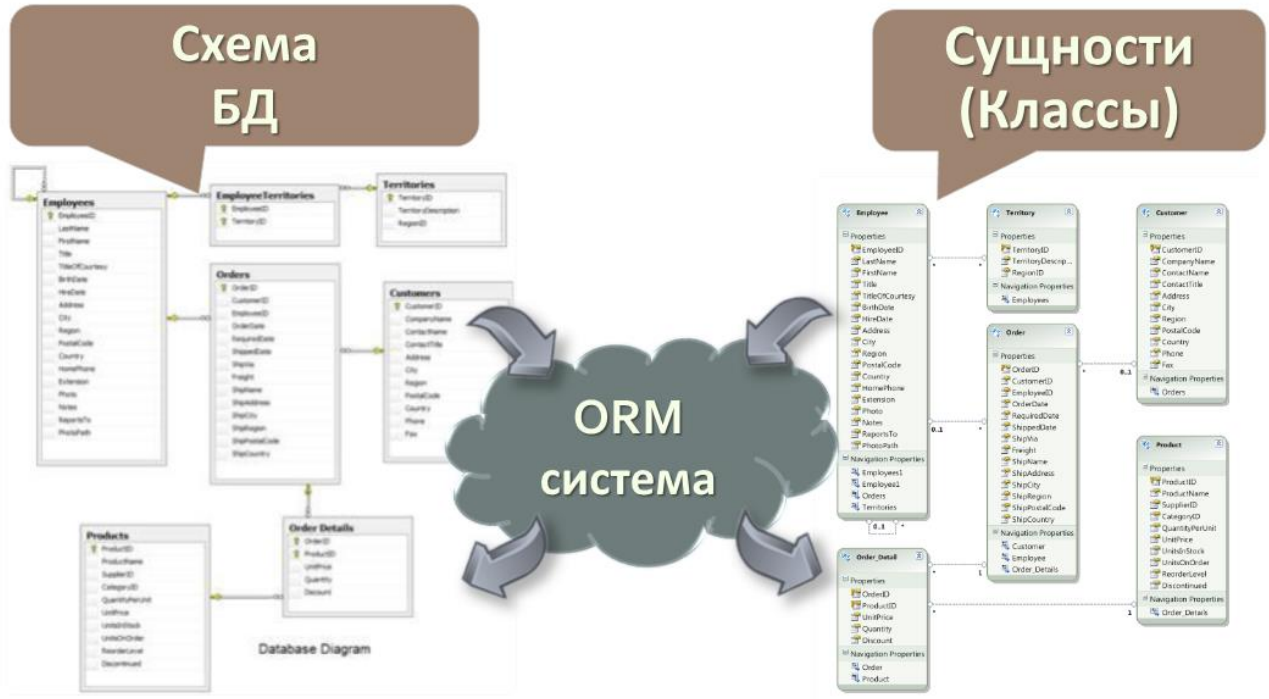
Object-relational mapping

Обьектно-реляционное отображение (ORM) технология программирования, для автоматического сопоставления и преобразования данных между реляционными СУБД и объектами из мира ООП.

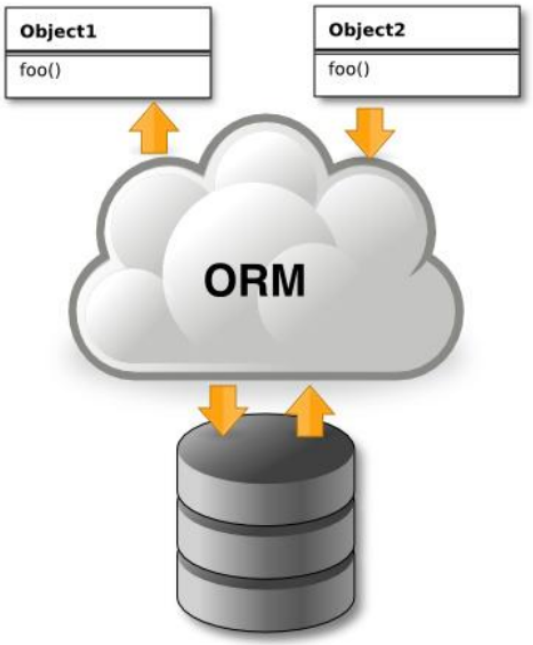
Функционал ORM систем:

- Создание объектной модели по БД
- Создание схемы БД по объектной модели
- Выполнение запросов к БД с помощью OOAPI
- CRUD –create, retrieve, update, delete

ORM-системы автоматически генерируют SQL запросы для выполнения операций над данными при вызове ОО



Преимущества



- Производительность
- Дизайн приложения
- Повторное использование
- Сопровождаемость

Архитектура

EF -набор технологий ADO.NET, обеспечивающих разработку приложений, связанных с обработкой данных

Возможности

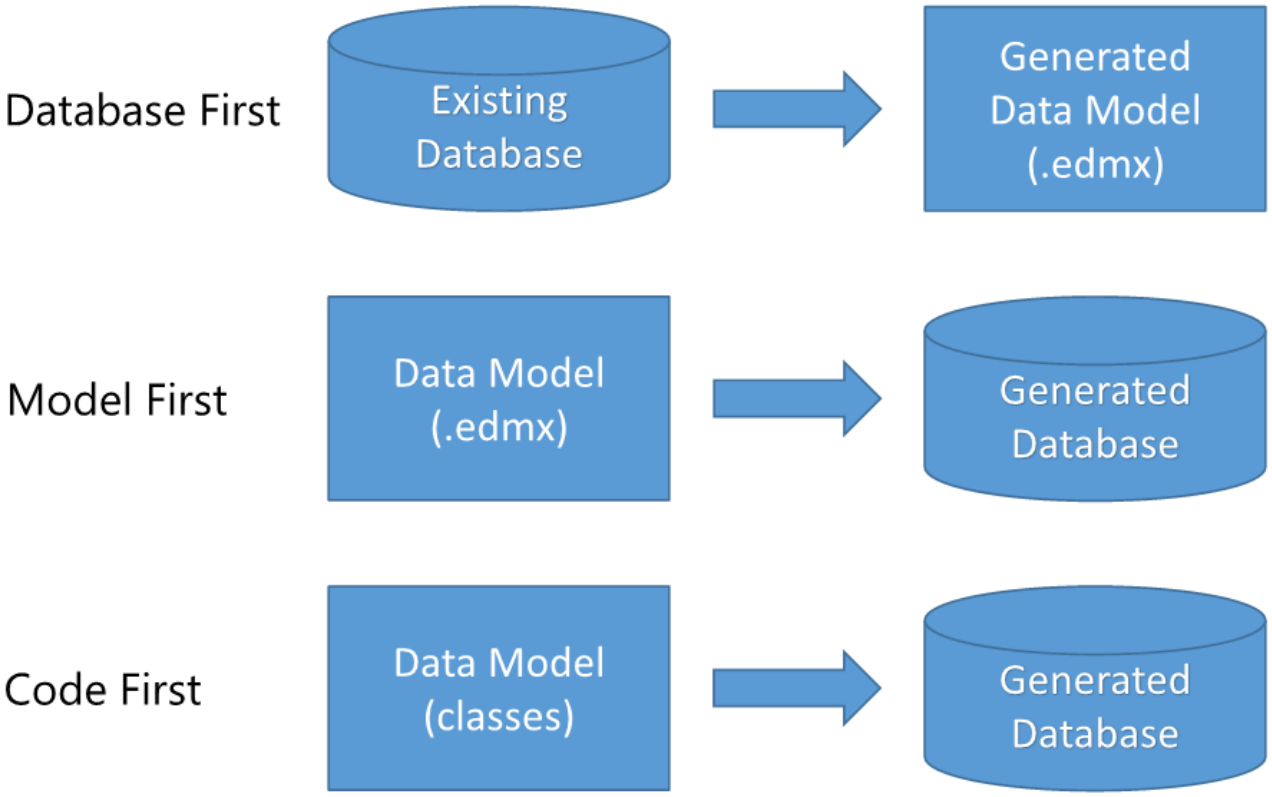
- Сопоставление объектов БД и связей между ними в объекты .NET и отношения между ними.
- Выполнение запросов к БД через работу с .NET объектами и использование API (LINQ2EF, ESQL,…)
- CRUD –Create/Read/Update/Delete
- Создание, изменение,удаление схемы БД

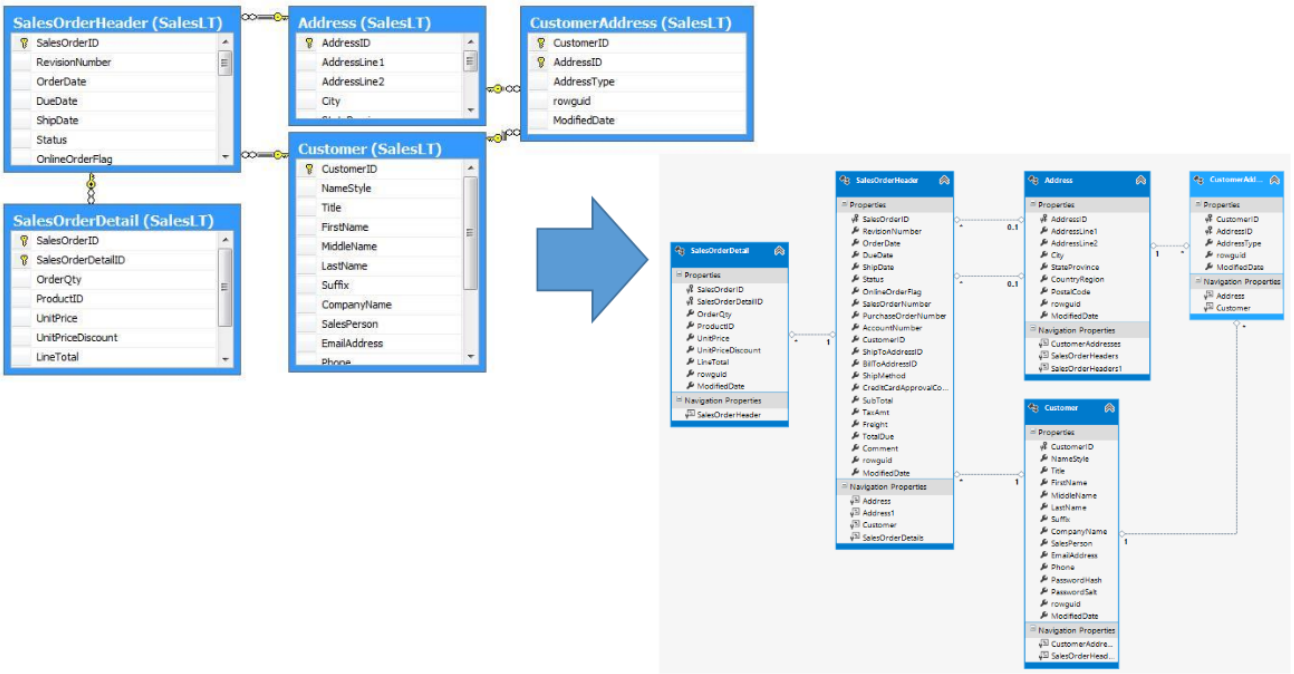
Пример

```
User user = context.Users.Include(“Payment”)
        .Single( u=>u.Id == this.User.Id );
```

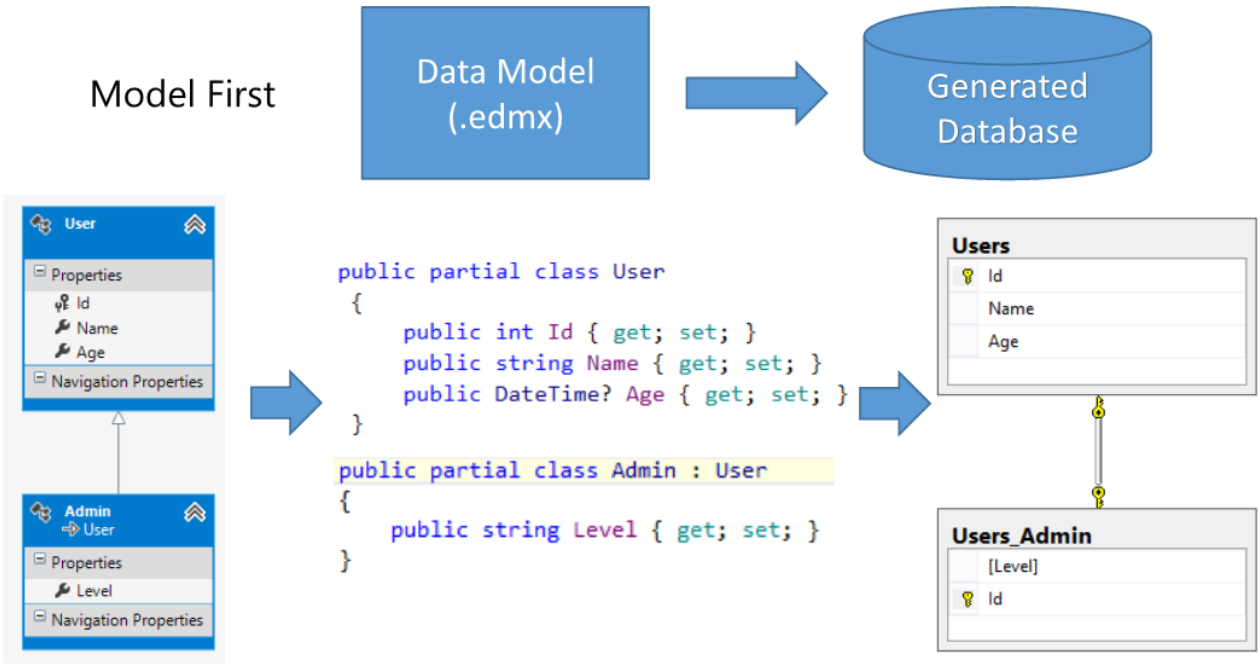
```
SELECT TOP(1) [Limit1].[Id] AS [Id], [Limit1].[UserId] AS [UserId], [Limit1].[StartDate] AS [StartDate], [Extent2].[Id] AS [Id1], [Extent2].[PaidAmount] AS [PaidAmount], [Limit1].[VideoSet_Id] AS [VideoSet_Id] FROM (SELECT TOP (1) [Extent1].[Id] AS [Id], [Extent1].[UserId] AS [UserId], [Extent1].[StartDate] AS [StartDate], [Extent1].[VideoSet_Id] AS [VideoSet_Id] FROM [Video].[VideoUserAccess] AS [Extent1]

WHERE [Extent1].[Payment_Id] IS NOT NULL ) AS [Limit1]

LEFT OUTER JOIN [Video].[Payments] AS [Extent2] ON [Limit1].[Payment_Id] = [Extent2].[Id]
```

Основные способы создания моделей

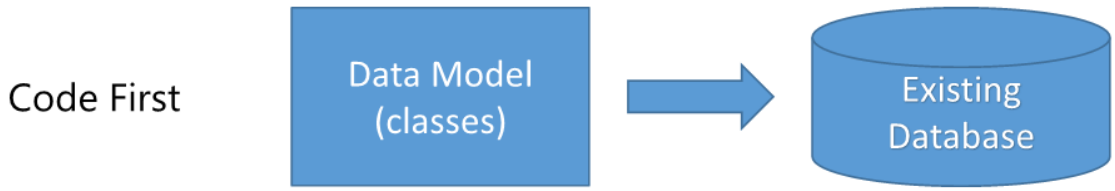




Database First – создание обьектной модели на основании существующей БД.



Model First - создание xml файла модели, при помощи дизайнера на основании которой генерируются БД и объектная модель.

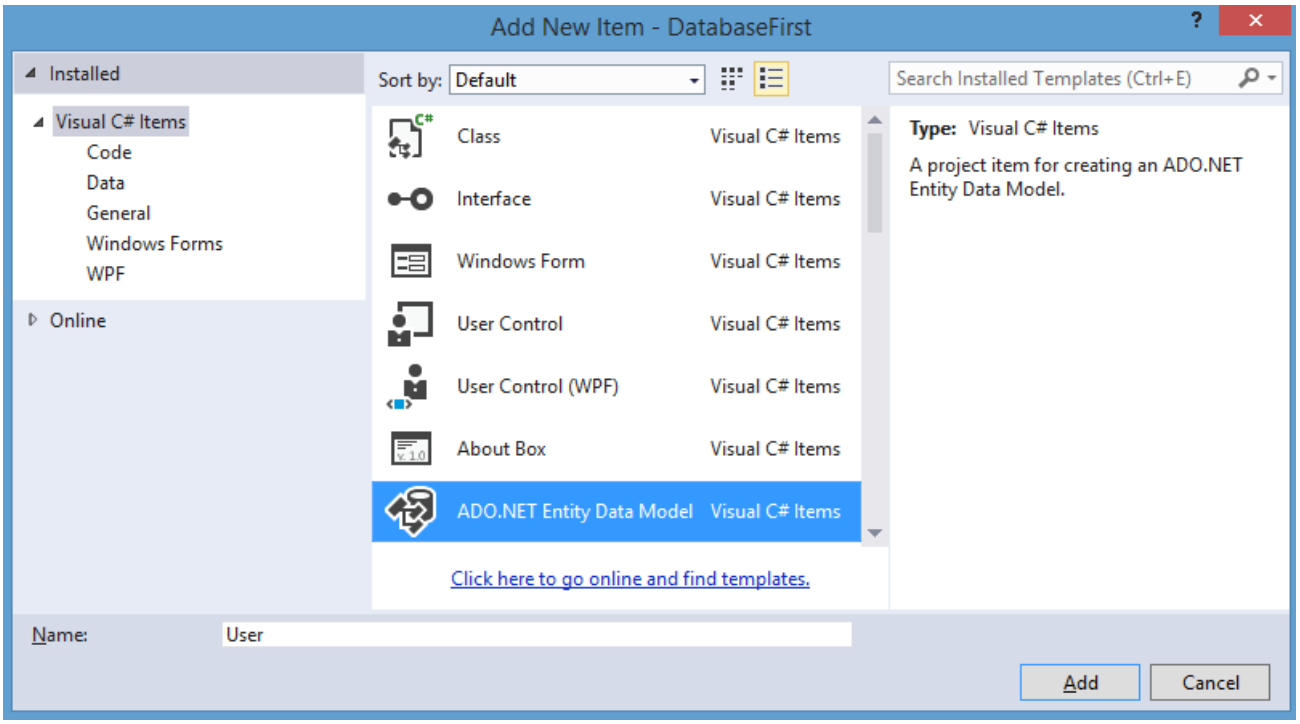


Code First – написание кода классов предметной области, при отсутствии модели и БД.
Генерация БД и модели сущностей EDM происходит после построения проекта.

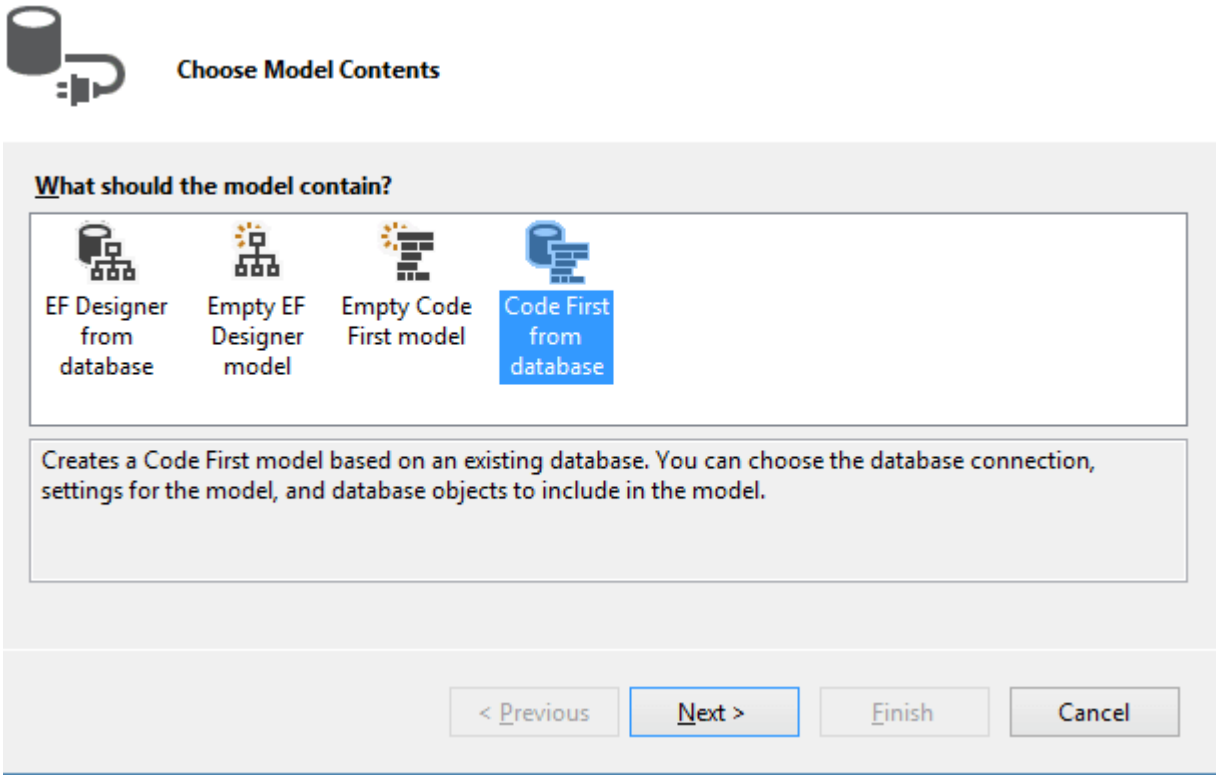
Автоматизация Code First

Вручную создавать классы по уже готовой бд со всеми полями и связями между собой довольно утомительно, особенно если таблиц в БД очень много. В обновленных версиях Visual Studio 2013 с пакетами обновления SP3 мы можем автоматизировать этот процесс.

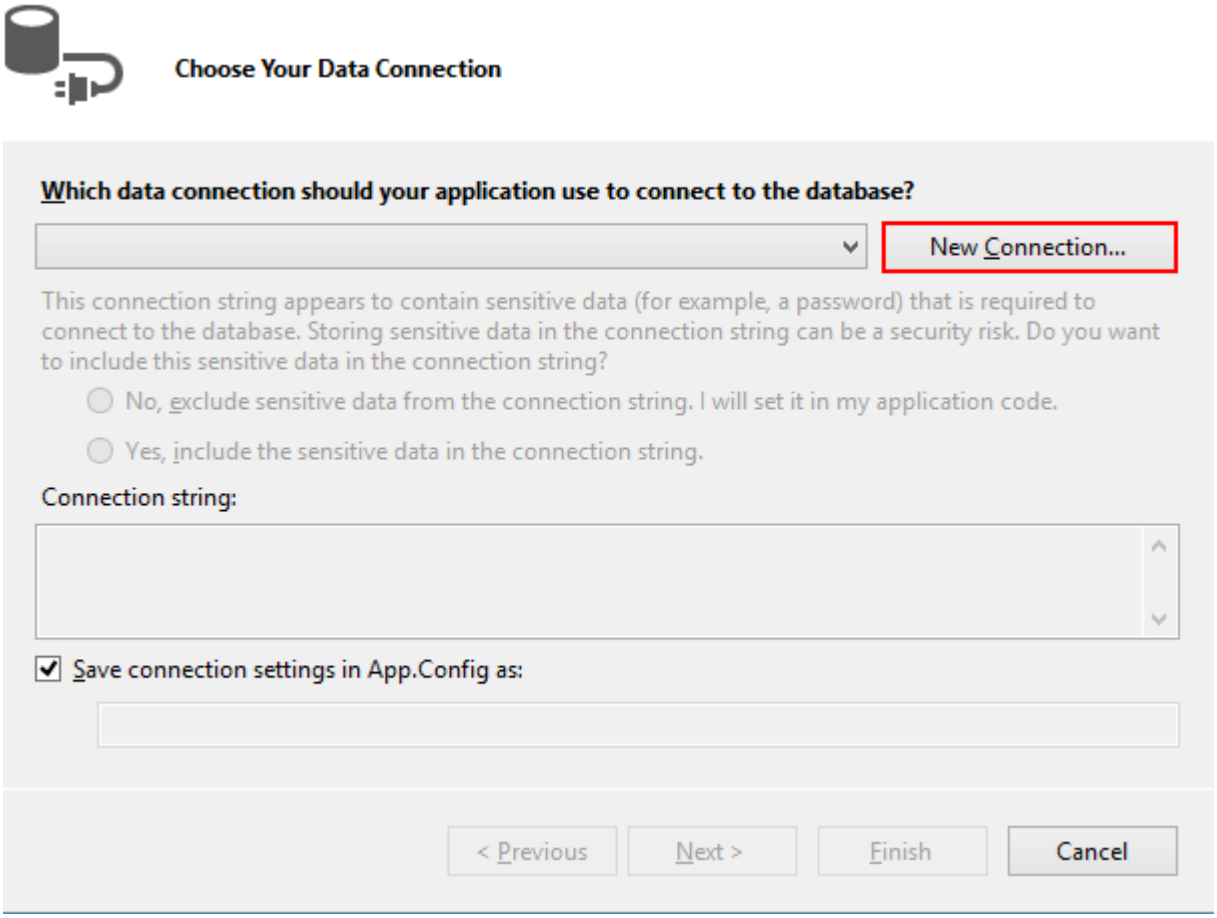
Для этого добавим в проект новый элемент**ADO.NET Entity Data Model**



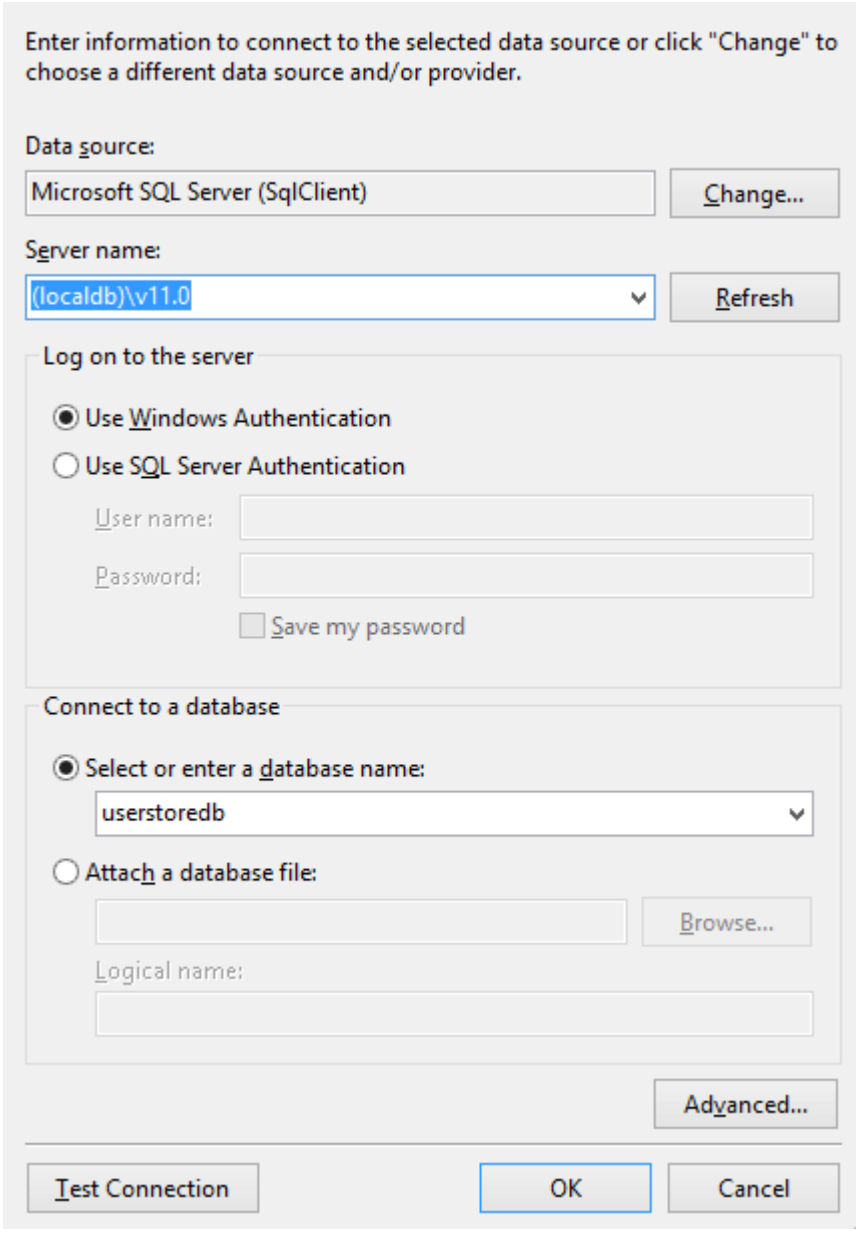
Нажмем OK и нам откроется мастер создания модели. Здесь нам надо выбрать пункт **Code First from database**



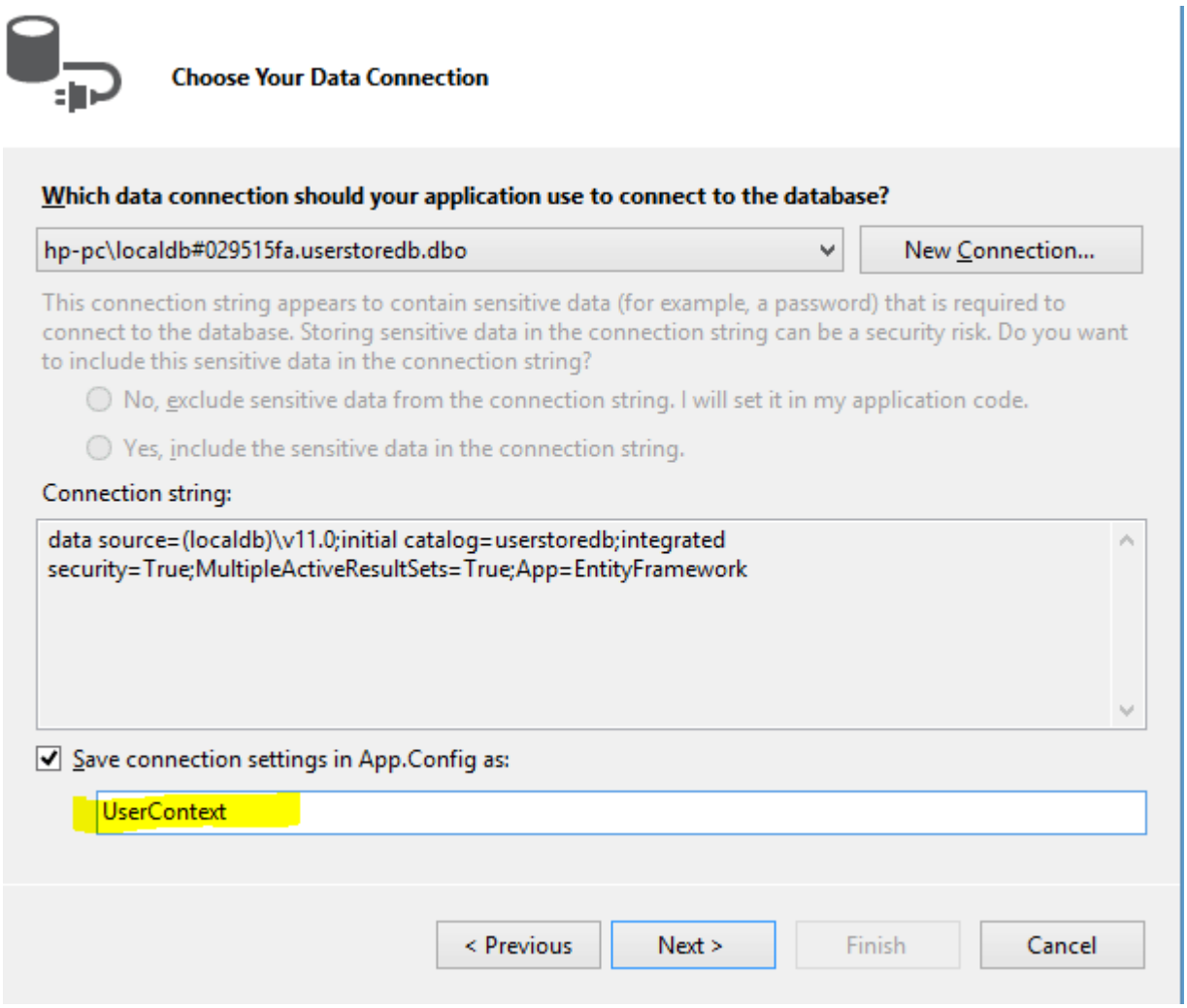
Далее на следующем шаге настройки модели надо будет установить подключение к имеющейся базе данных



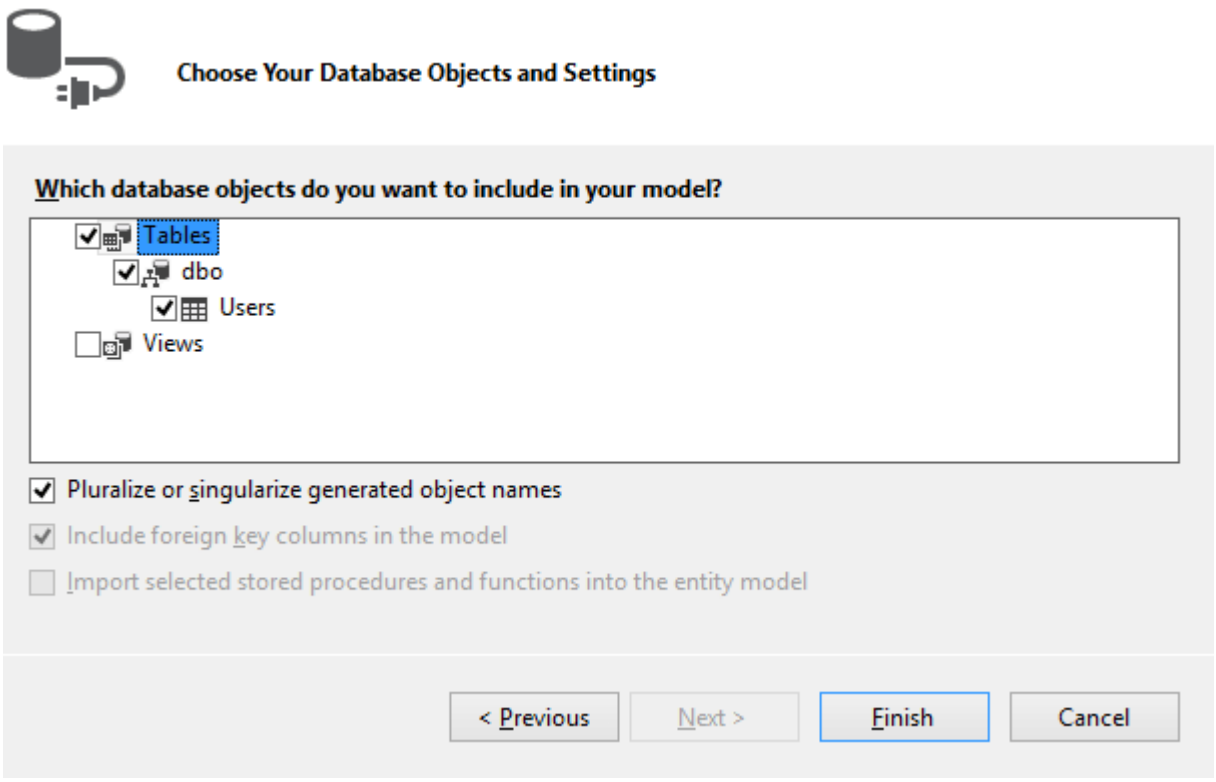
Нажмем на кнопку **New Connection** и в следующем окне настроек подключения выберем сервер и базу данных, с которой мы хотим работать



После этого в окне мастера настройки модели появится выбранное подключение. И также здесь мы можем установить название подключения, которое будет использоваться в файле конфигурации App.config. Изменим его, например, на UserContext



Нажмем Next, и на следующем шаге нам будет предложено выбрать те таблицы из бд, по которым нам надо создать модели



И затем нажмем Finish. После этого будут сгенерированы классы моделей

```
namespace AutoCodeSecond
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.Data.Entity.Spatial;

    public partial class User
    {
        public int Id { get; set; }

        [Required]
        [StringLength(50)]
        public string Name { get; set; }

        public int Age { get; set; }
    }
}
```

Соглашения по наименованию в Code First

При создании таблиц и их столбцов в базе данных в Entity Framework по умолчанию действуют некоторые соглашения по именованию, которые указывают, какие имена должны быть у таблиц, столбцов, какие типы и т.д. Рассмотрим некоторые из этих соглашений

Сопоставление типов

Типы SQL Servera и C# сопоставляются следующим образом:

- int : **int**
- bit : **bool**
- char : **string**
- date : **DateTime**
- datetime : **DateTime**
- datetime2 : **DateTime**
- decimal : **decimal**
- float : **double**
- money : **decimal**
- nchar : **string**
- ntext : **string**
- numeric : **decimal**
- nvarchar : **string**
- real : **float**
- smallint : **short**
- text : **string**
- tinyint : **byte**
- varchar : **string**

NULL и NOT NULL

Все первичные ключи по умолчанию имеют определение NOT NULL.

Столбцы, сопоставляемые со свойствами ссылочных типов (string, array), в базе данных имеют определение NULL, а все значимые типы (DateTime, bool, char, decimal, int, double, float) - NOT NULL.

Если свойство имеет тип `Nullable<T>`, то оно сопоставляется со столбцом с определением NULL.

Ключи

Entity Framework требует наличия первичного ключа, так как это позволяет ему отслеживать объекты. По умолчанию в качестве ключей EF рассматривает свойства с именем *Id* или *[Название_типа]Id* (например, PostId в классе Post).

Как правило, ключи имеют тип `int` или `GUID`, но также могут представлять и любой другой примитивный тип.

Названия таблиц и столбцов

С помощью специального класса **PluralizationService** Entity Framework проводит сопоставление между именами классов моделей и именами таблиц. При этом таблицы получают по умолчанию в качестве названия множественное число в соответствии с правилами английского языка, например, класс User - таблица Users, класс Person - таблица People (но не Persons!).

Названия столбцов получают названия свойств модели.

Если нас не устраивают названия таблиц и столбцов по умолчанию, то мы можем переопределить данный механизм с помощью Fluent API или аннотаций