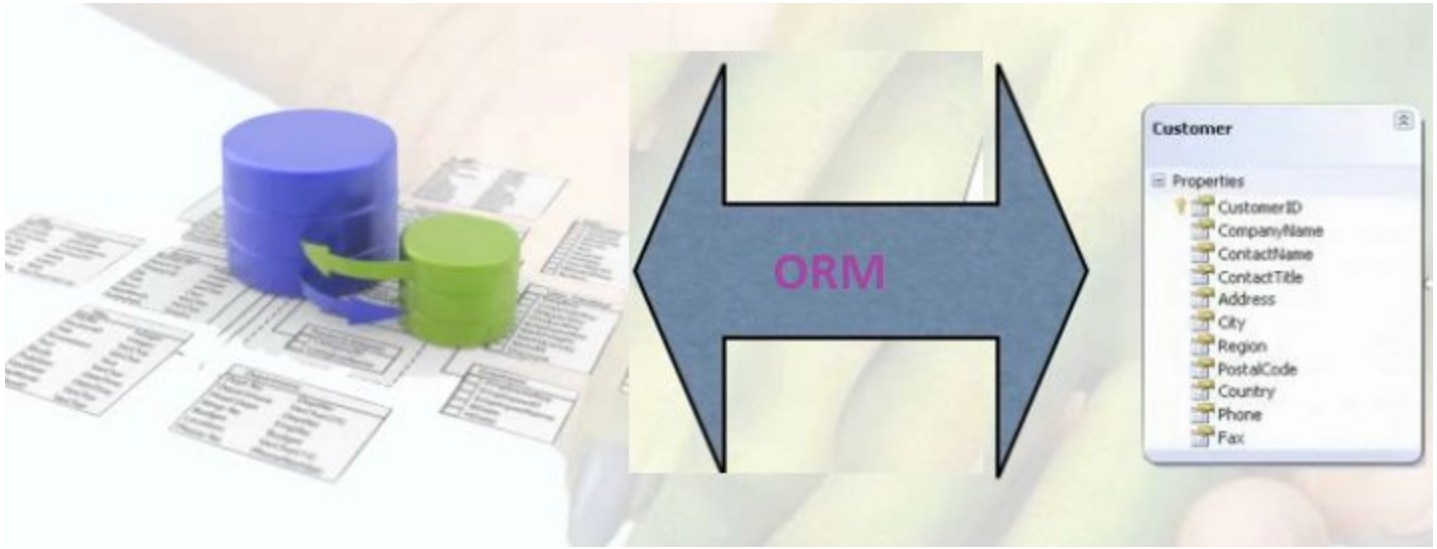
ADO.NET Entity Framework

1. ORM - object-relational mapping. Entity framework
2. Entity Data Model
3. Архітектура Entity Framework
4. EntityClient Managed Provider
5. Entity Object Services
6. Entity SQL і Linq до Entities
7. EDM Design Підходи: DB first, Model first, Code first

-

ORM - object-relational mapping

- **Relational Data Base – збереження даних**



- **Object oriented languages – керування даними**

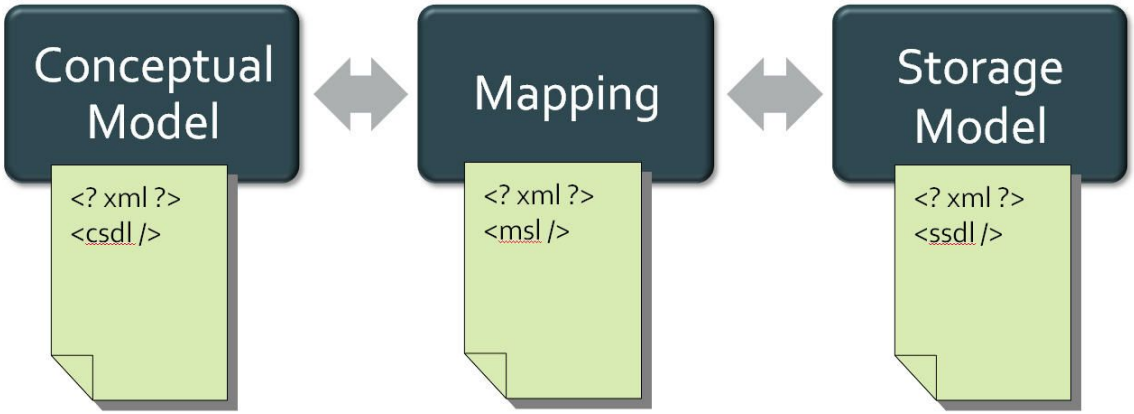
- **Основні принципи**: operations with sets – encapsulation, inheritance,…
- **Основні об'єкти** : tables, keys,… - objects, references,…
- **Різниці в типах**: проблеми конвертування

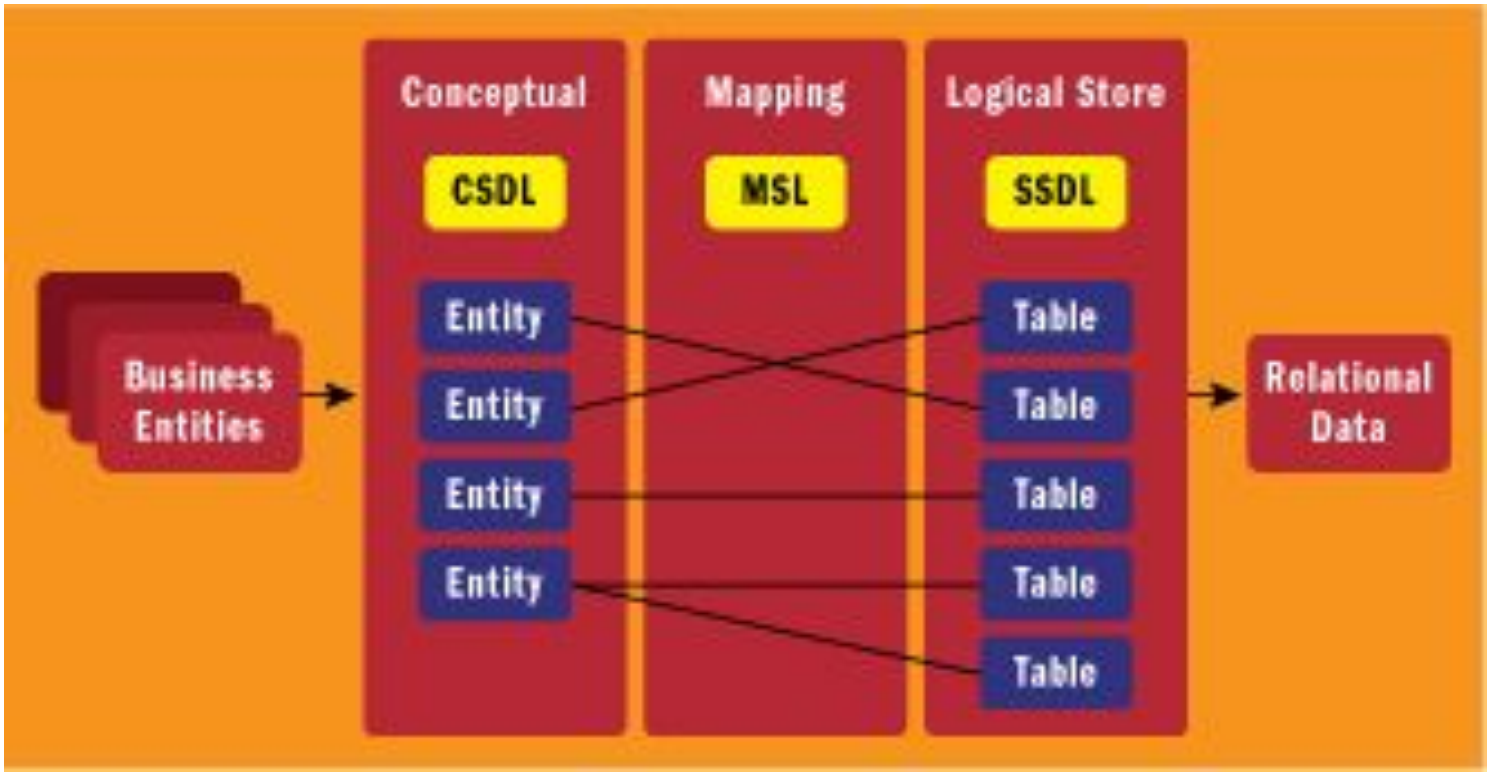
ADO.NET Entity Framework (EF)

- ADO.NET Entity Framework (EF)
  - технологія об’єктно-орієнтований доступ до даних
  - Microsoft рішення **object-relational mapping** (ORM) для . NET Framework
- Використовується як ADO.NET Data Services, дозволяє будувати багаторівневі аплікації, імплементуючи один з дизайн патернів MVC, MVP or MVVM

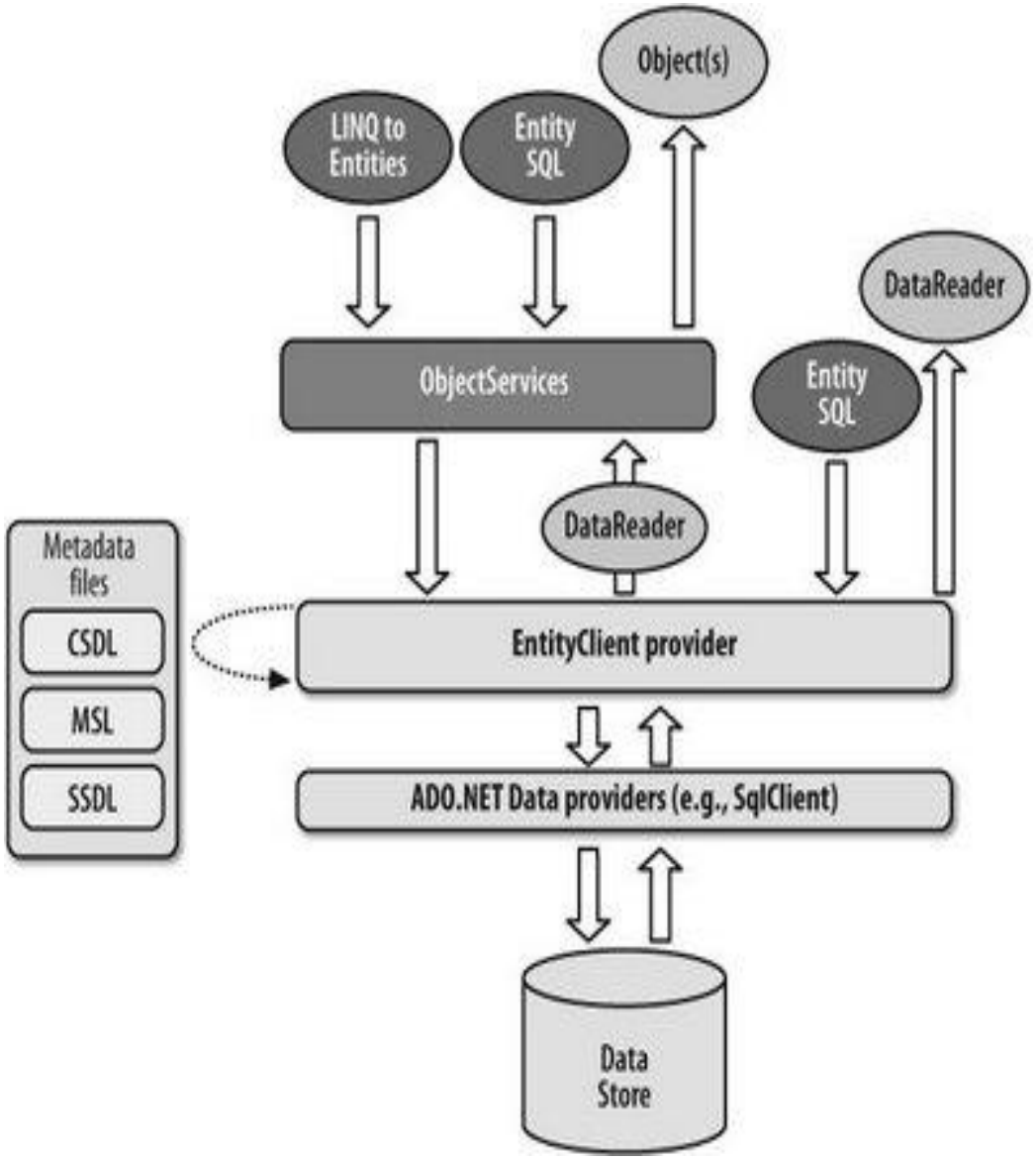
Entity Data Model

- **Entity Data Model (EDM)** це схематична мова для сутностей. Доступна для визначення сутностей, відношень між ними і логічних об'єднань відповідних сутностей.
  - Визначення сутностей наз. **Conceptual Model** і зберігається як XML документ **CSDL**
  - Для опису, як дані зберігаються - **SSDL**, наз. **Storage Model**. Це дозволяє визначити таблиці, поля, запити або stored procedures для видобування або зберігання даних. Теж зберігається як XML документ.
  - **Mapping schema** інформує Entity Framework як здійснювати мапення. (через **MSL** – Mapping specification language)




Conceptual
Mapping
Logical Store
CSDL
MSL
SSDL
Business Entities
Entity
Entity
Entity
Entity
Table
Table
Table
Table
Table
Relational Data

Entity Framework Architecture

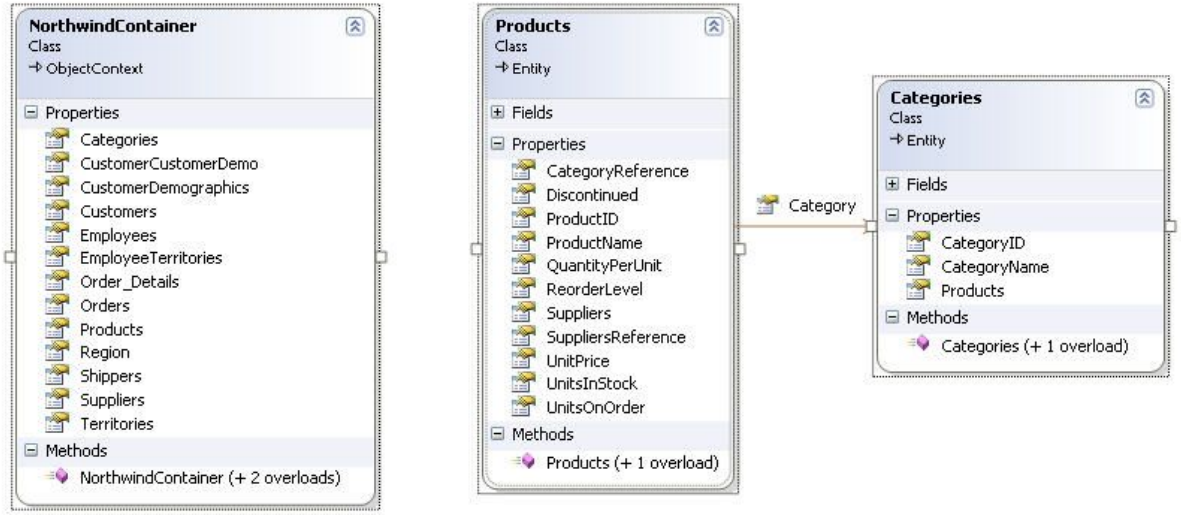


EntityClient Managed Provider. Example

```
string city = “London”;
using (EntityConnection cn = new EntityConnection(“Name=NorthwindEntities”))
{
   cn.Open();
   EntityCommand cmd = cn.CreateCommand();
   cmd.CommandText =  “SELECT VALUE c FROM NorthwindEntities.Customers “ +
                “AS c WHERE c.City = @city”;
   cmd.Parameters.AddWithValue(“city”, city);
   DbDataReader rdr = cmd.ExecuteReader( CommandBehavior.SequentialAccess);
   while (rdr.Read())
       Console.WriteLine(rdr[“CompanyName”].ToString());
   rdr.Close();
}
“metadata=.\NorthwindEntities.csdl|.\NorthwindEntities.ssdl|.
\NorthwindEntities.msl;provider=System.Data.SqlClient;provider connection string=’Data Source=DDVPC01
\SQLEXPRESS;Initial Catalog=Northwind;
Integrated Security=True’”
```

Entity Object Services

- В дизайн часі концептуальна модель EDM використовується для генерації конкретних класів сутностей (Entity) для використання в пам’яті. Ці класи утворюються для того, щоб можна було використовувати часткові (partial) типи для розширення класів сутностей.
- Додатково до сутностей (Entities) та відношень між ними генерується клас-контейнер (ObjectContext), який представляє множину (set) сутностей і використовується для створення запитів і керування базою.



Entity SQL і Linq до Entities

- Ми можемо використовувати Service Object Services для виконання запитів **Entity SQL** або можемо писати запити використовуючи **LINQ to Entities**.
- Приклад показує запит до Entity SQL, який виконується через сервіс Object Services для списку Customers.

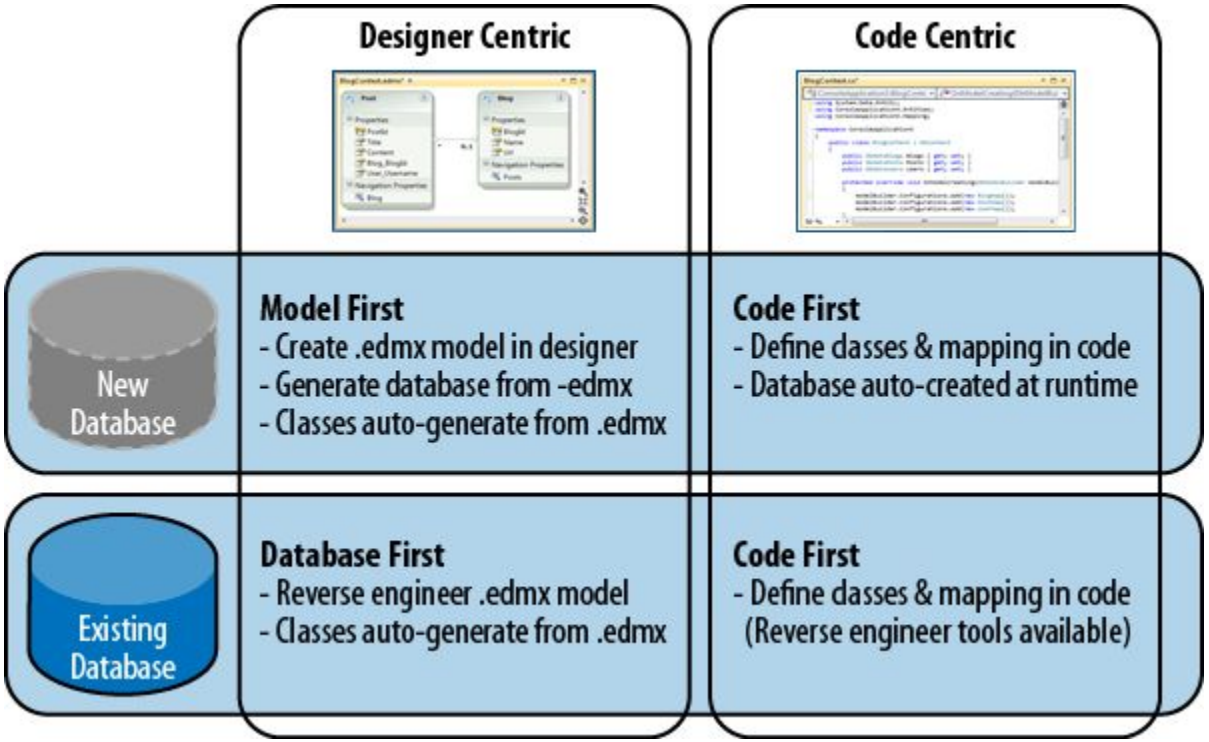
```
string city = "London";
ObjectQuery<Customers> query = northwindContext.CreateQuery<Customers>(
        "SELECT VALUE c FROM Customers AS c WHERE c.City = @city",
         new ObjectParameter("city", city));
foreach (Customers c in query)
{
   Console.WriteLine(c.CompanyName);
}
```

Entity Data Model Design Approaches

- **Database First**. Це історично перший підхід. Він виник з Entity Framework v1, і впроваджується із студією, починаючи з Visual Studio 2008 SP1. Він розглядає існуючу базу даних, або базу, яку щойно утворили. Entity Data Model генерується з цієї бази даних з допомогою візарду Entity Data Model Wizard.
- **Model First**. Підтримується з Visual Studio 2010, (Entity Framework v4). Спочатку концептуальна модель утворюється з дизайнером Entity Data Model Designer: сутності і відношення між ними додаються до моделі але мапінг не утворюється. Після цього візард Generate Database Wizard використовується для генерації сторидж (SSDL) і мапінг (MSL) частин з концептуальної моделі і зберігається в edmx файлі. Потім візард генерує **DDL** скрипт для утворення бази даних (ключів, таблиць). Якщо модель модифікують, візард Generate Database Wizard повинен бути використаний ще раз для узгодження змін з базою даних.

  Model First – працює тільки для MS SQL Server. Але існують **third-party** рішення для Oracle, MySQL, і PostgreSQL.
- **Code First**.Моделі визначаються через класи і конфікурації, написані девелоперами. Тут використовуються конвенції, які включені в бібліотеку. (використ. NuPack (http://nupack.codeplex.com/)

Modeling workflow options



Code First. Example

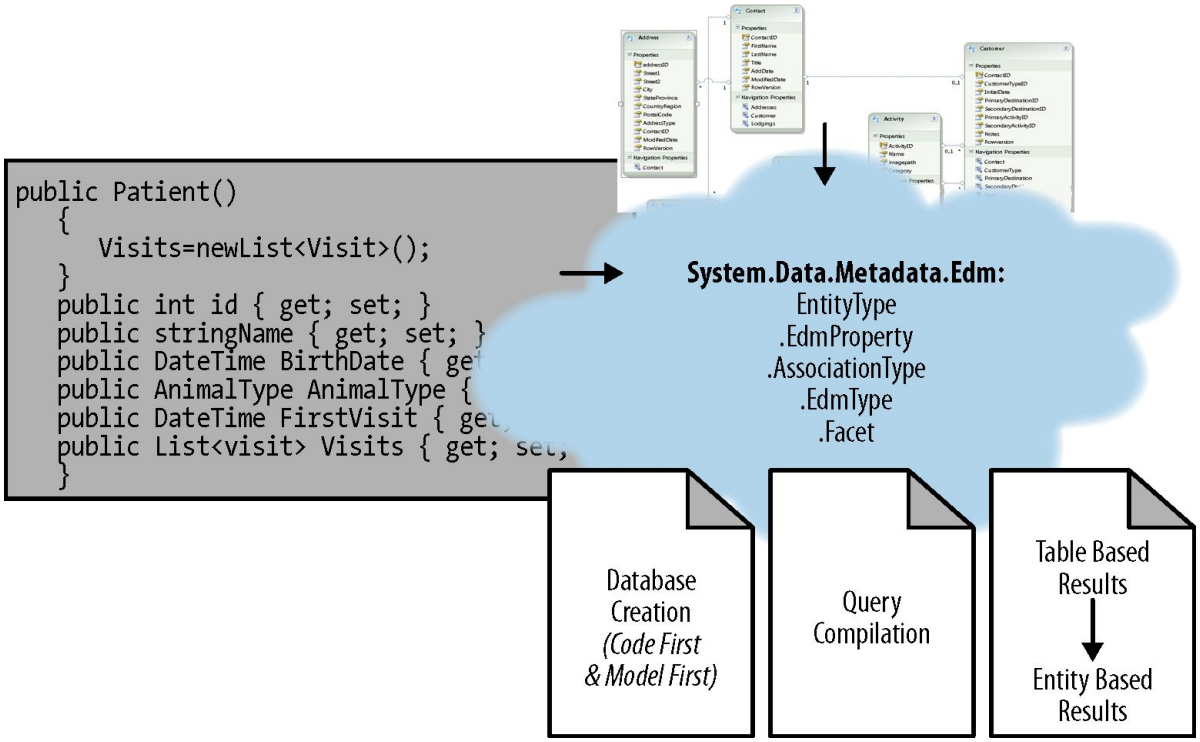
```
class Patient
{
public Patient()
{
Visits = new List<Visit>();
}
public int Id { get; set; }
public string Name { get; set; }
public DateTime BirthDate { get; set; }
public AnimalType AnimalType { get; set; }
public DateTime FirstVisit { get; set; }
public List<Visit> Visits { get; set; }
}
class Visit
{
public int Id { get; set; }
public DateTime Date { get; set; }
public String ReasonForVisit { get; set; }
public String Outcome { get; set; }
public Decimal Weight { get; set; }
public int PatientId { get; set; }
}
```

```
class AnimalType
{
public int Id { get; set; }
[Required]
public string TypeName { get; set; }
}
```

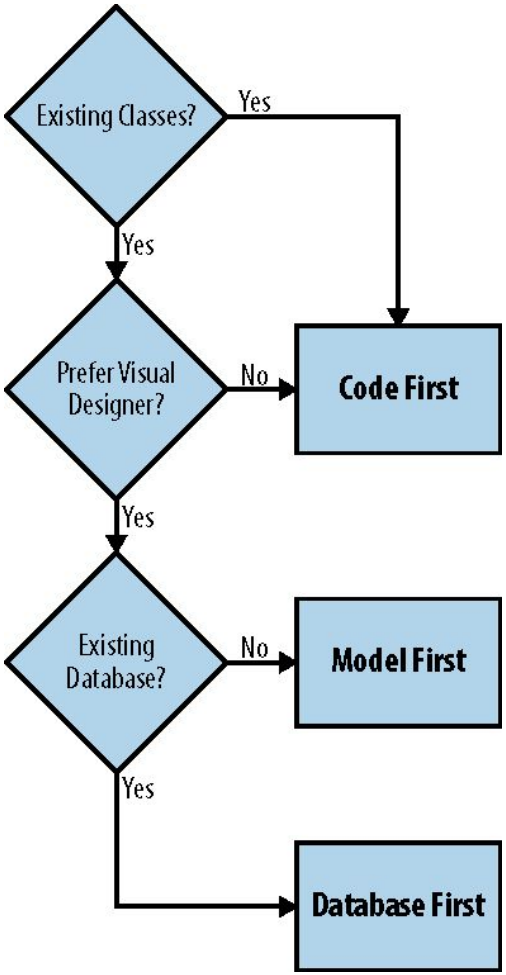
```
class VetContext:DbContext
{
public DbSet<Patient> Patients { get; set; }
public DbSet<Visit> Visits { get; set; }
}
```

Code First

- *Метадані в пам'яті утворюється з коду або EDMX моделі*



Workflow decision tree



Questions?