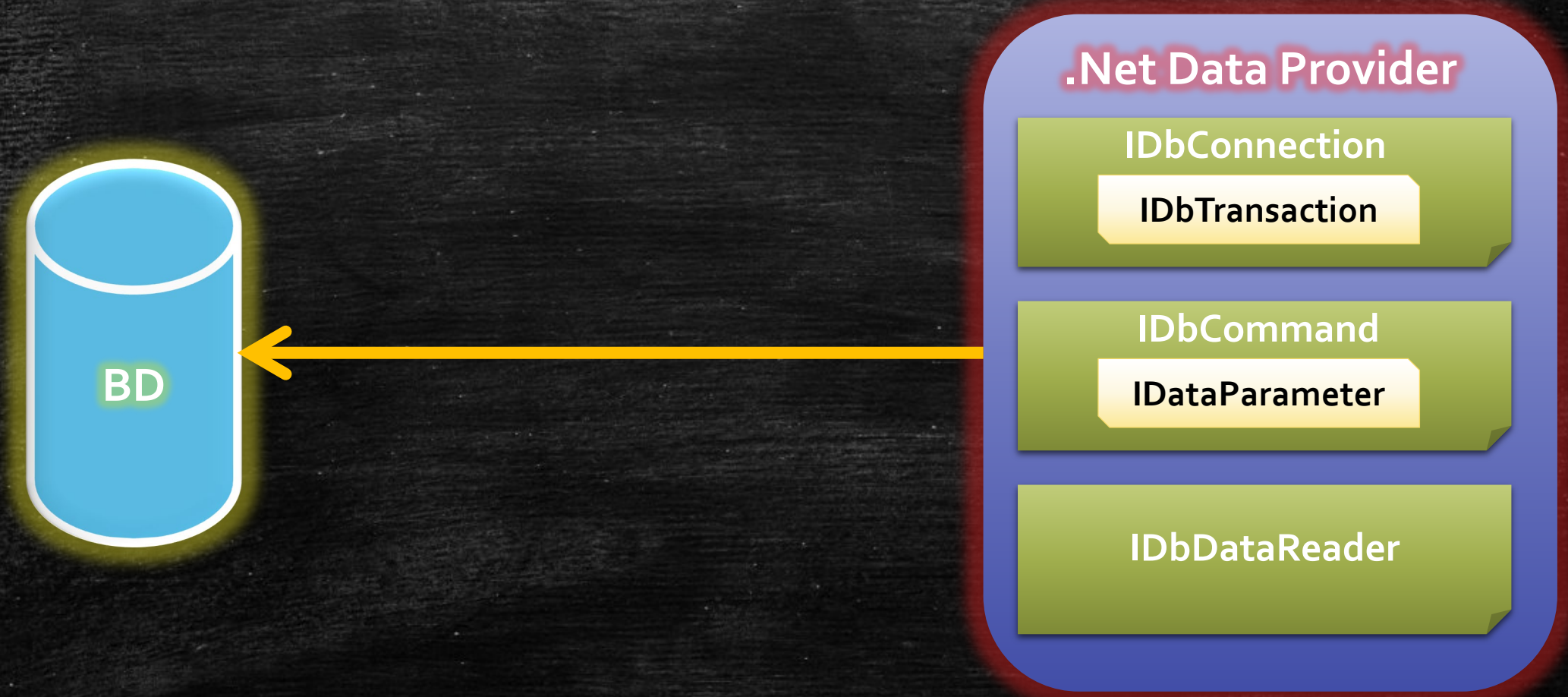
ADO.NET

Programmation .Net avec C#

Présentation

- ADO.NET est la version managée de ADO.
  - Un ensemble de classes qui exposent les services d'accès aux données.
  - Permet d'accéder à des données relationnelles et XML.
- Supporte deux scénarios d'accès :
  - Accès connecté par curseur.
  - Accès déconnecté via une base de données en mémoire.
- Une offre de connecteurs en standard :
  - Accès OLEDB : System.Data.OleDb.
  - Accès SQL Server : System.Data.SqlClient.
  - Accès Oracle : System.Data.OracleClient.
  - Accès ODBC : System.Data.Odbc.

Accès connecté



Accès connecté

Interfaces	SQL Server	Oracle	OLEDB	ODBC
System.Data.DB	**System.Data.SqlClient**	**System.Data.OracleClient**	**System.Data.OleDb**	**System.Data.Odbc**
IDbConnection	SqlConnection	OracleConnection	OleDbConnection	OdbcConnection
IDbTransaction	SqlTransaction	OracleTransaction	OleDbTransaction	OdbcTransaction
IDbCommand	SqlCommand	OracleCommand	OleDbCommand	OdbcCommand
IDbParameter	SqlParameter	OracleParameter	OleDbParameter	OdbcParameter
IDbDataReader	SqlDataReader	OracleDataReader	OleDbDataReader	OdbcDataReader

Objet Connection

- Permet la connexion à une source de donnée
- Principales propriétés :
  - ConnectionString :
    - Permet de spécifier / récupérer une chaine de connexion à la base.
  - ConnectionTimeout :
    - Permet de spécifier / récupérer le temps à attendre pour les tentatives de connexion (avant erreur).
- Principales méthodes :
  - Open :
    - Permet l'ouverture de la connexion.
  - Close :
    - Permet la fermeture de la connexion.

Objet Connection

```
SqlConnection conSQL = new SqlConnection();
conSQL.ConnectionString = @"Data Source=.\SQLEXPRESS;
                            AttachDbFilename=|DataDirectory|\Scolarisation.mdf;
                            Integrated Security=True;
                            User Instance=True";
conSQL.Open();
```

Exemple d'ouverture d'une « SqlConnection »

Objet Command

- Permet l'exécution des requêtes SQL ou procédures stockées.
- Création d'une commande :
  - Utilisation du constructeur de l'objet « Command ». (Avec la connexion).
  - Utilisation de la méthode « CreateCommand » de l'objet « Connection ».
- Principale propriétés :
  - Connection :
    - Connexion à la base.
  - CommandText :
    - Commande SQL à exécuter.
  - CommandType :
    - Type de commande SQL à exécuter (Text, StoredProcedure).
  - CommandTimeout :
    - Temps d'attente lors de l'exécution de la commande avant la génération d'une erreur.

Objet Command

- Principales méthodes :
  - ExecuteReader :
    - Retourne un DataReader.
  - ExecuteScalar :
    - Retourne une valeur unique.
  - ExecuteNonQuery :
    - Pour les commandes qui ne renvoient rien.
  - ExecuteXMLReader :
    - Retourne un XMLReader.

Objet Command

```
SqlCommand comSQL = new SqlCommand();
comSQL.Connection = conSQL;
comSQL.CommandType = System.Data.CommandType.Text;
comSQL.CommandText = "Select Count(*) from Etudiant";
Console.WriteLine(comSQL.ExecuteScalar());
```

Exemple d'exécution d'une requête SQL.

Objet DataReader

- Permet l'accès au données en lecture seule en mode déconnecté.
  - Accès séquentiel du début à la fin et uniquement dans ce sens.
- Création d'un « DataReader » :
  - Instanciation avec la méthode « ExecuteReader » de l'objet « Command ».
- Principales propriétés :
  - Item :
    - Retourne la valeur d'une colonne.
  - FieldCount :
    - Retourne le nombre de colonne dans la ligne courante.
  - IsClosed :
    - Indique si le « DataReader » est fermé.
  - RecordsAffected :
    - Retourne le nombre de colonnes modifiées, insérées ou supprimées par l'exécution d'une commande.

Objet DataReader

- Principales méthodes :
  - Read :
    - Permet d'avancer au prochain enregistrement.
  - GetName :
    - Permet de connaitre le nom de la colonne passée en paramètre.
  - IsDBNull :
    - Permet de savoir si la valeur dans l'enregistrement courant de la colonne passée en paramètre ne contient aucune valeur.
  - GetOrdinal :
    - Permet de connaitre le numéro de colonne de la colonne passée en paramètre.
  - GetString, GetDecimal, GetDateTime, …
    - Permet de récupérer des données typées.

Objet DataReader

```
using (SqlCommand cmd = new SqlCommand())
{
    try
    {
        Connection.Open();
        cmd.Connection = Connection;
        cmd.CommandType = System.Data.CommandType.Text;
        cmd.CommandText = "Select * From Etudiant";

        using (SqlDataReader dr = cmd.ExecuteReader())
        {
            while (dr.Read())
            {
                Console.WriteLine(dr["Name"]);
            }
        }
    }
    catch (Exception)
    {
        Console.WriteLine("Erreur de BD");
    }
    finally
    {
        Connection.Close();
    }
}
```

Exemple d'exécution d'une requête, et lecture de données avec « DataReader »

Objet DataReader

```
SqlCommand comSQL = new SqlCommand();
comSQL.Connection = conSQL;
comSQL.CommandType = System.Data.CommandType.StoredProcedure;
comSQL.CommandText = "SP_GET_STUDENTS_BY_GROUP";

SqlParameter paramSQL = new SqlParameter("@Group", System.Data.SqlDbType.Int);
paramSQL.Direction = System.Data.ParameterDirection.Input;
paramSQL.Value = 10;

comSQL.Parameters.Add(paramSQL);

SqlDataReader drSQL = comSQL.ExecuteReader();

while (drSQL.Read())
{
    Console.WriteLine(drSQL[0].ToString());
}

drSQL.Close();
```

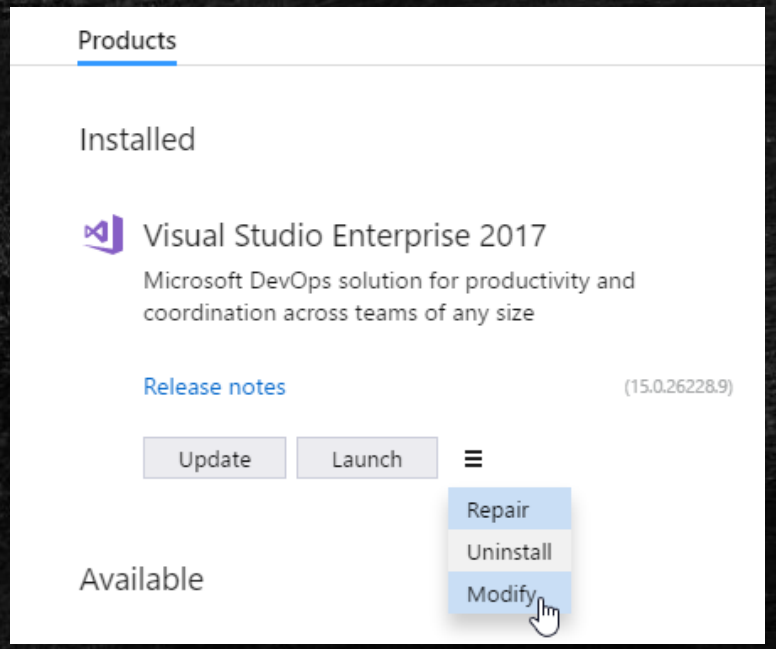
Exemple d'exécution d'une procédure stockée avec paramètres, et lecture de données avec « DataReader »

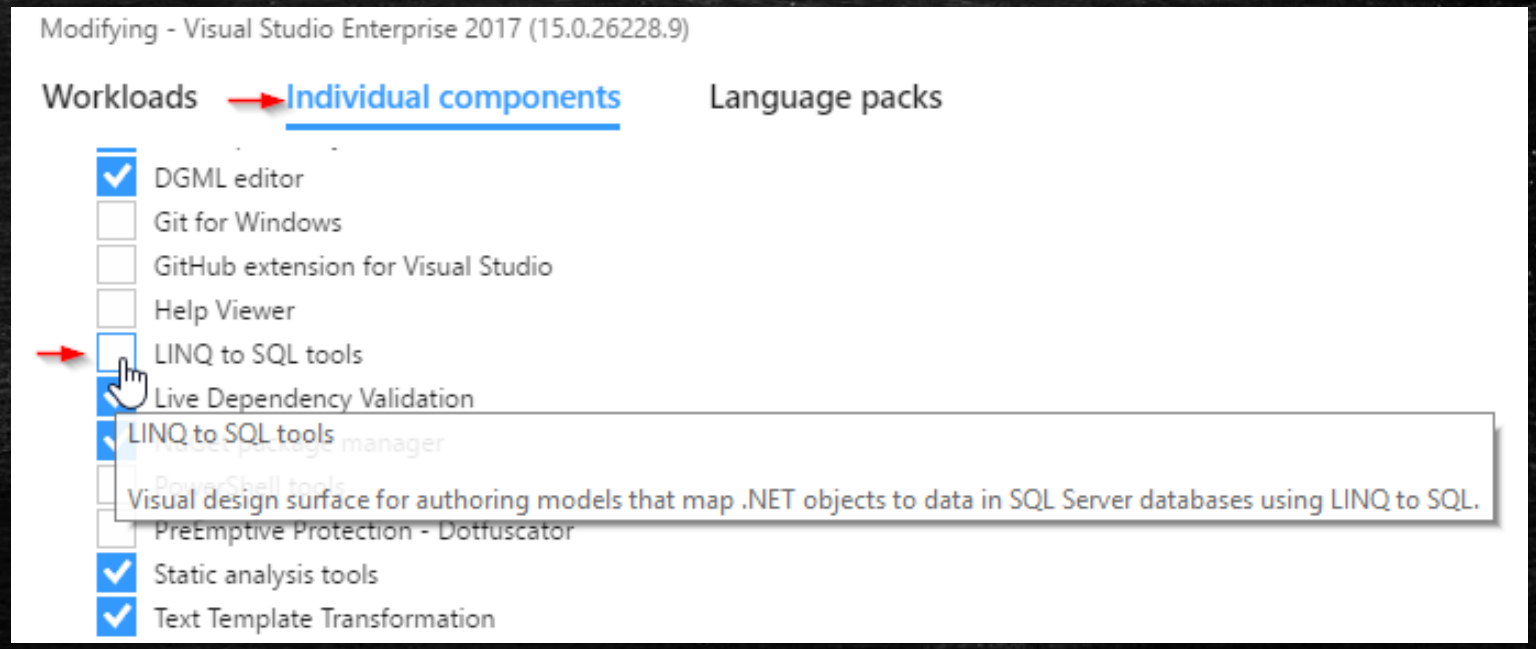
Linq To SQL

- LINQ to SQL est une implémentation de O/RM (object relational mapping) incluse dans le .NET Framework.
- Permet la modélisation d'une base de données relationnelle avec des classes .NET.
- Il est possible de récupérer des données d'une base en utilisant LINQ, mais également mettre à jour, insérer et supprimer des données dans celle-ci.
- Il prend totalement en charge les transactions, les vues et les procédures stockées
- Création d'un fichier « .dbml » qui contient un « DataContext ».
- Chaque objet est associé à une source (Table).
  - Les requêtes « CRUD » sont générées automatiquement au runtime.

Linq To SQL

- Linq To SQL n'est pas installé par défaut dans Visual Studio 2017 :
  - Ouvrir « Visual Studio Installer ».
  - Sélectionner l'option « Modifier ».
  - Dans l'onglet « Individual Components », cocher la case « LINQ To SQL tools ».
  - Cliquer sur « Modifier ».

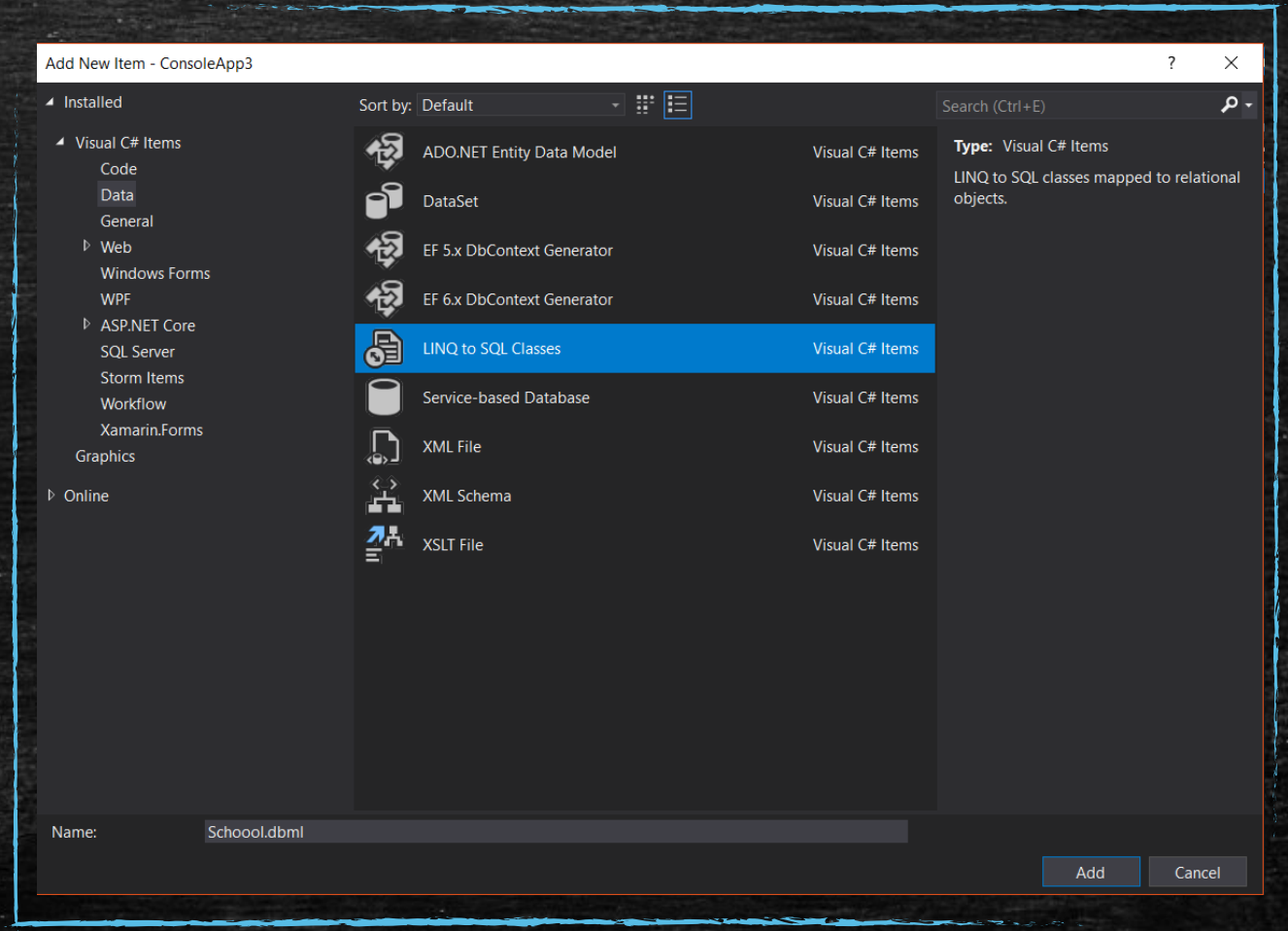




Linq To SQL

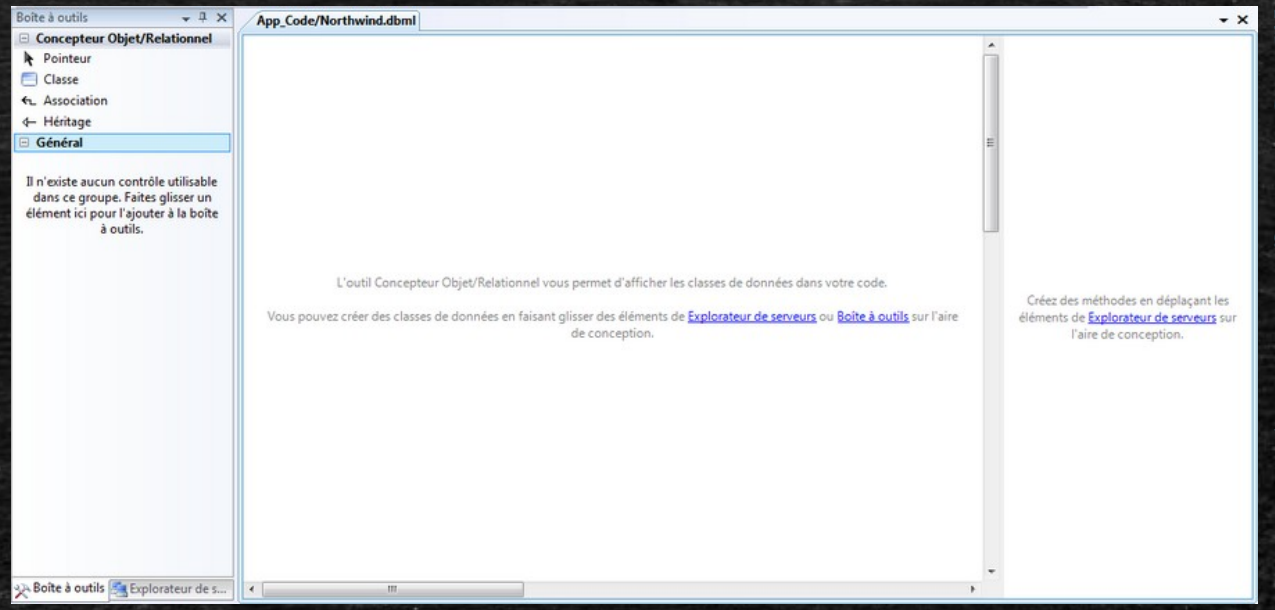
- Etapes d'utilisation de Linq To SQL :
  1. Création d'un nouveau modèle de données LINQ to SQL.
  2. Création des classes Entity depuis une base de données avec glisser/déposer sur l'aire de conception LINQ to SQL.
  3. Utilisation des requêtes Linq To SQL :
     1. Initialisation du « DataContext ».
     2. Utilisation des requêtes Linq To SQL :
        - Sélection.
        - Modification.
        - Insertion.
        - Suppression.
        - Pagination.

Linq To SQL



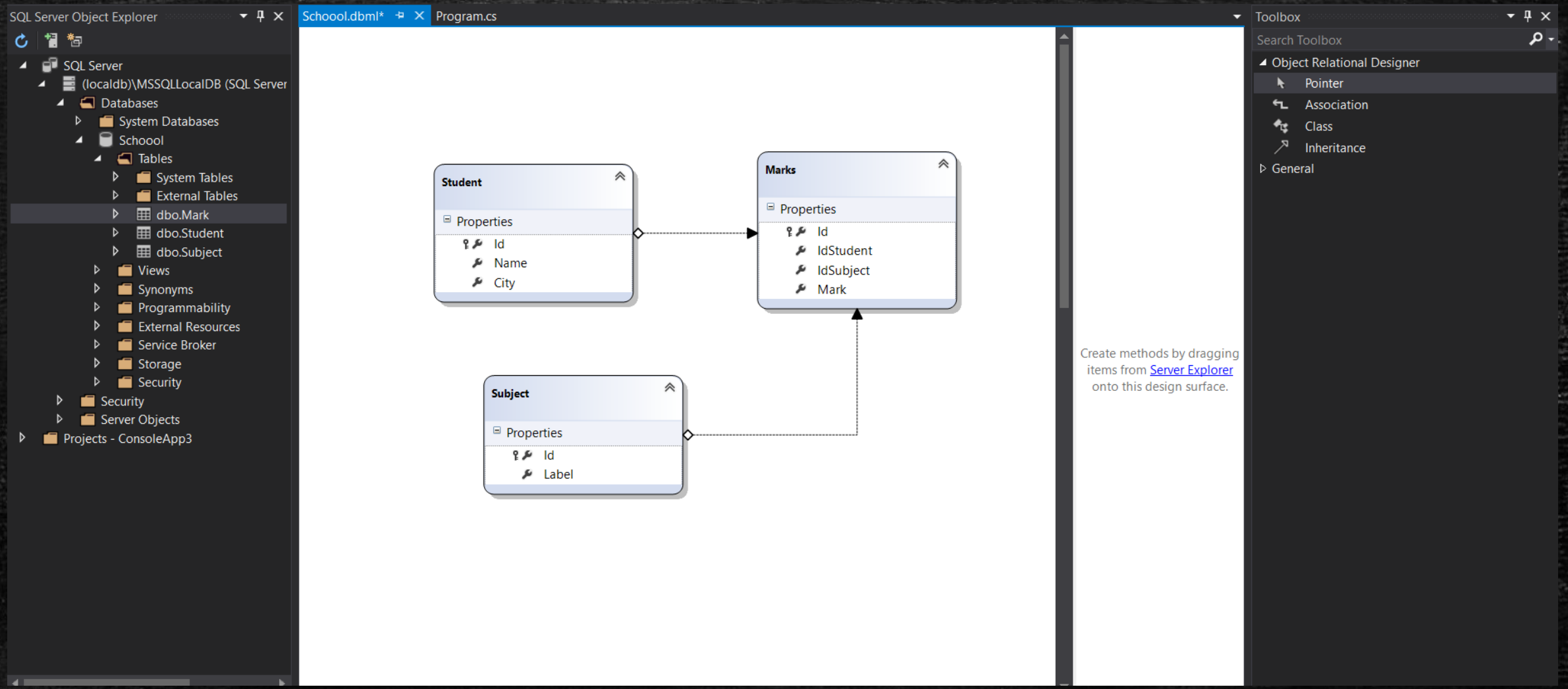
Création d'un nouveau modèle de données LINQ to SQL.

Linq To SQL



Exemple d'un fichier « .dbml » vide.

Linq To SQL


SQL Server Object Explorer
SQL Server
(localdb)\MSSQLLocalDB (SQL Server
Databases
System Databases
Schoool
Tables
System Tables
External Tables
dbo.Mark
dbo.Student
dbo.Subject
Views
Synonyms
Programmability
External Resources
Service Broker
Storage
Security
Security
Server Objects
Projects - ConsoleApp3
Schoool.dbml*
Program.cs
Student
Properties
Id
Name
City
Marks
Properties
Id
IdStudent
IdSubject
Mark
Subject
Properties
Id
Label
Create methods by dragging items from Server Explorer onto this design surface.
Toolbox
Search Toolbox
Object Relational Designer
Pointer
Association
Class
Inheritance
General

Linq To SQL

```
SchooolDataContext dtx = new SchooolDataContext();

var students = dtx.Students.Where(s => s.City == "Oujda").ToList();

students.ForEach(s => Console.WriteLine(s.Name));
```

Exemple de recherche/sélection avec Linq to SQL

Linq To SQL

```
SchooolDataContext dtx = new SchooolDataContext();

var students = dtx.Students.Where(s => s.Marks.Any(m => m.Mark < 10)).ToList();

students.ForEach(s => Console.WriteLine(s.Name));
```

Exemple de jointure avec Linq to SQL

Linq To SQL

```
SchooolDataContext dtx = new SchooolDataContext();

var student = dtx.Students.SingleOrDefault(s => s.Id == 3);
student.City = "Oujda";

dtx.SubmitChanges();
```

Exemple de modification avec Linq to SQL

Linq To SQL

```
SchooolDataContext dtx = new SchooolDataContext();

Subject subject = new Subject
{
    Label = "French"
};

dtx.Subjects.InsertOnSubmit(subject);
dtx.SubmitChanges();
```

Exemple de création d'une nouvelle matière avec Linq to SQL

Linq To SQL

```
SchooolDataContext dtx = new SchooolDataContext();

var subjects = dtx.Subjects.Where(s => s.Label.Contains("French"));

dtx.Subjects.DeleteAllOnSubmit(subjects);
dtx.SubmitChanges();
```

Exemple de suppression avec Linq to SQL