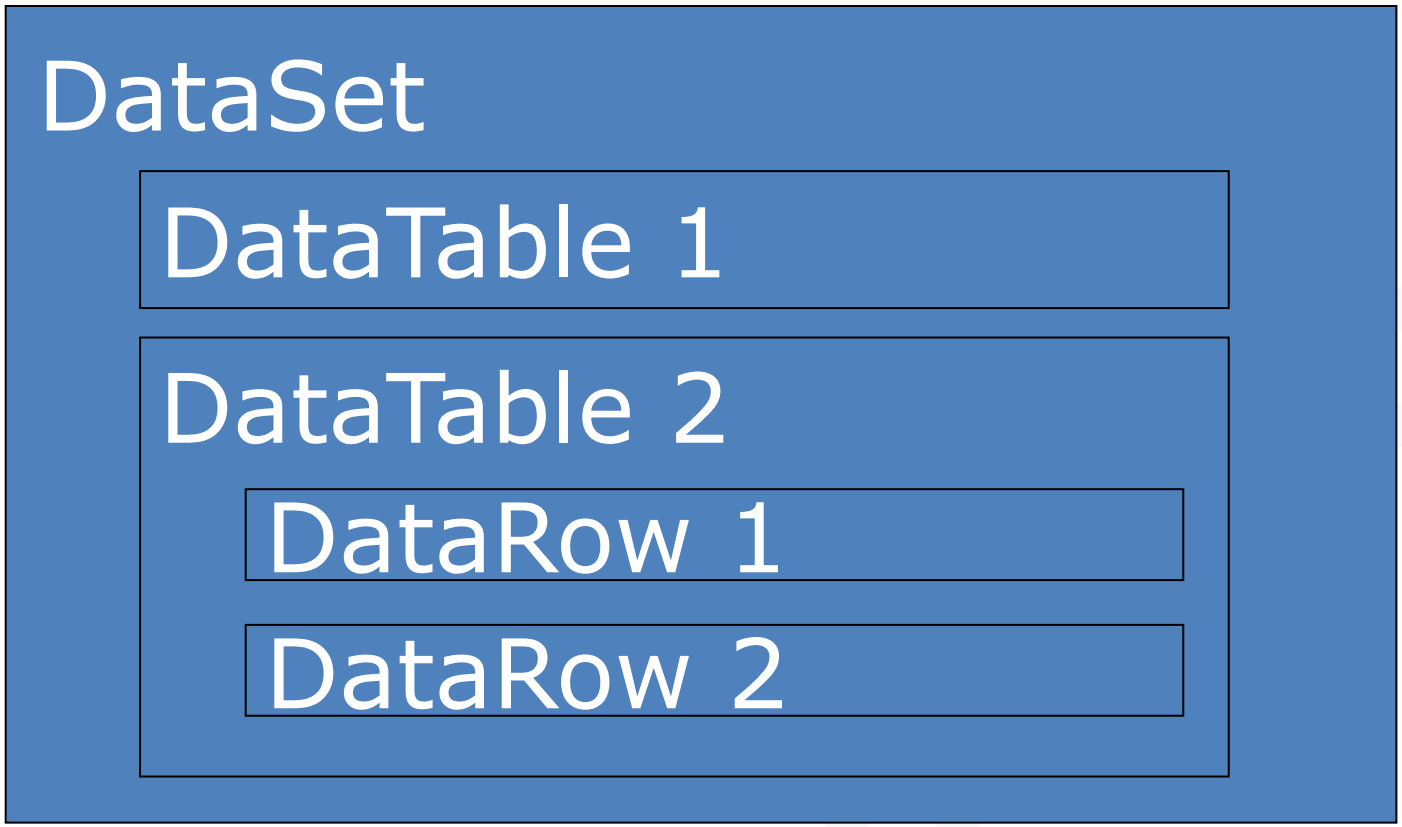
- `DataColumn` ➔

  Voorstelling van één kolom met gegevens uit een `DataTable`

```
DataColumn colName = new DataColumn();

colName.DataType =
                Type.GetType("System.String");
```

Sets, Tables en Rows



DataTable ➔ **Voorstelling van een tabel (in geheugen)**

- **kolommen definiëren en toevoegen**
- **rijen toevoegen en vullen**
- **bepaling van de primaire sleutel**

```
DataColumn[] PK = new DataColumn[1];
PK [0] = MyTable.Columns["KlantID"];
MyTable.PrimaryKey = PK;
```

```
DataRow row = wnTable.NewRow(); …
```

Kolom toevoegen aan een DataTable:

```
DataTable myTable =
                    new DataTable("MyTable");

myTable.Columns.Add(colName);
```

AutoIncrementele kolom definiëren:

```
DataColumn klantIDKolom = new DataColumn
   ("KlantID", Type.GetType("System.Int32"));

klantID.AutoIncrement = true;

klantID.AutoIncrementSeed = 100;

klantID.AutoIncrementStep = 10;

DataColumn kolomVoornaam = new DataColumn();

colName.DataType =
Type.GetType("System.String");
```

```
DataTable wnTable = new DataTable("Werknemer");
// kolommen ID, Voornaam, Familienaam toevoegen
wnTable.Columns.Add(KlantIDKolom);
wnTable.Columns.Add(kolomVoornaam);
wnTable.Columns.Add(kolomFamilienaam);
```

DataRow ➔ Voorstelling van één rij van een DataTable

```
DataRow row = wnTable.NewRow();     // !
row["Voornaam"] = "Jos";
row["Familienaam"] = "De Klos";
wnTable.Rows.Add(row);
// .DataRow.Delete()
// .AcceptChanges(), .RejectChanges(),
// .DataRow.Deleted , .Detached, .Modified
// .New, .Unchanged
```

Fouten opvangen via

`try{} catch{} (finally{})` constructie:

```
try
{       wnTable.Rows[rowNumber].Delete();
        wnTable.AcceptChanges();
}
catch(Exception ex)
{
      Console.WriteLine(ex.Message);
}
```

Filteren van DataTable

```
string filterStr = "Voornaam ='Jan'";
DataRow[] rijenMetVoornaamJan =
 wnTable.Select(filterStr);
```

Mogelijke Filter operatoren en Methodes:

```
AND   OR   // Logische operatoren
< <= > >= = <> IN LIKE   // vergelijkingen
+   -   *   /   %      // rekenkundige operatoren
*xxx   xxx*   %xxx   xxx%   // wildcards * en %
Sum Avg Min Max Count StDev Var   // Methoden
```

Sorteren van gegevens in DataTable

```
DataRow[] gesorteerd = wnTable.Select
          ( filterStr, "Familienaam DESC");
// "Familienaam ASC" om alfabetisch te sorteren
```

Voorbeelden:

```
DataRow dr;
for(int i=0; i < gesorteerd.Length; i++)
 {
     temp = gesorteerd[i];
     Console.WriteLine(temp["Familienaam"])
 }
```

Lezen / schrijven naar / van bestanden

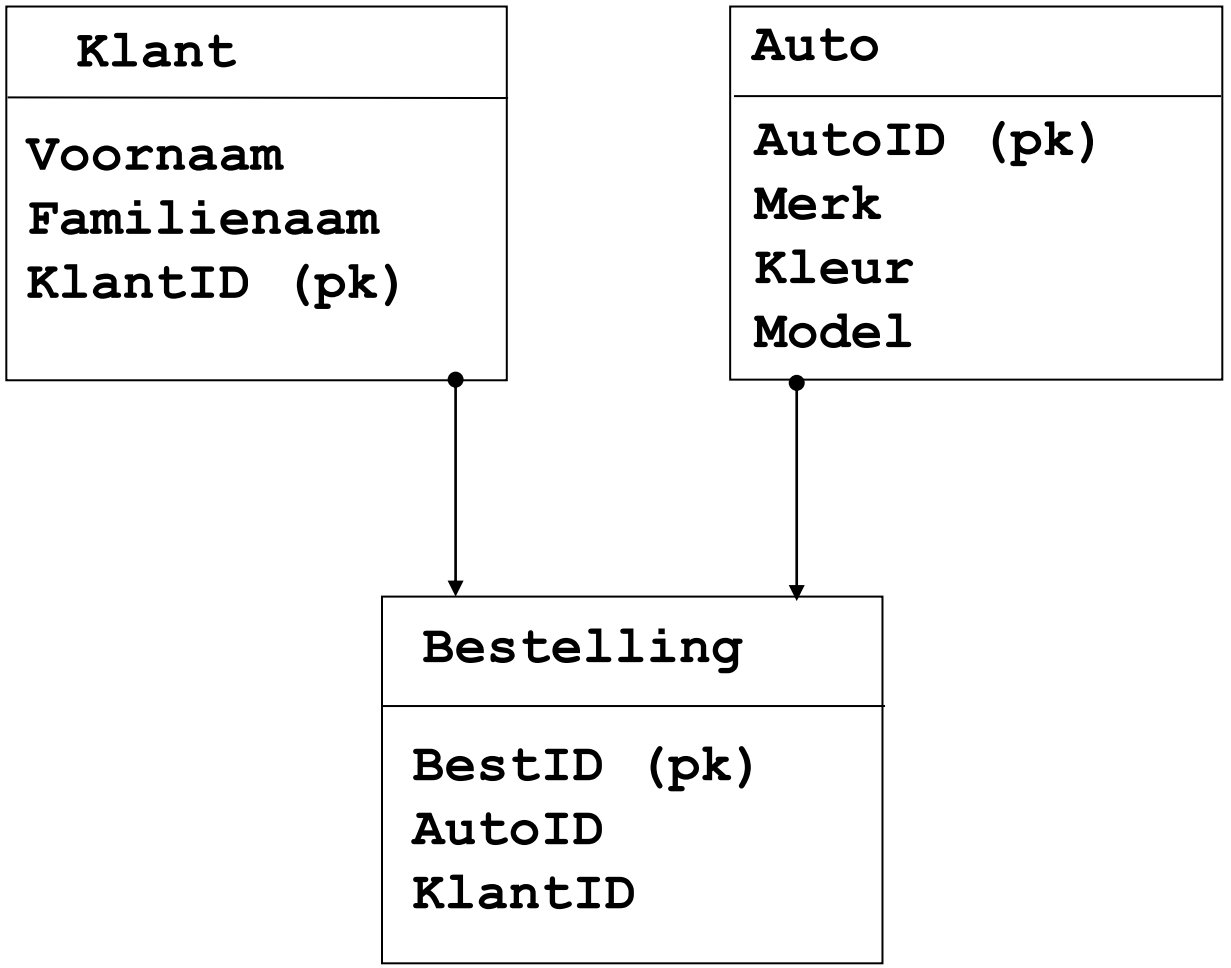
- binaire serialisatie
- XML-serialisatie

```
DataTableReader dtReader = wnTable.CreateDataReader();

while (dtReader.Read())
{
    for (int i = 0; i < dtReader.FieldCount; i++)
    //FieldCount geeft het aantal kolommen terug
    {
        Console.Write(dtReader[i] + " ");
    }
    Console.WriteLine();
}
dtReader.Close();
```

- **`DataSet` ➔ voorstelling van relationele database**
- **Bevat één of meerdere `DataTable` objecten met onderlinge relaties**

Bv:



Relationele dataset aanmaken (in geheugen)

```
// Data tabellen


DataTable autosTable =
                new DataTable("Auto");
DataTable klantenTable =
                new DataTable("Klant");
DataTable bestellingenTable =
                new DataTable("Bestelling");
```

```
//  Data Set
DataSet autosDataSet =
                    new DataSet("AutosDataSet");
// tabellen toevoegen aan dataset
autosDataSet.Tables.Add(bestellingenTable);
autosDataSet.Tables.Add(klantenTable);
autosDataSet.Tables.Add(autosTable);
```

Lezen/Schrijven van/naar XML vanuit een DataSet object

```
autosDataSet.WriteXml("autos.xml");

autosDataSet.WriteXml("autos.xml",
                      XmlWriteMode.WriteSchema);


autosDataSet.ReadXml("autos.xml");

autosDataSet.ReadXml("autos.xml",
                      XmlReadMode.ReadSchema);
```

- **Methoden van `DataSet`**

```
.AcceptChanges()
.Clear()
.Clone()
.Copy()
.GetChanges()
.HasChanges()
.Merge()
.RejectChanges()
```

Grid

Relaties tussen tabellen - `DataRelation` klasse

```
DataRelation dr1 = new
      DataRelation("KlantenBestelling",

// parent
         autosDataSet.Tables["Klant"].
                          Columns["KlantID"],

// child
        autosDataSet.Tables["Bestelling"].
                         Columns["BestID"]);

autosDataSet.Relations.Add(dr1); // Exception

dr1 = new DataRelation("AutosBestelling",
autosDataSet.Tables["Auto"].
                           Columns["AutoID"],


autosDataSet.Tables["Bestelling"].

                                  Columns["AutoID"]);

autosDataSet.Relations.Add(dr1);
```

Rij toevoegen aan tabel klant in autosDataSet

Eerste rij van tabel Klant in autosDataSet ophalen

```
DataRow drKlant =
autosDataSet.Tables["Klant"].
                                Rows[0];

Console.WriteLine( "Klant Naam: " + drKlant["Voornaam"]);
```

```
// navigeer van klanten tabel naar
//bestelling tabel

DataRow[] drsBestellingen = drKlant.GetChildRows

(autosDataSet.Relations["KlantenBestelling"]);


// haal de bestellingen van klant op


if(drsBestellingen != null)

     foreach(DataRow r in drsBestellingen )

          Console.WriteLine("Bestelling Id: "
               + r["BestID"]);
```

Constraints: Unique, Foreign Key

```
try
{
  wnTable.Rows.Add(dr);
}
catch (ConstraintException ce)
{
   Console.WriteLine(ce.Message );
}
```

Werken met DataAdapter

- Aanmaken SqlConnection object
- Aanmaken SqlDataAdapter object
- Dataset object creëren
- .Fill() methode op SqlDataAdapter object aanroepen om Dataset object op de vullen met gegevens uit data source

DataAdapters

- ‘Pijplijn’ tussen DataSets en data sources
- Niet ontworpen voor performantie, maar voor functionaliteit
- Is standaard gedisconnecteerd van datasource
- ondersteunt select, insert, delete, update commands en methoden

DataAdapters

- **Moeten steeds een ‘select’ command specifiëren**
- **Alle andere commands kunnen worden gegenereerd of gespecifieerd**

SQL DataAdapter aanmaken

- `using System.Data.SqlClient;`

```
string sConnectionString =
  "Initial Catalog= BierenDb;
   Data Source=localhost;
   Integrated Security=SSPI;";

SqlDataAdapter sqlAdp= new
SqlDataAdapter(sConnectionString);

sqlAdp.Close();
```

DataAdapter gebruiken

```
SQLDataAdapter sqlDA =
  new SqlDataAdapter();

sqlDA.SelectCommand =
  new SqlCommand ("select * from
  bieren“, sqlConnection);

DataSet sqlDS = new DataSet("
  bierentabel");
sqlDA.Fill(sqlDS, "bierentabel");
```

DataAdapters

- **Je kan je eigen InsertCommand, UpdateCommand en DeleteCommand specifiëren**
- **Roep GetChanges aan om de updates, adds en deletes door te voeren op de database sinds de laatste synchronisatie. En synchroniseer dan elk type.**

DataTables

- **Een DataSet één of meer DataTables.**
- **Kolommen (Fields) en rijen met gegevens worden bijgehouden in de DataTable.**
- **In DataColumns en DataRows objecten**

DataTables gebruiken

Via een DataTable kunnen we gegevens

- **Toevoegen, Wijzigen en verwijderen**
- **Opzoeken**
- **Views toepassen**
- **vergelijken**
- **Volledige tabel leegmaken**
- **Klonen en kopiëren**

DataRelaties

- **Tussen DataTables in een DataSet kunnen relaties gedefinieerd worden, (PK-FK's) met 1-1 en 1-many relaties, met referentiële integriteit**
- **Ook ondersteuning van cascading updates en deletes.**

DataViews

- **Zoals SQL view**
- **Kan view zijn over één of meerdere tabellen**
- **Kan gebruikt worden in GUI applicaties via Data Binding.**

Referenties

- **ADO.Net Programmer's Reference**
  Bilbija, Dickenson et al.
  Wrox Press