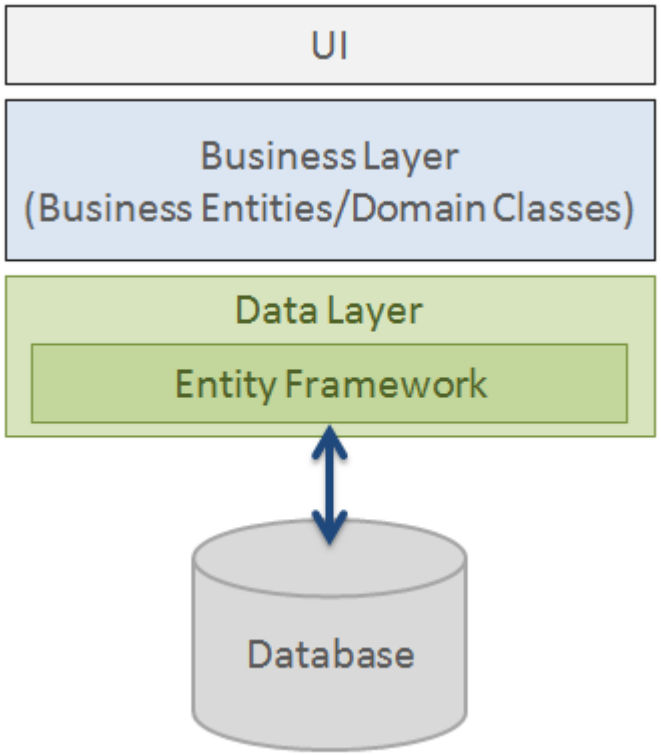
MISA®
TIN CẬY - TIỆN ÍCH - TẬN TÌNH

Entity Framework và ADO.NET

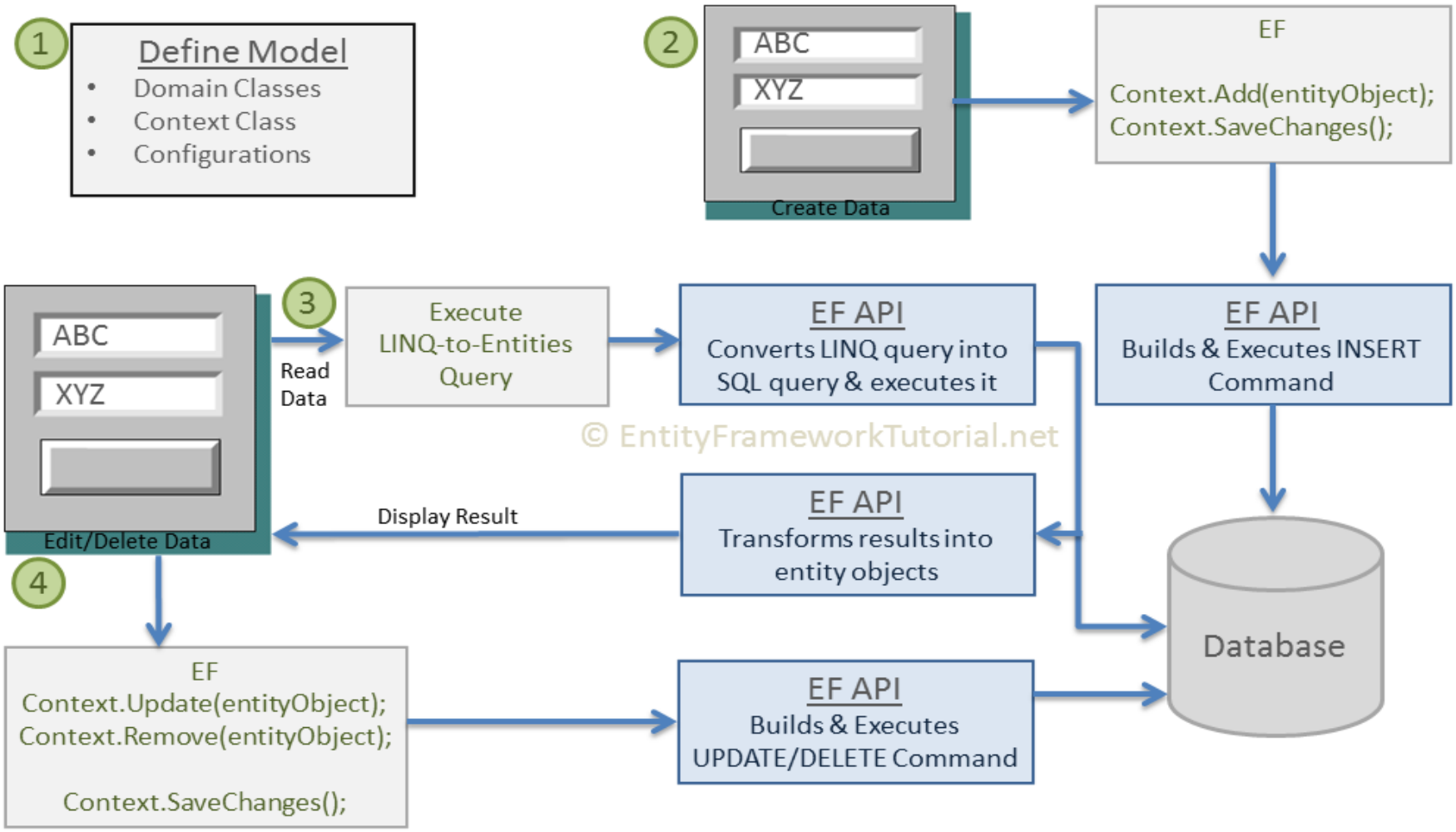
Nguyễn Văn Mạnh

Tổng quan về Entity Framework

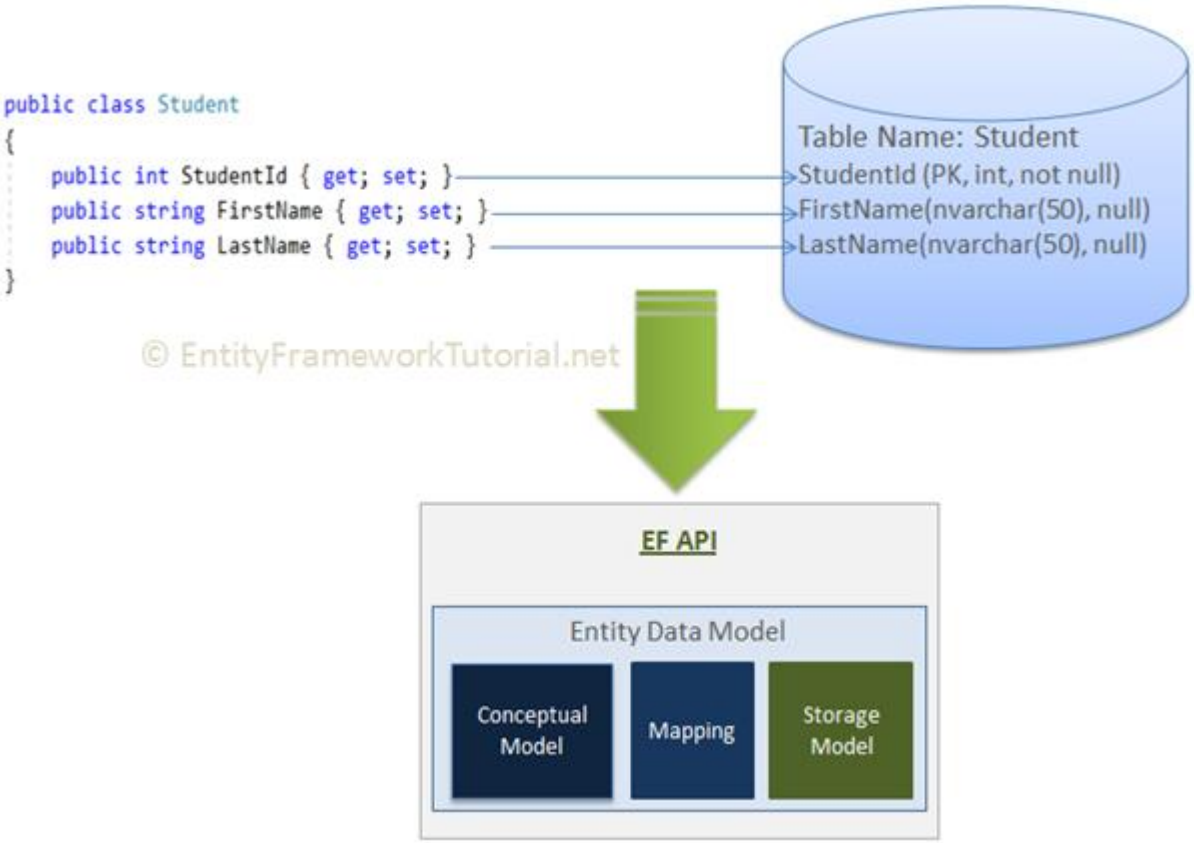
- Entity Framework là ORM framework mã nguồn mở bên cạnh đó là Linq to SQL
- EF được cung cấp bởi Microsoft
- EF cho phép ta có thể làm việc với dữ liệu thông qua đối tượng
- Phiên bản mới nhất là EF6 và EF Core



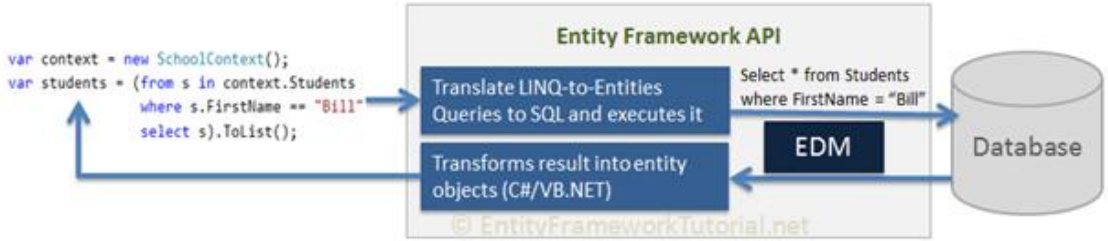
Luồng xử lý của Entity Framework



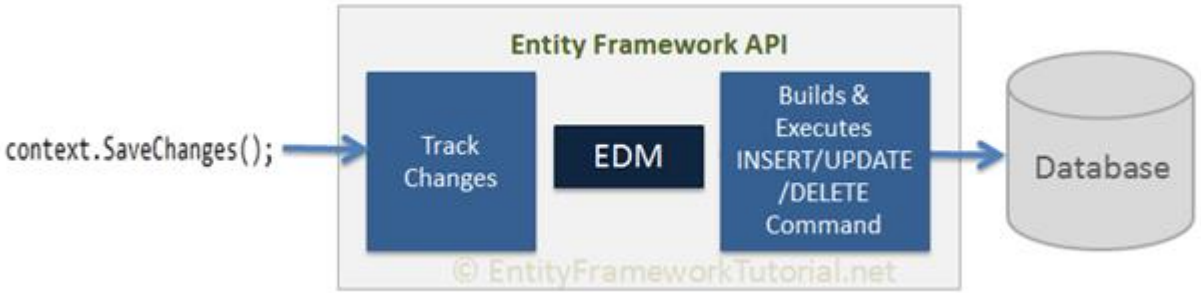
Entity Data Model (EDM)



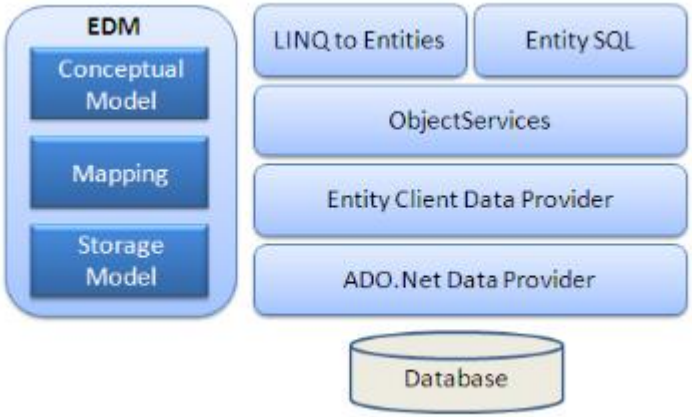
- Querying



- Saving



Kiến trúc tổng quan của Entity Framework

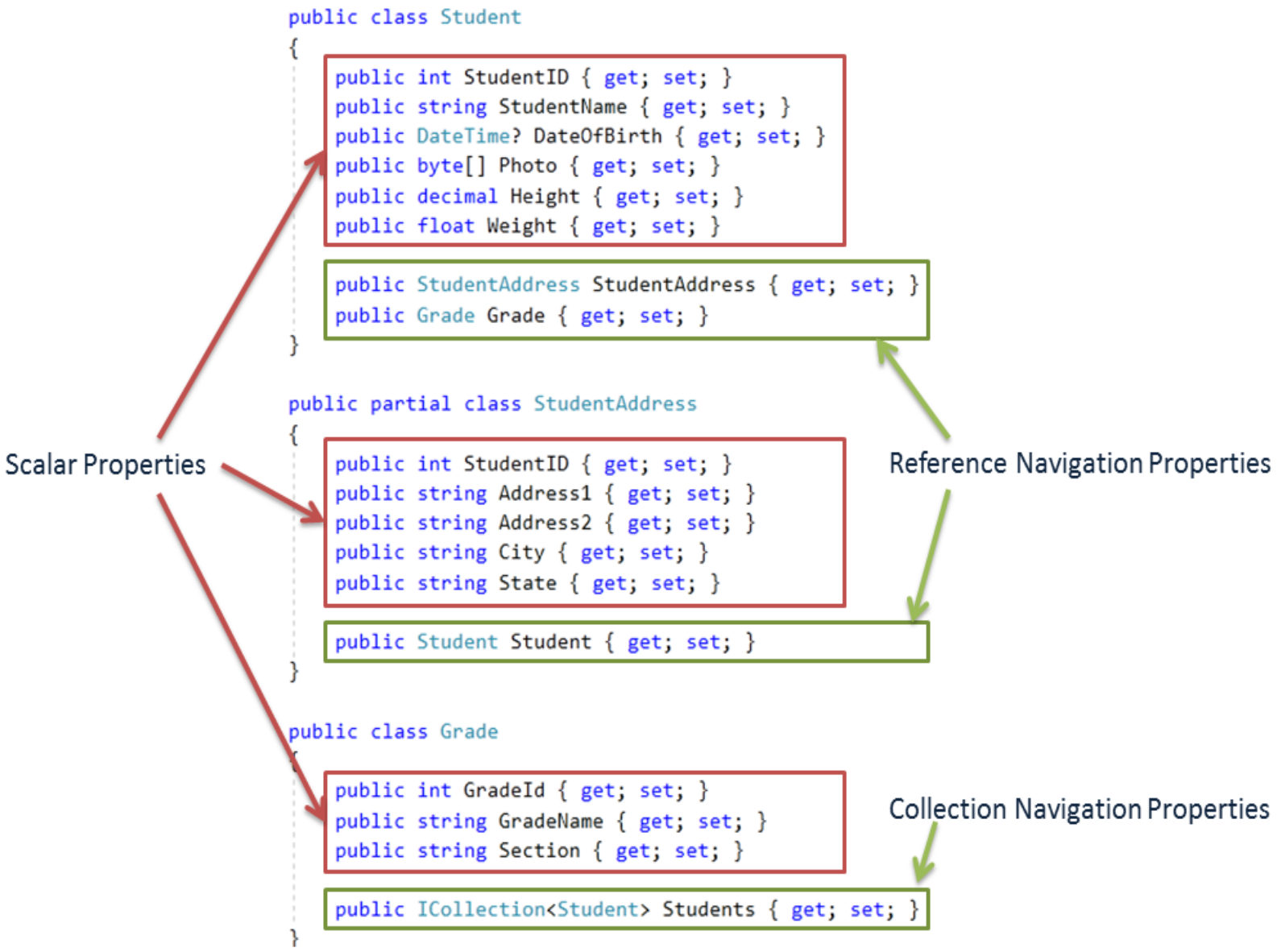


Context Class

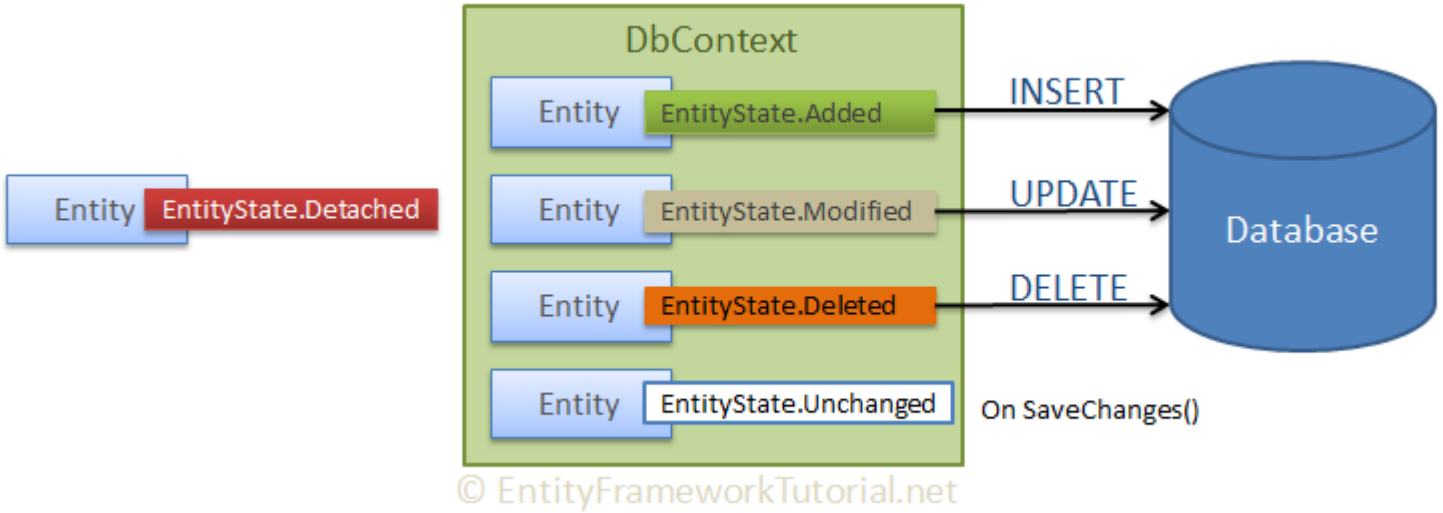
- Context class để query hoặc lưu trữ data, thực hiện savechanges()

```
public class SchoolContext : DbContext
{
    public SchoolContext()
    {

    }

    public DbSet<Student> Students { get; set; }
    public DbSet<StudentAddress> StudentAddresses { get; set; }
    public DbSet<Grade> Grades { get; set; }

}
```

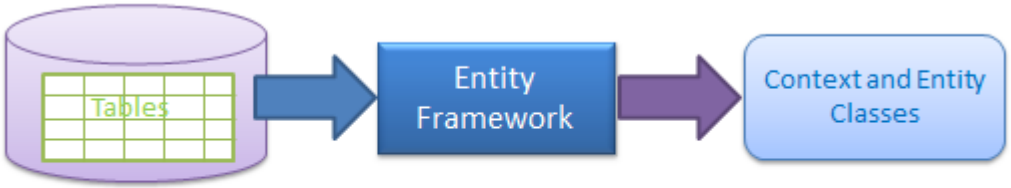
Entity Class



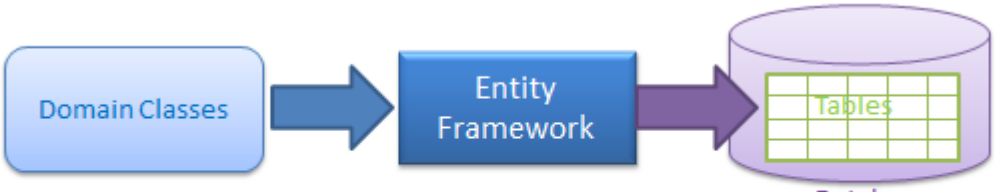
Entity State



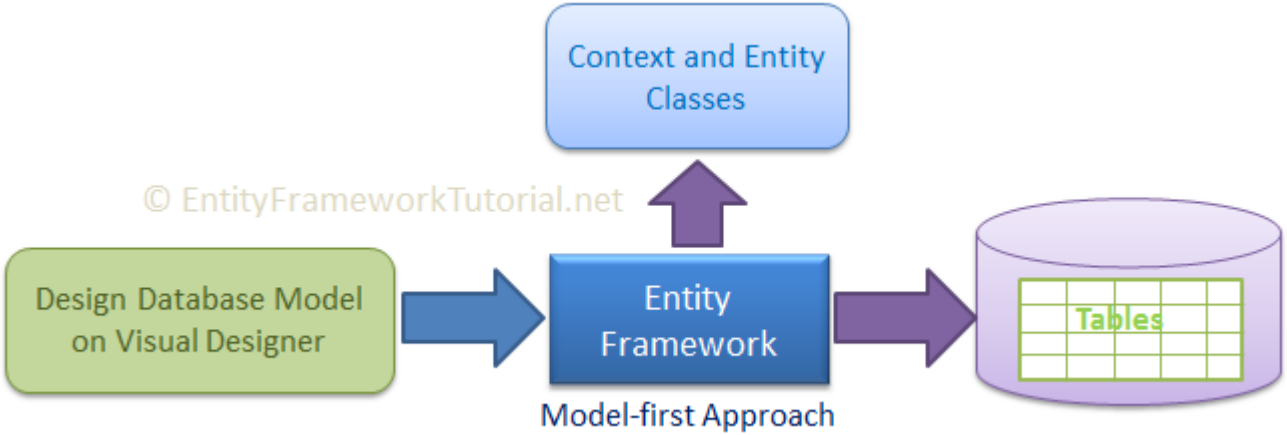
Cách tiếp cận Entity Framework



Database-First Approach

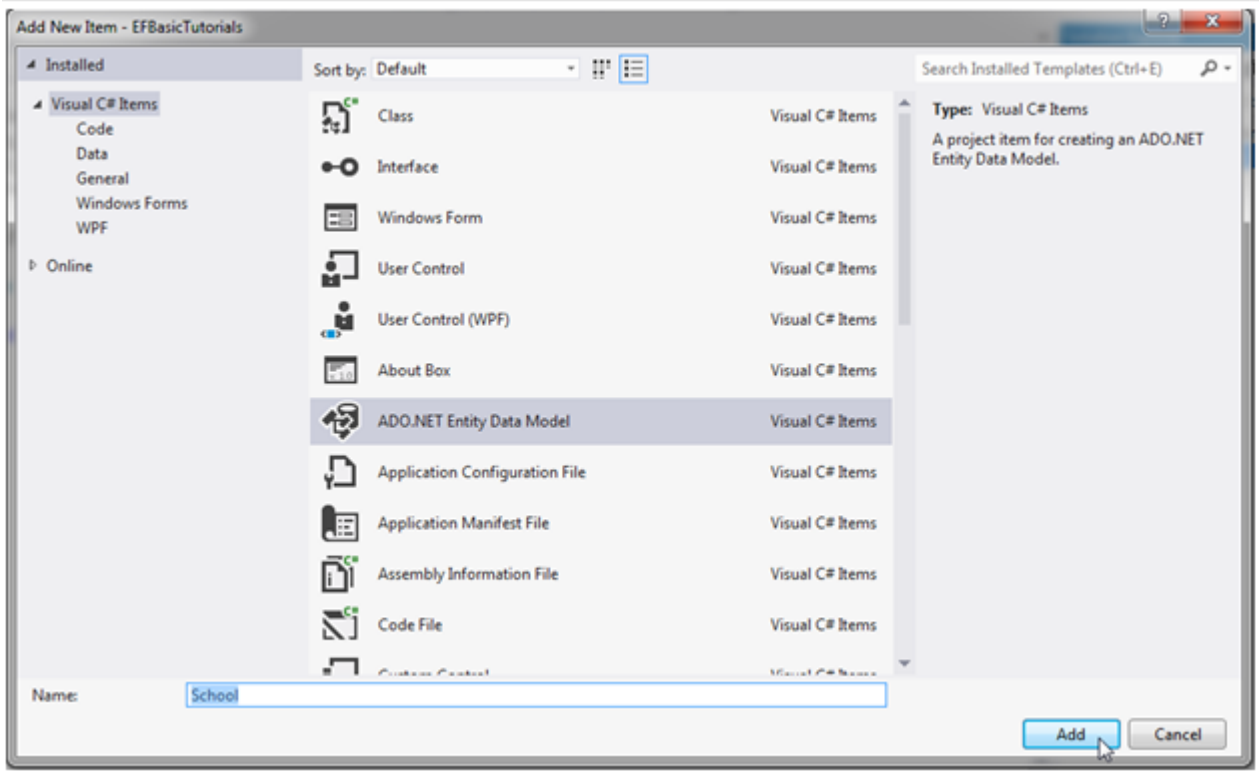


Code-First Approach

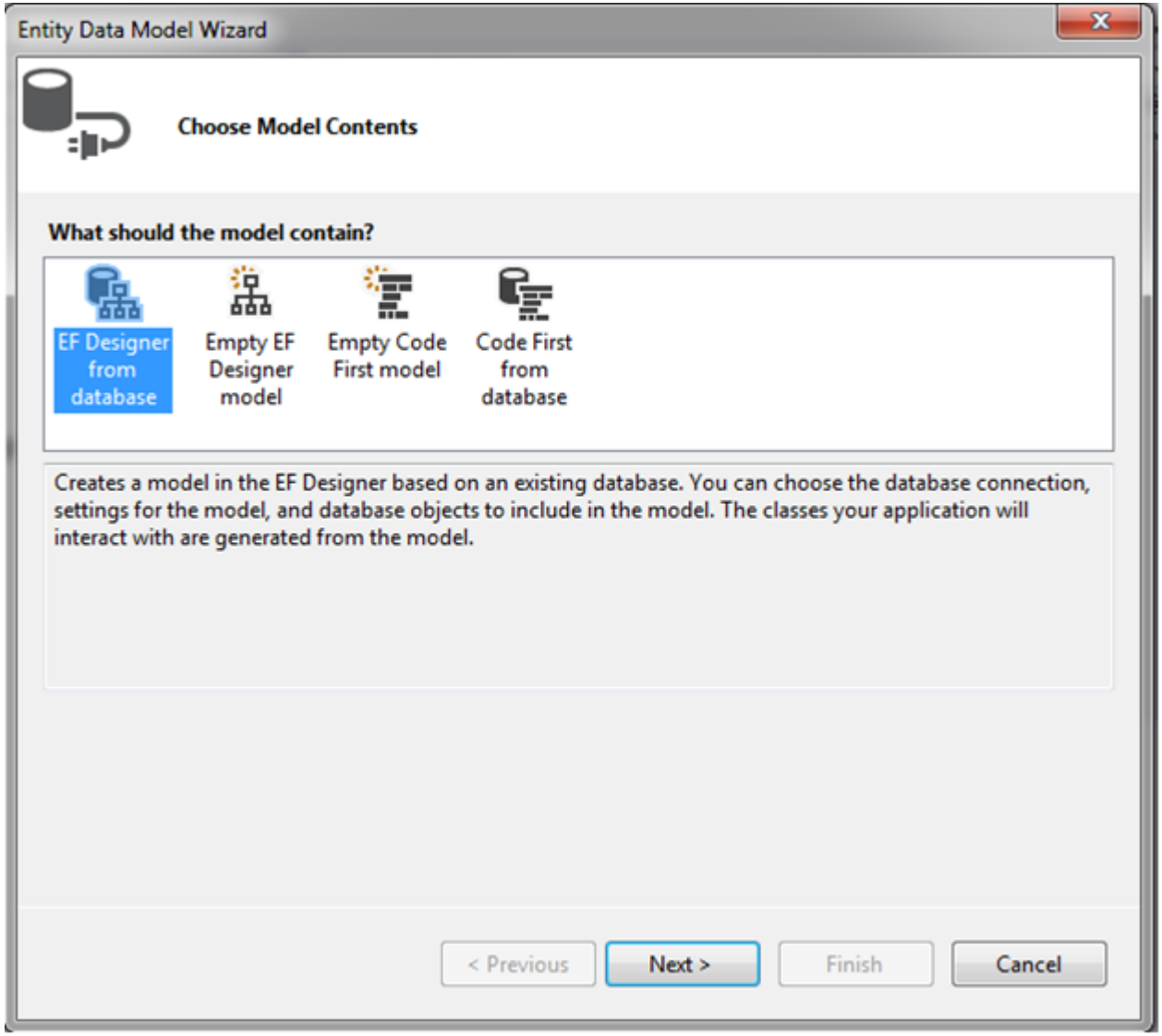


Model-first Approach

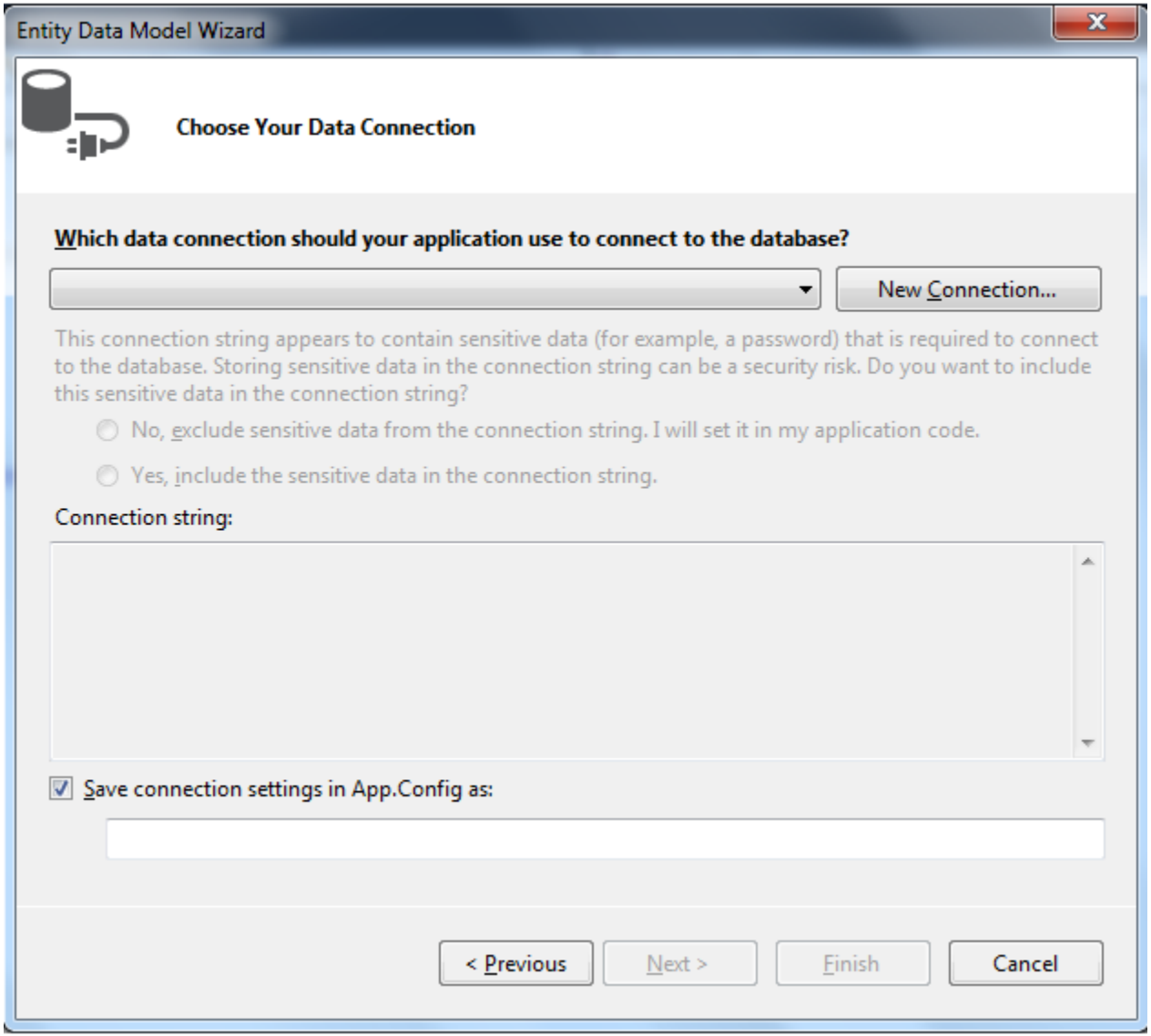
Tạo Entity Data Model Database First



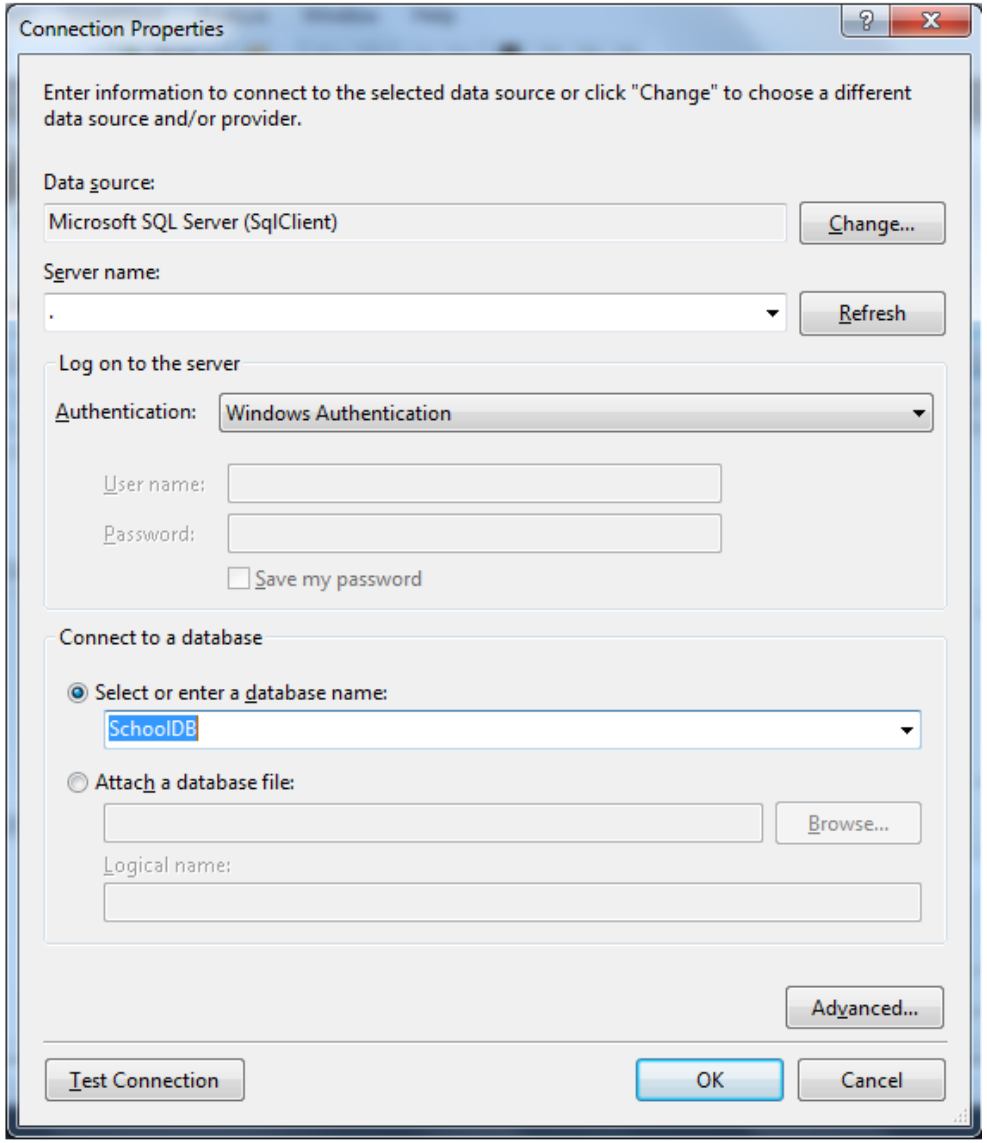
Tạo Entity Data Model Database First



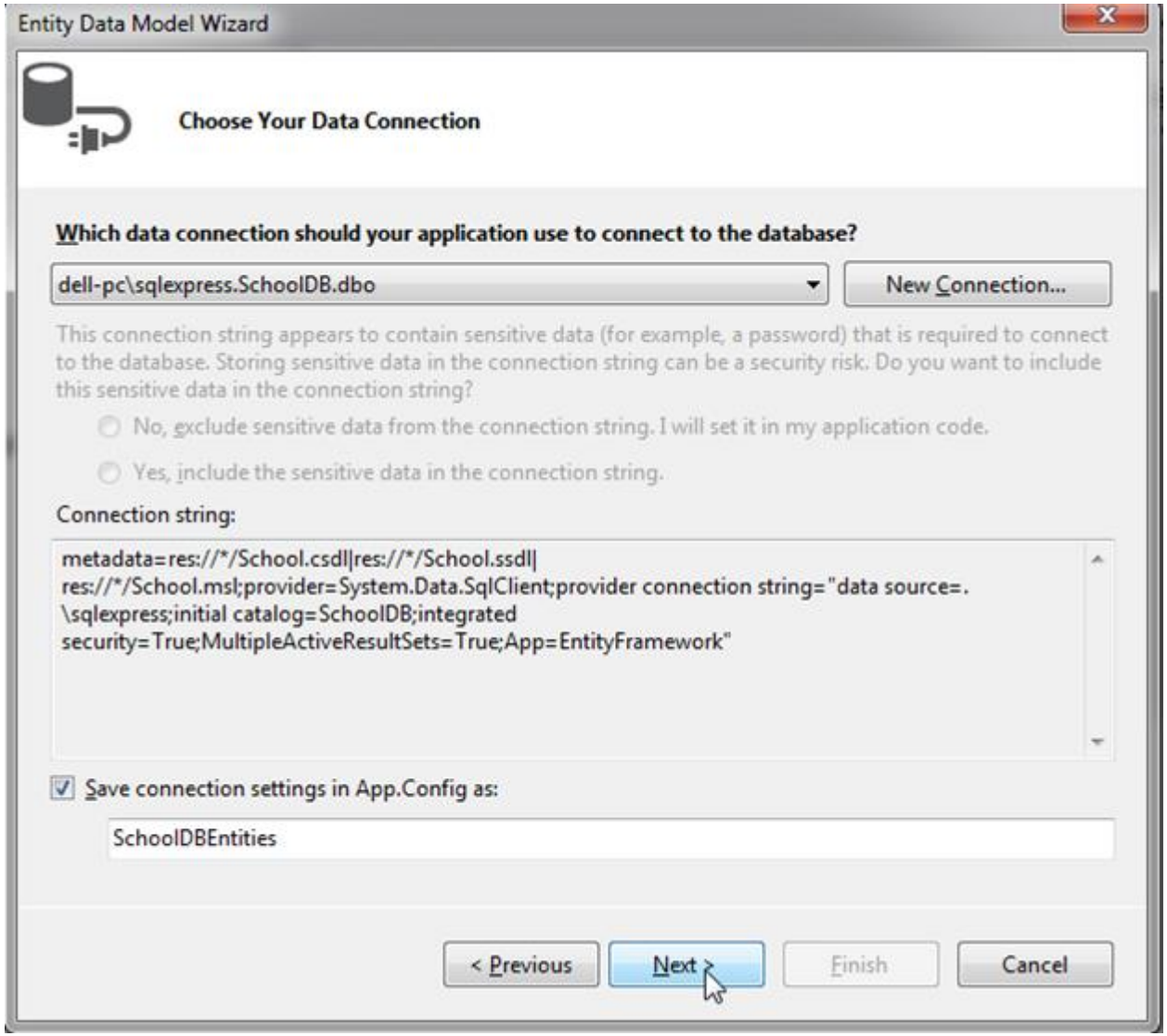
Tạo Entity Data Model Database First



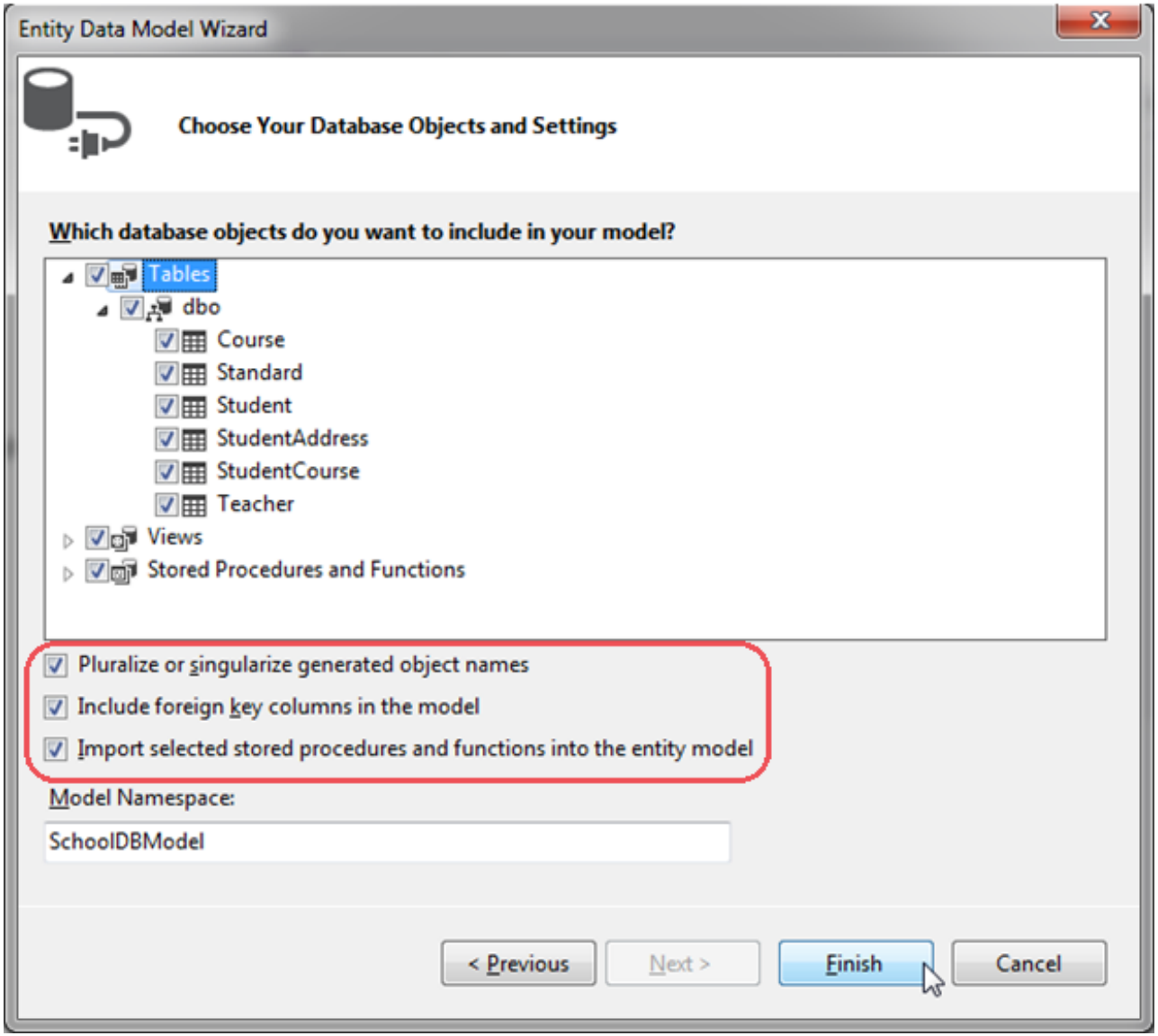
Tạo Entity Data Model Database First

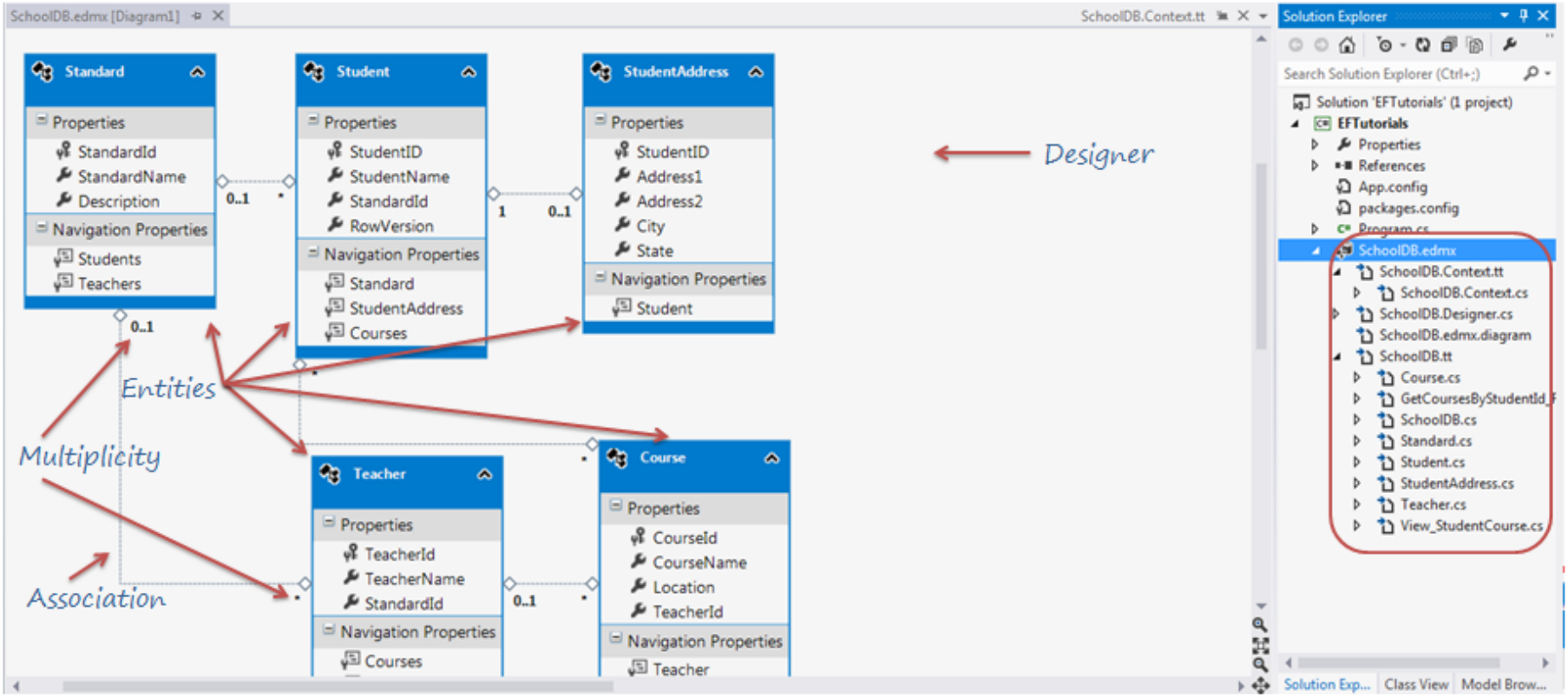


Tạo Entity Data Model Database First

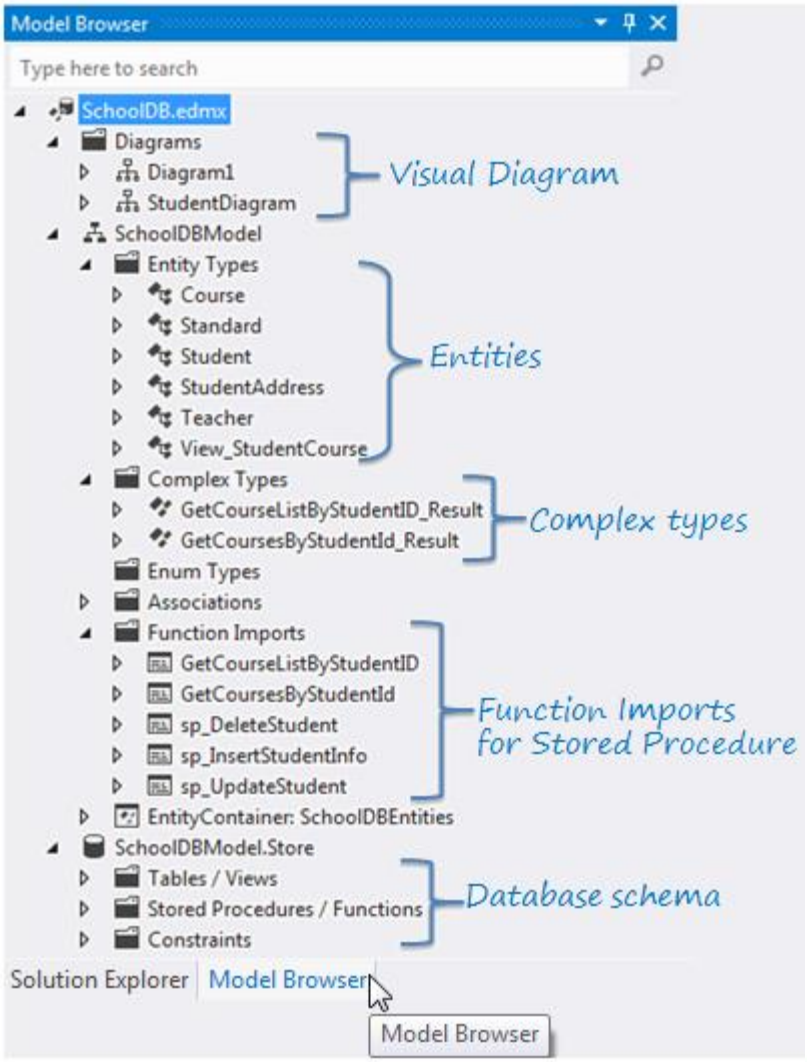


Tạo Entity Data Model Database First

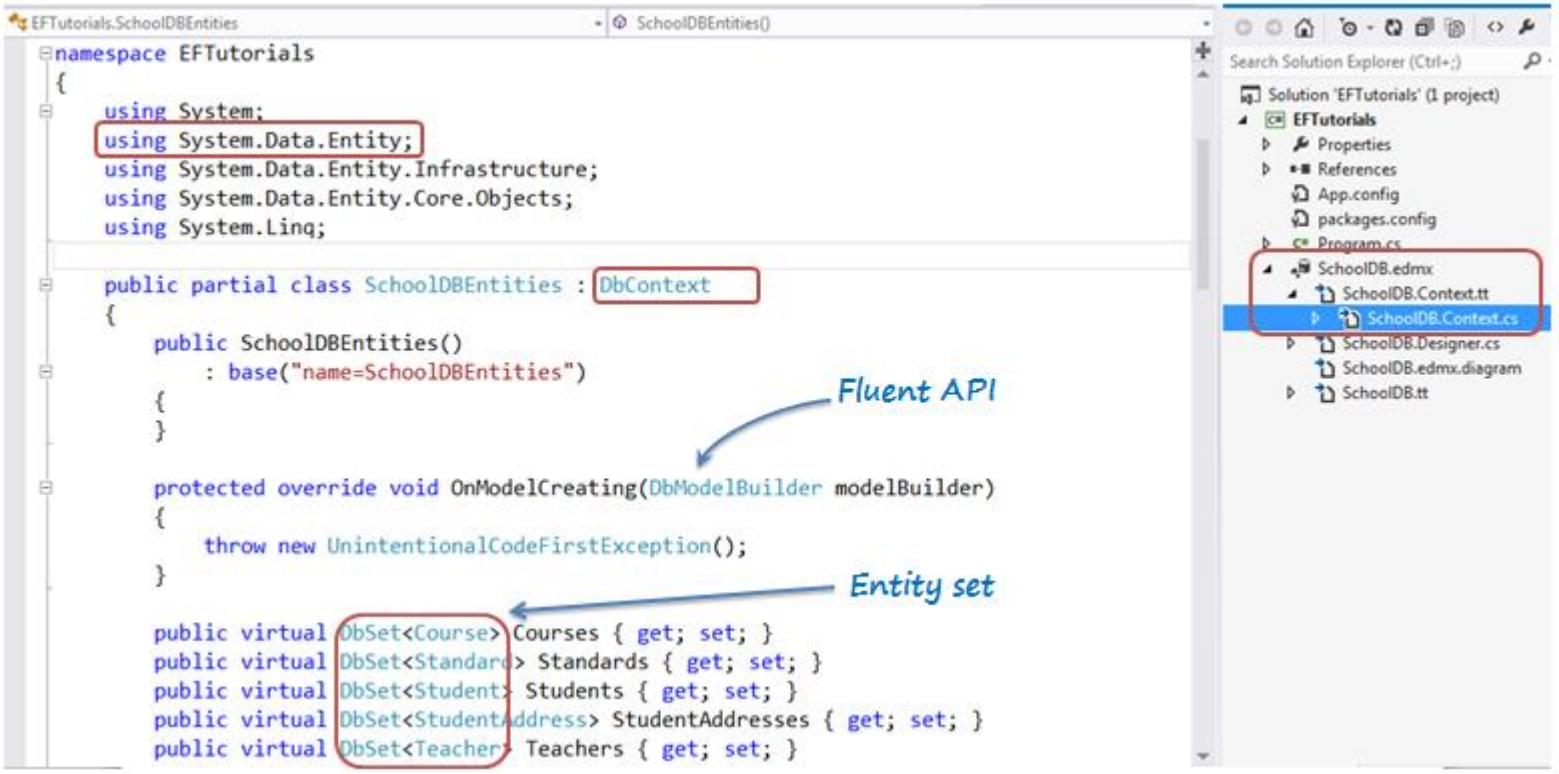



SchoolDB.edmx [Diagram1]
SchoolDB.Context.tt
Standard
Properties
StandardId
StandardName
Description
Navigation Properties
Students
Teachers
Student
Properties
StudentID
StudentName
StandardId
RowVersion
Navigation Properties
Standard
StudentAddress
Courses
StudentAddress
Properties
StudentID
Address1
Address2
City
State
Navigation Properties
Student
Teacher
Properties
TeacherId
TeacherName
StandardId
Navigation Properties
Courses
Course
Properties
CourseId
CourseName
Location
TeacherId
Navigation Properties
Teacher
0..1
1
Designer
Entities
Multiplicity
Association
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'EFTutorials' (1 project)
EFTutorials
Properties
References
App.config
packages.config
Program.cs
SchoolDB.edmx
SchoolDB.Context.tt
SchoolDB.Context.cs
SchoolDB.Designer.cs
SchoolDB.edmx.diagram
SchoolDB.tt
Course.cs
SchoolDB.cs
Standard.cs
Student.cs
StudentAddress.cs
Teacher.cs
View_StudentCourse.cs
Solution Exp...
Class View
Model Brow...

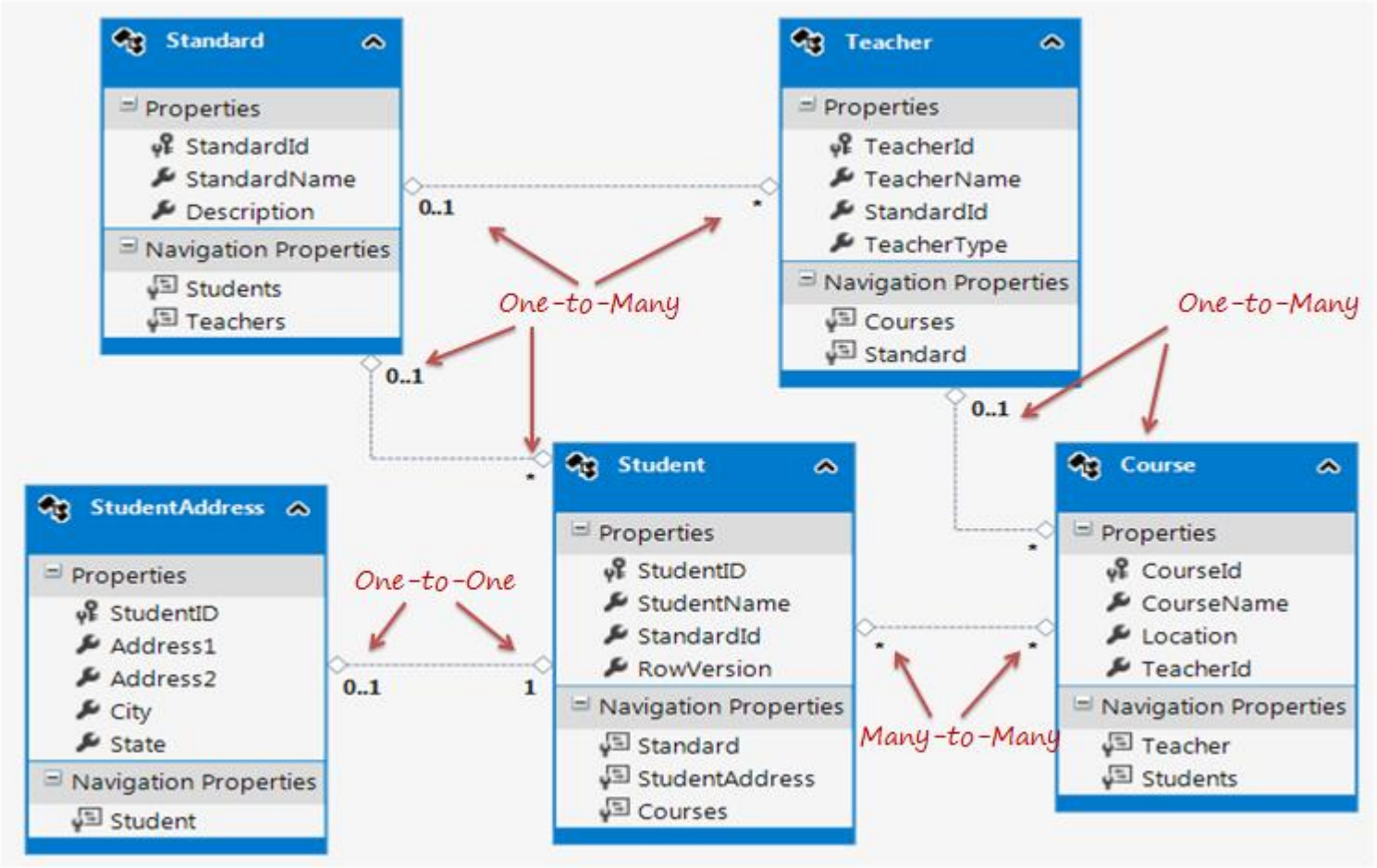
Model Browser



DbContext


EFTutorials.SchoolDBEntities
SchoolDBEntities()
namespace EFTutorials
{
using System;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Core.Objects;
using System.Linq;
public partial class SchoolDBEntities : DbContext
{
public SchoolDBEntities()
: base("name=SchoolDBEntities")
{
}
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
throw new UnintentionalCodeFirstException();
}
public virtual DbSet<Course> Courses { get; set; }
public virtual DbSet<Standard> Standards { get; set; }
public virtual DbSet<Student> Students { get; set; }
public virtual DbSet<StudentAddress> StudentAddresses { get; set; }
public virtual DbSet<Teacher> Teachers { get; set; }
Fluent API
Entity set
Search Solution Explorer (Ctrl+;)
Solution 'EFTutorials' (1 project)
EFTutorials
Properties
References
App.config
packages.config
Program.cs
SchoolDB.edmx
SchoolDB.Context.tt
SchoolDB.Context.cs
SchoolDB.Designer.cs
SchoolDB.edmx.diagram
SchoolDB.tt

Relationship



Query

- Linq to entities
  - Linq Method

```
//Querying with LINQ to Entities
using (var context = new SchoolDBEntities())
{
    var query = context.Students
                       .where(s => s.StudentName == "Bill")
                       .FirstOrDefault<Student>();
}
```

  - Linq Syntax

```
using (var context = new SchoolDBEntities())
{
    var query = from st in context.Students
                where st.StudentName == "Bill"
                select st;

    var student = query.FirstOrDefault<Student>();
}
```

Query

- Linq to entities
  - First()
  - FirstOrDefault()
  - Single()
  - SingleOrDefault()
  - ToList()
  - Count()

```
using (var ctx = new SchoolDBEntities())
{
    var student = (from s in ctx.Students
                   where s.StudentName == "Bill"
                   select s).FirstOrDefault<Student>();
}
```

```
var ctx = new SchoolDBEntities();
var student = ctx.Students.Find(1);
```

```
using (var ctx = new SchoolDBEntities())
{
        var students = ctx.Students.OrderBy(s => s.StudentName).ToList();
        // or descending order
        var  descStudents = ctx.Students.OrderByDescending(s => s.StudentName).ToList();
}
```

Query

- Entity SQL

```
//Querying with Object Services and Entity SQL
string sqlString = "SELECT VALUE st FROM SchoolDBEntities.Students " +
                   "AS st WHERE st.StudentName == 'Bill'";

var objctx = (ctx as IObjectContextAdapter).ObjectContext;

ObjectQuery<Student> student = objctx.CreateQuery<Student>(sqlString);
Student newStudent = student.First<Student>();
```

- Native SQL

```
using (var ctx = new SchoolDBEntities())
{
    var studentName = ctx.Students.SqlQuery("Select studentid, studentname, standardId from Student where studentname='Bill'").FirstOrDefault<Student>();
}
```

Eager Loading

- Là quá trình tải một loại thực thể thì cũng tải cho loại thực thể liên quan như một phần của truy vấn. Được sử dụng bằng hàm Include()

```
using (var ctx = new SchoolDBEntities())
{
    var stud1 = ctx.Students
                   .Include("Standard")
                   .Where(s => s.StudentName == "Bill")
                   .FirstOrDefault<Student>();
}
```



```
SELECT TOP (1)
[Extent1].[StudentID] AS [StudentID],
[Extent1].[StudentName] AS [StudentName],
[Extent2].[StandardId] AS [StandardId],
[Extent2].[StandardName] AS [StandardName],
[Extent2].[Description] AS [Description]
FROM  [dbo].[Student] AS [Extent1]
LEFT OUTER JOIN [dbo].[Standard] AS [Extent2] ON [Extent1].[StandardId] = [Extent2].[StandardId]
WHERE 'Bill' = [Extent1].[StudentName]
```

Lazy Loading

- Entity chỉ được load khi có lời gọi.

```
using (var ctx = new SchoolDBEntities())
{
    //Loading students only
    IList<Student> studList = ctx.Students.ToList<Student>();

    Student std = studList[0];

    //Loads Student address for particular Student only (seperate SQL query)
    StudentAddress add = std.StudentAddress;
}
```

- Để sử dụng lazyloading
  - Context.Configuration.ProxyCreationEnable = true
  - Context.Configuration.LazyLoadingEnable = true
  - Navigation property phải để public và virtual

Explicit Loading

- Load tường minh sử dụng hàm Load() để gọi các thực thể quan hệ

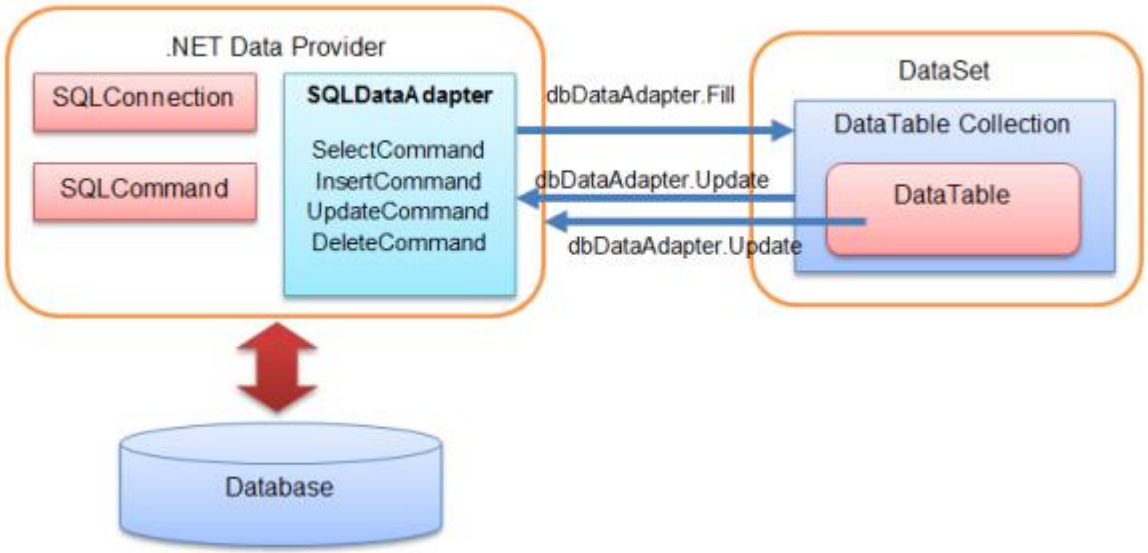
```
using (var context = new SchoolContext())
{
    var student = context.Students
                         .Where(s => s.FirstName == "Bill")
                         .FirstOrDefault<Student>();

    context.Entry(student).Reference(s => s.StudentAddress).Load(); // loads StudentAddress
    context.Entry(student).Collection(s => s.StudentCourses).Load(); // loads Courses collection
}
```

So sánh performance giữa các query

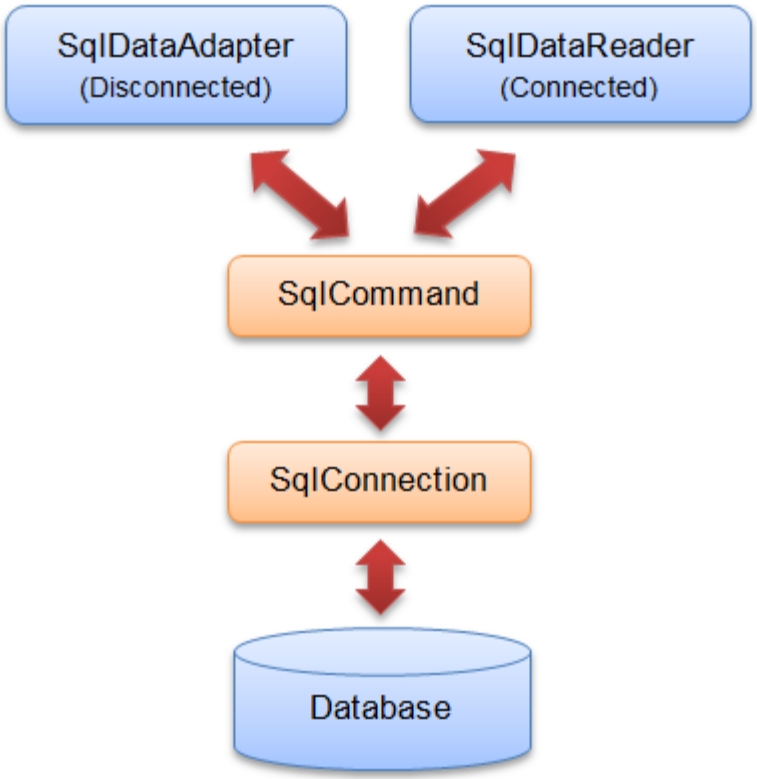
	Linq to entities	Entity SQL	Raw SQL
Lần 1	1,402,392,150 ms	82ms	7ms
Lần 2	1,402,392,206 ms	4ms	5ms
Lần 3	1,402,392,197 ms	4ms	5ms

Giới thiệu về ADO.NET

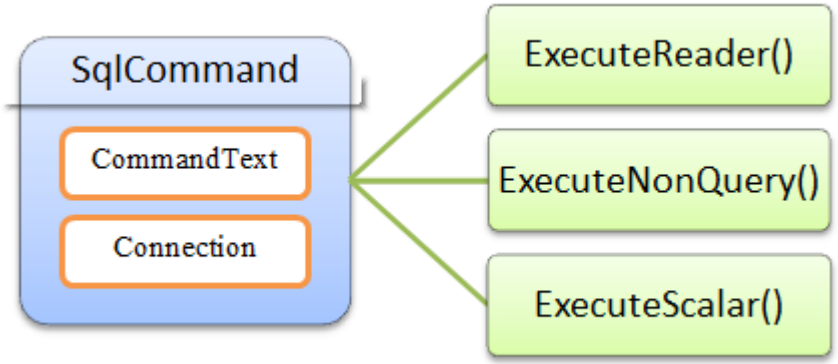
- Thư viện hỗ trợ thao tác với database từ Microsoft
- Các đối tượng
  - SqlConnection
  - SqlCommand
  - SqlDataReader
  - DataSet
  - SqlDataAdapter



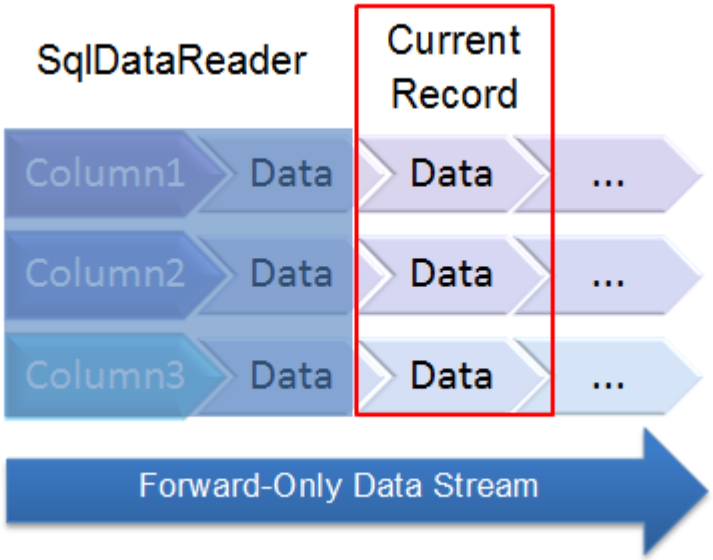
ADO.NET


SqlDataAdapter
(Disconnected)
SqlDataReader
(Connected)
SqlCommand
SqlConnection
Database

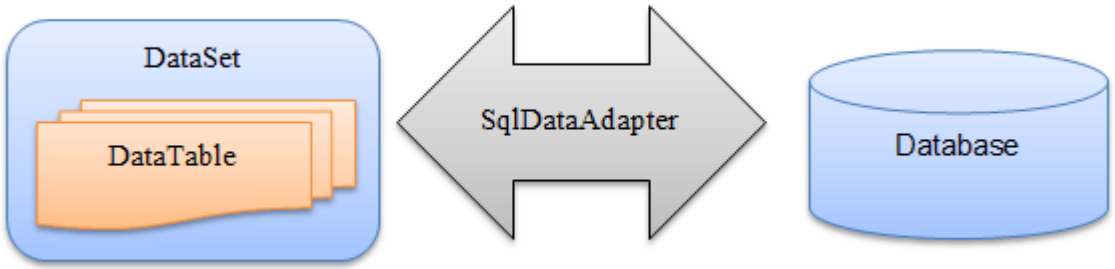
SqlCommand


SqlCommand
CommandText
Connection
ExecuteReader()
ExecuteNonQuery()
ExecuteScalar()

SqlDataReader



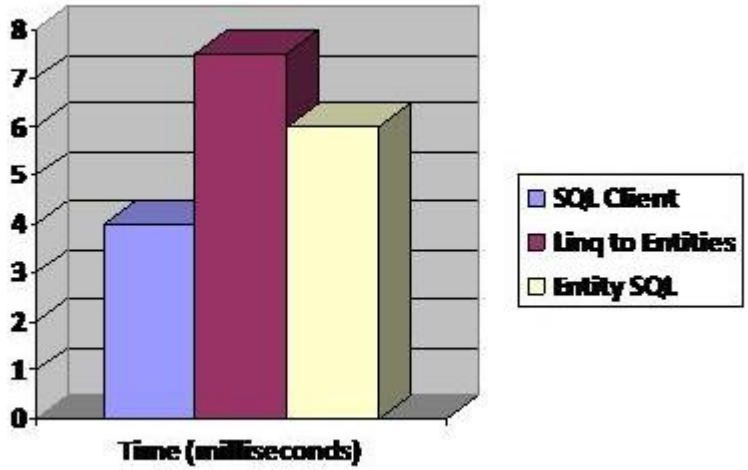
DataSet



So sánh giữa Entity Framework và ADO.NET

	Entity Framework	ADO.NET
Performance	Chậm hơn ADO.NET	Rất Nhanh
Số lượng code	Code ít hơn, đã hỗ trợ code ở tầng Data Layer => tiết kiệm thời gian	Code nhiều hơn, cấu hình phức tạp hơn => mất nhiều thời gian code
Khả năng mở rộng	phụ thuộc vào framework và khó maintain	Có thể mở rộng dễ maintain hơn

Performance




THANK YOU