

# ENTITY FRAMEWORK

---

Seminari – Desenvolupament de  
Programari en Visual Studio 2019

# Objectius

- Conèixer el model de persistència denominat ***Entity Framework***(EF)
- Conèixer l'enfocament de desenvolupament per a EF denominat ***Code-First***
- Desenvolupar una aplicació d'exemple basada en l'aproximació ***Code-First*** en ***Entity Framework***

# Continguts

1. Introducció
2. EF ***DBContext***
3. Convencions *Code-First*
4. Anotacions de dades
5. Inicialització BD
6. Operacions BD
7. Estratègies de Càrrega
8. Conclusions

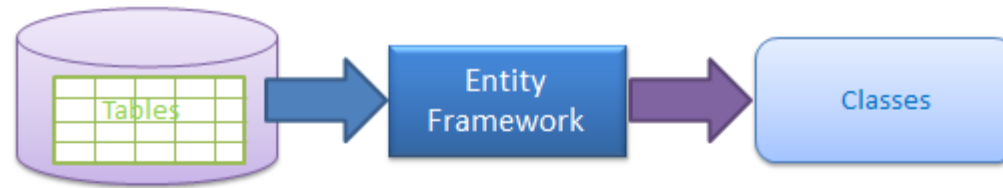
# INTRODUCCIÓ

---

# Introducció

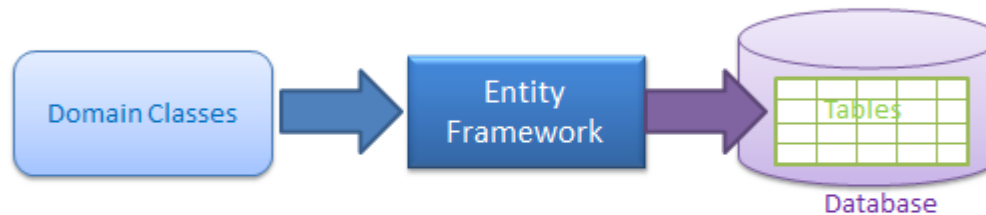
- EF es un marc de treball per a la correspondència Objecte-Relacional (*Object/Relational Mapping-O/RM*)
  - Manté el disseny de la base de dades separat del disseny de les classes del domini.
  - Automatitza operacions CRUD estàndard (*Create, Read, Update & Delete*) de forma que el desenvolupador no necessita programar-les manualment.
- EF suporta tres aproximacions de desenvolupaments:
  - **Database-first:** es disposa prèviament de la base de dades o es vol dissenyar abans que altres parts de l'aplicació.
  - **Code-first:** es vol concentrar prèviament en les classes del domini i després es vol crear la base de dades per a eixes classes del domini.
  - **Model-first:** es vol dissenyar l'esquema de base de dades en el dissenyador visual i després crear la bases de dades i les classes del domini.

# Introducció



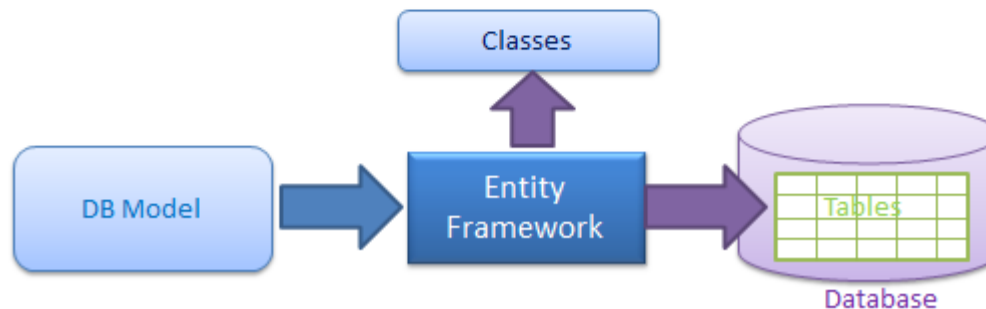
**Database-First**

Generate Data Access Classes for Existing Database



**Code-First**

Create Database from the Domain Classes



**Model-First**

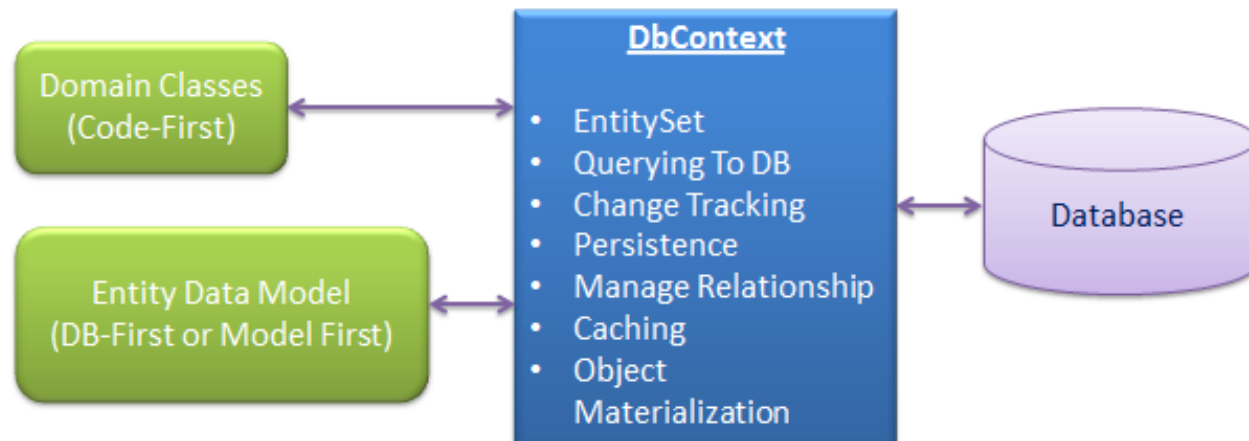
Create Database and Classes from the DB Model design

# DBCONTEXT

---

## 2. Classe *DbContext*

- Defineix el mapeig d'objectes del domini a taules en la base de dades seguint el **Patró Repositori+UoW**.
- Ofereix un mecanisme d'accés en memòria als objectes persistits. Cada classe a persistir es representa mitjançant una col·lecció de tipus **IDbSet<TEntity>** (la implementació del qual serà el **verdader contenidor/repositori** de les dades)



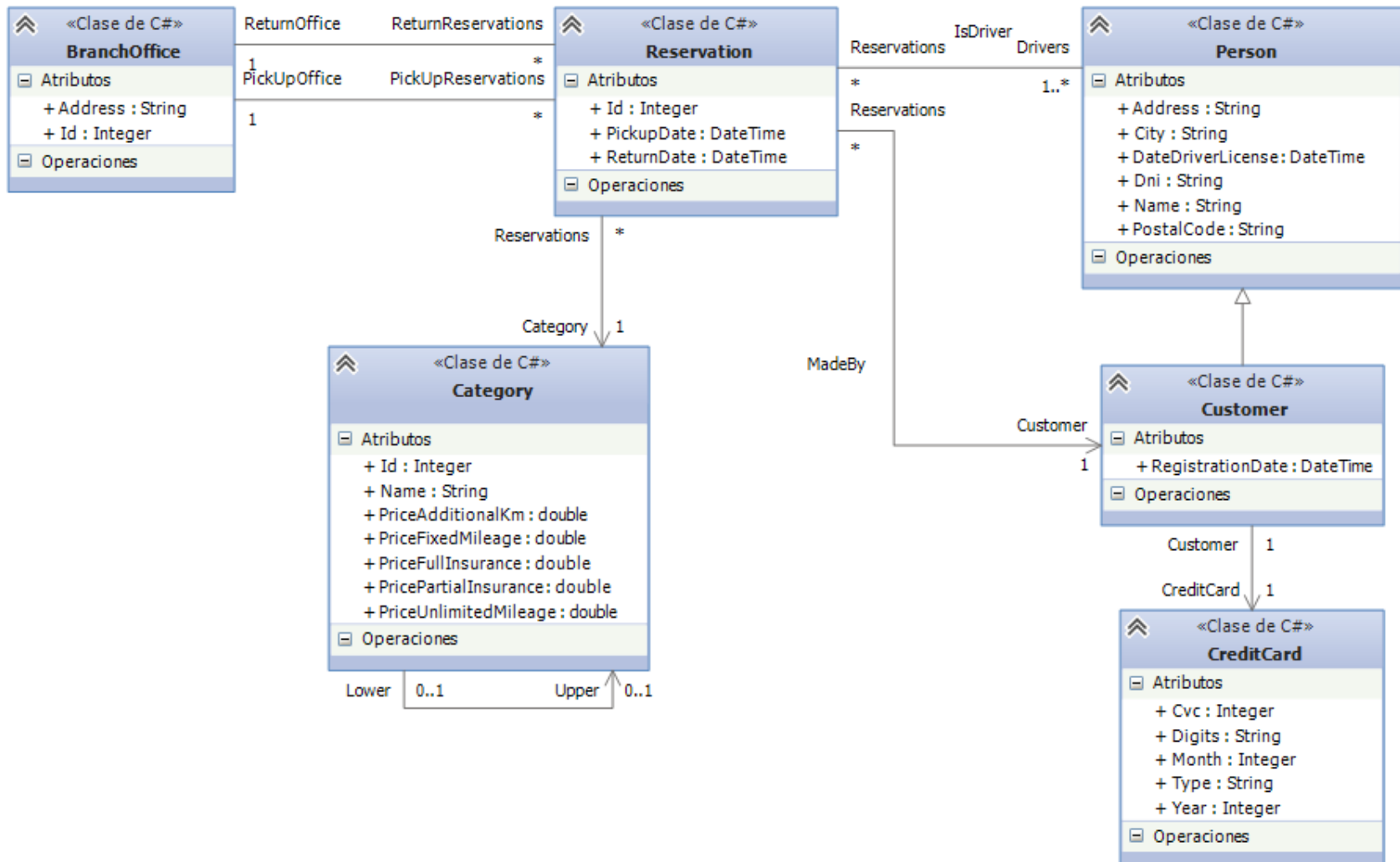


# Funcionalitat de *DbContext*

- **EntitySet:** conté **conjunts entitats** per a totes les entitats que es corresponen en taules de la base de dades (*IDbSet<TEntity>*).
- **Consultes:** converteix consultes *LINQ-to-Entities* a consultes SQL i les envia a la base de dades.
- **Control de canvis:** manté constància dels canvis que passen en les entitats després de que hagen sigut recuperades de la bases de dades.
- **Persistència de dades:** realitza operacions *Insert*, *Update* i *Delete* en la base de dades, en funció de lo definit per l'entitat.
- **Caching:** realitza el “caching” de primer nivell, per defecte. Emmagatzema en memòria les entitats que han sigut recuperades de la base de dades durant el seu cicle de vida.
- **Gestió de relacions:** també gestiona relacions gastant *fluent API* en l'aproximació *Code-First*.
- **Materialització d'objectes:** convertix dades de les taules a objectes entitat.

# ISWVehicleRental


## *(Exemple de model de domini)*



# Exemple DbContext

```
internal class VehicleRentalDbContext : DbContext
{
    public VehicleRentalDbContext() : base("Name=VehicleRentalDbConnection")
    {
    }

    public IDbSet<BranchOffice> BranchOffices { get; set; }
    public IDbSet<Reservation> Reservations { get; set; }
    public IDbSet<Category> Categories { get; set; }
    public IDbSet<Person> Persons { get; set; }
    public IDbSet<Customer> Customers { get; set; }
    public IDbSet<CreditCard> CreditCards { get; set; }
}
```



Entity Sets  
(Repositories)

- [IDbSet<TEntity>](#) representa la col·lecció de totes les entitats del contexte, o que se poden consultar de la base de dades, de un tipus concret.
- [DbSet<TEntity>](#) es una implementació concreta de IDbSet.

# IDbSet & DbSet

- `IDbSet<TEntity>` representa la collecció de totes les entitats del contexte, o que se poden consultar de la base de dades, de un tipus concret.
- `DbSet<TEntity>` es una implementació concreta de `IDbSet`.
- Els objectes `DbSet` es creen a partir d'un `DbContext` utilitzant el mètode `DbContext.Set<TEntity>()`

# CONVENCIONS CODE-FIRST

---

### 3. Convencions *Code-First*

- Les *APIs Code-First* creen la base de dades i fan el mapeig entre les classes del domini i la base de dades (taules i atributs) gastant les convencions *Code-First* per defecte.
  - Convencions **Type Discovery**
  - Convencions **Primary Key**
  - Convencions **Relationship**
  - Convencions **Foreign key**
  - Convencions **Inheritance**
- Una convenció es un conjunt de regles per defecte per a configurar automàticament un model conceptual basat en les definicions de les classes del domini

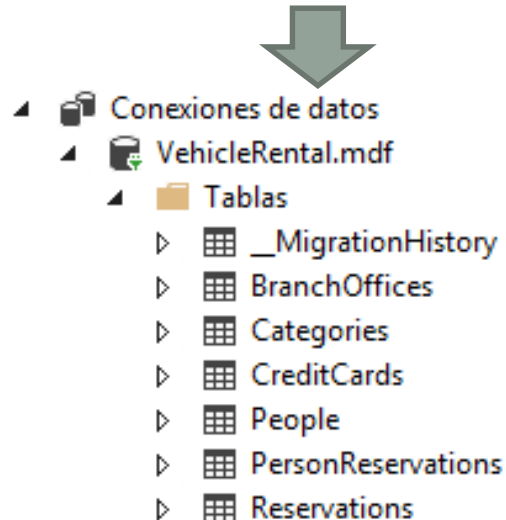
# Convenció *Type-Discovery*

- *Code-First* crearà les **taules** per a les classes incloses com propietats ***DbSet***
- *Code-First* també inclou qualsevol **tipus referenciat** que inclouen les classes

# Exemple de convenció *Type-Discovery*

EF genera automàticament una taula per cada entitat *DbSet*

```
public IDbSet<BranchOffice> BranchOffices { get; set; }  
public IDbSet<Reservation> Reservations { get; set; }  
public IDbSet<Category> Categories { get; set; }  
public IDbSet<Person> Persons { get; set; }  
public IDbSet<Customer> Customers { get; set; }  
public IDbSet<CreditCard> CreditCards { get; set; }
```



Noms de las taules en Plural.  
Ex. People en lloc de Person

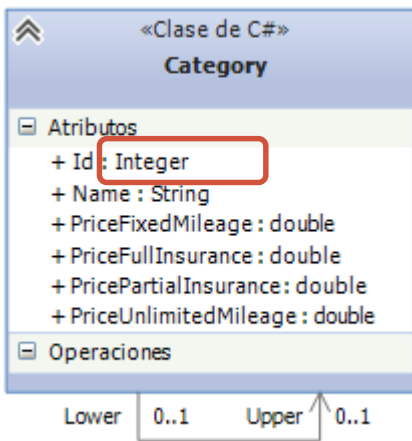
Taules addicionals per a  
relacions molts-a-molts. Ex.  
PersonReservations



# Convenció *Primary-Key*

- *Code-First* crearà una clau primària per a cada propietat que s'anomene **Id** o **<nombre de classe>Id**
- El tipus d'una propietat de clau primària pot ser qualsevol, però si el tipus de la propietat clau primària es **numèric** o **GUID** (*Identificador Únic Global*), serà configurat como una **columna identitat (auto incrementable)**.

# Exemple de Convenció *Primary-Key*



```
CREATE TABLE [dbo].[Categories] (
    [Id] INT IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NULL,
    [PriceFixedMileage] FLOAT (53) NOT NULL,
    [PriceFullInsurance] FLOAT (53) NOT NULL,
    [PriceUnlimitedMileage] FLOAT (53) NOT NULL,
    [PricePartialInsurance] FLOAT (53) NOT NULL,
    [PriceAdditionalKm] FLOAT (53) NOT NULL,
    [Upper_Id] INT NULL,
    CONSTRAINT [PK_dbo.Categories] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_dbo.Categories_dbo.Categories_Upper_Id] FOREIGN KEY ([Upper_Id])
        REFERENCES [dbo].[Categories] ([Id])
);
```

	Nombre	Tipo de datos	Permitir valores NULL
	Id	int	<input type="checkbox"/>
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PriceFixedMileage	float	<input type="checkbox"/>
	PriceFullInsurance	float	<input type="checkbox"/>
	PriceUnlimitedMileage	float	<input type="checkbox"/>
	PricePartialInsurance	float	<input type="checkbox"/>
	PriceAdditionalKm	float	<input type="checkbox"/>
	Upper_Id	int	<input checked="" type="checkbox"/>

▲ **Claves** (1)  
PK\_dbo.Categories (Clave principal, Clustered: Id)

**Restricciones CHECK** (0)

▲ **Índices** (1)  
IX\_Upper\_Id (Upper\_Id)

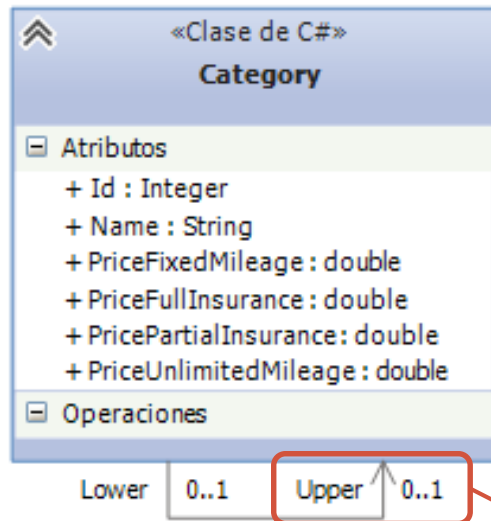
▲ **Claves externas** (1)  
FK\_dbo.Categories\_dbo.Categories\_Upper\_Id (Id)

**Desencadenadores** (0)

# Convencions *Relationship*

- Code-First infereix el **tipus de relació** en la base de dades basant-se en les **propietats de navegació**:
  - **referència** si la multiplicitat màxima és 1, o **col·lecció** si és major que 1
- Code-First modelarà una **relació un a un** si les classes inclouen dues propietats referència
- Code-First modelarà una **relació un a molts** si les classes contenen una referència i una col·lecció
- Code-First modelarà una **relació molts a molts** si les classes inclouen dues propietats de col·lecció

# Exemple de relació Un a Un



```
public class Category
{
    public virtual int Id {get;set;}

    public virtual string Name {get;set;}

    public virtual double PriceFixedMileage
    {get;set;}

    public virtual double PriceFullInsurance
    {get;set;}

    public virtual double
    PriceUnlimitedMileage {get;set;}

    public virtual double
    PricePartialInsurance {get;set;}
}
```

```
public virtual Category Upper
{
    get;
    set;
}
```

# Exemple de relació Un a Un

«Clase de C#»  
**Category**

Atributos

- + Id : Integer
- + Name : String
- + PriceFixedMileage : double
- + PriceFullInsurance : double
- + PricePartialInsurance : double
- + PriceUnlimitedMileage : double

Operaciones

Lower 0..1 Upper 0..1

```
CREATE TABLE [dbo].[Categories] (
    [Id] INT IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NULL,
    [PriceFixedMileage] FLOAT (53) NOT NULL,
    [PriceFullInsurance] FLOAT (53) NOT NULL,
    [PriceUnlimitedMileage] FLOAT (53) NOT NULL,
    [PricePartialInsurance] FLOAT (53) NOT NULL,
    [PriceAdditionalKm] FLOAT (53) NOT NULL,
    [Upper_Id] INT NULL,
    CONSTRAINT [PK_dbo.Categories] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_dbo.Categories_dbo.Categories_Upper_Id] FOREIGN KEY ([Upper_Id])
        REFERENCES [dbo].[Categories] ([Id])
);
```

	Nombre	Tipo de datos	Permitir valores NULL
Id	Id	int	<input type="checkbox"/>
Name	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
PriceFixedMileage	PriceFixedMileage	float	<input type="checkbox"/>
PriceFullInsurance	PriceFullInsurance	float	<input type="checkbox"/>
PriceUnlimitedMileage	PriceUnlimitedMileage	float	<input type="checkbox"/>
PricePartialInsurance	PricePartialInsurance	float	<input type="checkbox"/>
PriceAdditionalKm	PriceAdditionalKm	float	<input type="checkbox"/>
Upper_Id	Upper_Id	int	<input checked="" type="checkbox"/>

▲ **Claves** (1)  
PK\_dbo.Categories (Clave principal, Clustered: Id)

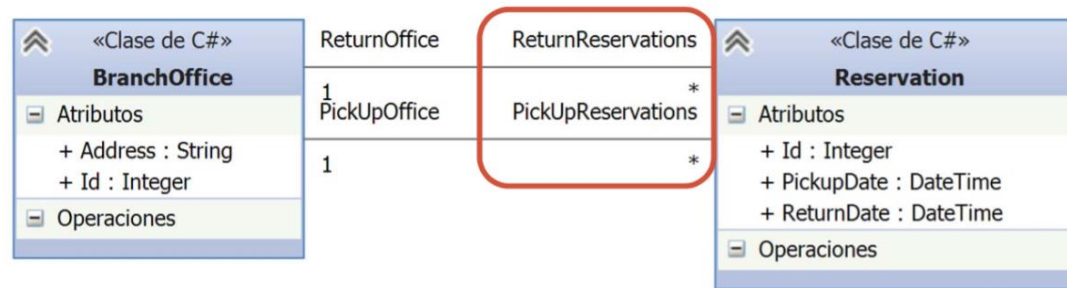
**Restricciones CHECK** (0)

▲ **Índices** (1)  
IX\_Upper\_Id (Upper\_Id)

▲ **Claves externas** (1)  
FK\_dbo.Categories\_dbo.Categories\_Upper\_Id (Id)

**Desencadenadores** (0)

# Exemple de relació Un a Molts



```

public partial class BranchOffice
{
    public string Address {get;set;}

    public int Id {get;set;}

```

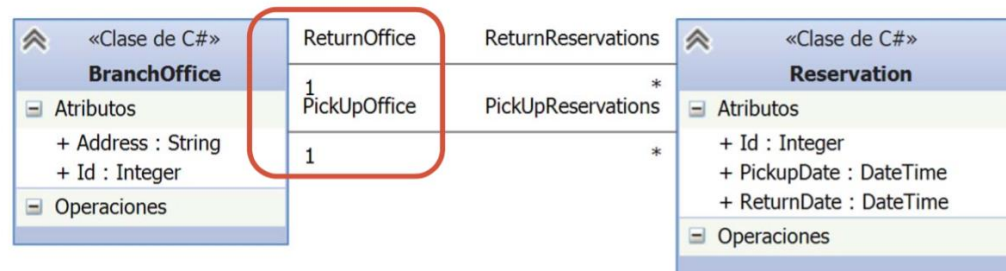
```

    public virtual ICollection<Reservation> PickUpReservations
    {
        get;
        set;
    }

    public virtual ICollection<Reservation> ReturnReservations
    {
        get;
        set;
    }
}

```

# Exemple de relació Un a Molts



```

public partial class Reservation
{
    public DateTime PickupDate {get;set;}

    public DateTime ReturnDate {get; set;}

    public int Id {get; set;}

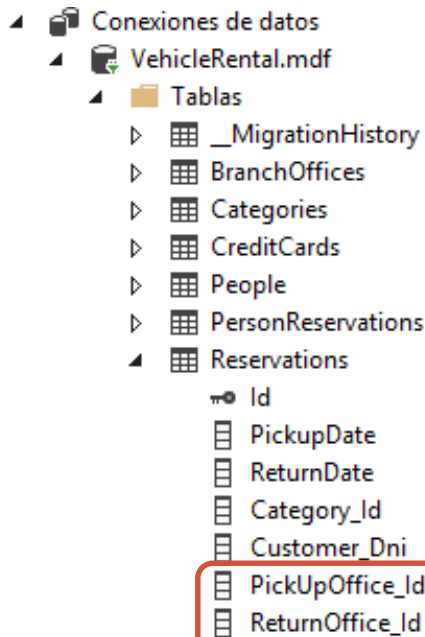
    public virtual BranchOffice PickUpOffice {get;set;}
    public virtual BranchOffice ReturnOffice {get;set;}

    public virtual Category Category {get;set;}

    public virtual ICollection<Person> Drivers {get;set;}

    public virtual Customer Customer {get;set;}
}
  
```

# Exemple de relació Un a Molts

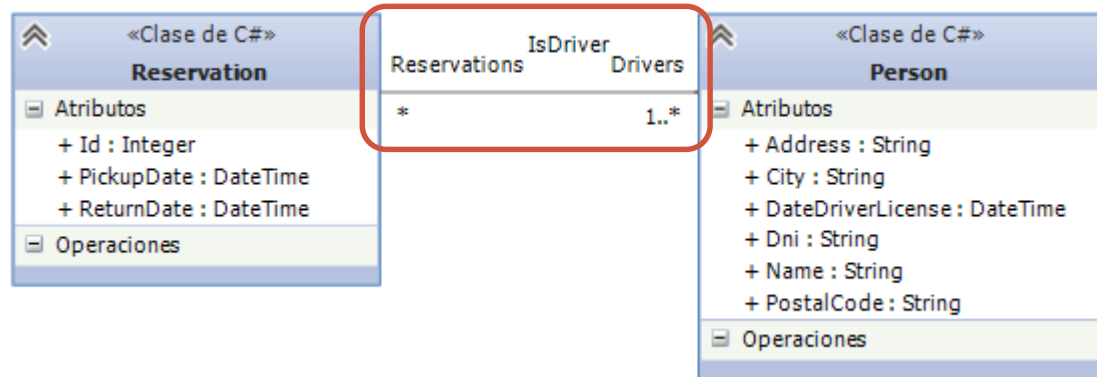


	Nombre	Tipo de datos	Permitir valores NULL
PK	Id	int	<input type="checkbox"/>
	PickupDate	datetime	<input type="checkbox"/>
	ReturnDate	datetime	<input type="checkbox"/>
	Category_Id	int	<input checked="" type="checkbox"/>
	Customer_Dni	nvarchar(128)	<input checked="" type="checkbox"/>
	PickUpOffice_Id	int	<input type="checkbox"/>
	ReturnOffice_Id	int	<input type="checkbox"/>

- Claves (1)**
  - PK\_dbo.Reservations (Clave principal, Clustered: Id)
- Restricciones CHECK (0)**
- Índices (4)**
  - IX\_Category\_Id (Category\_Id)
  - IX\_Customer\_Dni (Customer\_Dni)
  - IX\_PickUpOffice\_Id (PickUpOffice\_Id)
  - IX\_ReturnOffice\_Id (ReturnOffice\_Id)
- Claves externas (4)**
  - FK\_dbo.Reservations\_dbo.Categories\_Category\_Id (Id)
  - FK\_dbo.Reservations\_dbo.People\_Customer\_Dni (Dni)
  - FK\_dbo.Reservations\_dbo.BranchOffices\_PickUpOffice\_Id (Id)
  - FK\_dbo.Reservations\_dbo.BranchOffices\_ReturnOffice\_Id (Id)
- Desencadenadores (0)**



# Exemple de relació Molts a Molts



```
public partial class Reservation
{
    public DateTime PickupDate {get;set;}

    public DateTime ReturnDate {get; set;}

    public int Id {get; set;}

    public virtual BranchOffice PickupOffice {get;set;}

    public virtual BranchOffice ReturnOffice {get;set;}

    public virtual Category Category {get;set;}

    public virtual ICollection<Person> Drivers {get;set;}

    public virtual Customer Customer {get;set;}
}
```

```
public partial class Person
{
    public string Dni {get; set;}

    public string Name {get; set;}

    public string Address {get; set;}

    public string City {get; set;}

    public string PostalCode {get; set;}

    public DateTime DateDriverLicense {get;set;}

    public virtual ICollection<Reservation> Reservations
    {get; set;}
}
```

# Exemple de relació Molts a Molts

- ▲ Conexiones de datos
  - ▲ VehicleRental.mdf
    - ▲ Tablas
      - ▷ \_\_MigrationHistory
      - ▷ BranchOffices
      - ▷ Categories
      - ▷ CreditCards
      - ▷ People
      - ▲ PersonReservations
        - Person\_Dni
        - Reservation\_Id
      - ▷ Reservations

Nova taula, con clau primària composta

	Nombre	Tipo de datos	Permitir valores NULL
	Person_Dni	nvarchar(128)	<input type="checkbox"/>
	Reservation_Id	int	<input type="checkbox"/>

- ▲ Claves (1)
  - PK\_dbo.PersonReservations (Clave principal, Clustered: Person\_Dni, Reservation\_Id)

Restricciones CHECK (0)

- ▲ Índices (2)
  - IX\_Person\_Dni (Person\_Dni)
  - IX\_Reservation\_Id (Reservation\_Id)

- ▲ Claves externas (2)
  - FK\_dbo.PersonReservations\_dbo.People\_Person\_Dni (Dni)
  - FK\_dbo.PersonReservations\_dbo.Reservations\_Reservation\_Id (Id)

Desencadenadores (0)

Clau primària composta

Claves alienes

# Convenció de l'herència

- **Taula per Jerarquia (TPH):** Aquesta aproximació suggereix una taula per a tota la jerarquia de herència de classes.
  - La taula inclou una columna discriminadora que distingeix entre les subclasses.
  - Estratègia de “mapeo” de herència predeterminada en *Entity Framework*.
- **Taula per Tipus (TPT):** Aquesta aproximació suggereix una taula separada para cada classe del domini.
- **Taula per Classe Concreta (TPC):** Aquesta aproximació suggereix una taula per cada classe concreta sense incloure les classes abstractes.
  - Les propietats de la classe abstracta seran part de cada taula de la classe concreta.

# Exemple de Taula per Jerarquia

Conexiones de datos

VehicleRental.mdf

Tablas

- \_\_MigrationHistory
- BranchOffices
- Categories
- CreditCards
- People
  - Dni
  - Name
  - Address
  - City
  - PostalCode
  - DateDriverLicense
  - RegistrationDate
  - Discriminator
  - CreditCard\_Digits
- PersonReservations
- Reservations

	Nombre	Tipo de datos	Permitir valores NULL
	Dni	nvarchar(128)	<input type="checkbox"/>
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Address	nvarchar(MAX)	<input checked="" type="checkbox"/>
	City	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PostalCode	nvarchar(MAX)	<input checked="" type="checkbox"/>
	DateDriverLicense	datetime	<input type="checkbox"/>
	RegistrationDate	datetime	<input checked="" type="checkbox"/>
	Discriminator	nvarchar(128)	<input type="checkbox"/>
	CreditCard_Digits	nvarchar(128)	<input checked="" type="checkbox"/>

## Claves (1)

PK\_dbo.People (Clave principal, Clustered: Dni)

## Restricciones CHECK (0)

## Índices (1)

IX\_CreditCard\_Digits (CreditCard\_Digits)

## Claves externas (1)

FK\_dbo.People\_dbo.CreditCards\_CreditCard\_Digits (Digits)

## Desencadenadores (0)

Dni	Name	Address	City	PostalCode	DateDriverLic...	RegistrationDate	Discriminator	CreditCard_Di...
11111111A	Javier Murillo	Av. de las Tres ...	StartUpCity	11111	12/07/2015 0:00...	04/02/2016 0:00...	Customer	1223344556677...
22222222A	Carlos García	Plaza de los Cas...	HackerCity	99999	23/05/2014 0:00...	NULL	Person	NULL

# Elements Clau de les Convencions

Default Convention For	Description
Table Name	<Entity Class Name> + 's' EF will create DB table with entity class name suffixed by 's'
Primary key Name	1) Id 2) <Entity Class Name> + "Id" (case insensitive)  EF will create primary key column for the property named Id or <Entity Class Name> + "Id" (case insensitive)
Foreign key property Name	By default EF will look for foreign key property with the same name as principal entity primary key name. If foreign key property does not exists then EF will create FK column in Db table with <Dependent Navigation Property Name> + "_" + <Principal Entity Primary Key Property Name> e.g. EF will create Standard_StandardId foreign key column into Students table if Student entity does not contain foreignkey property for Standard where Standard contains StandardId
Null column	EF creates null column for all reference type properties and nullable primitive properties.
Not Null Column	EF creates NotNull columns for PrimaryKey properties and non-nullable value type properties.
DB Columns order	EF will create DB columns same as order of properties in an entity class. However, primary key columns would be moved first.
Properties mapping to DB	By default all properties will map to database. Use [NotMapped] attribute to exclude property or class from DB mapping.
Cascade delete	Enabled By default for all types of relationships.

## 4. Configuració de les Classes del domini

- **Sobreescriure** les convencions prèvies configurant les teues pròpies classes del domini per a proveir a EF amb la informació que necessita.
- Dos formes de configurar les classes del domini:
  - ***DataAnnotations***: Configuració basada en atributs, que pot ser aplicada a las classes de domini i les seues propietats.
  - ***Fluent API***: Forma avançada d'especificar la configuració model que cobreix tot lo que les *DataAnnotations* poden fer i, a més permet configuracions avançades que no son possibles amb les anotacions.

# DATA ANNOTATIONS

---

# *Data Annotations*

- Anotacions que afecten a les característiques del camp en l'esquema de la base de dades (ex. si és clau, si ha de ser VNN, etc.):
  - *Key*
  - *Timestamp*
  - *ConcurrencyCheck*
  - *Required*
  - *MinLength*
  - *MaxLength*
  - *StringLength*

***System.ComponentModel.DataAnnotations***



# *Data Annotations*

- L'altres anotacions que determinen l'esquema de la base de dades:
  - *Table*
  - *Column*
  - *Index*
  - *ForeignKey*
  - *NotMapped*
  - *InverseProperty*
- ***System.ComponentModel.DataAnnotations.Schema***

# Exemple de *Data Annotations*

```
public partial class Person
{
    [Key]
    public string Dni {get; set;}

    public string Name {get; set;}

    public string Address {get; set;}

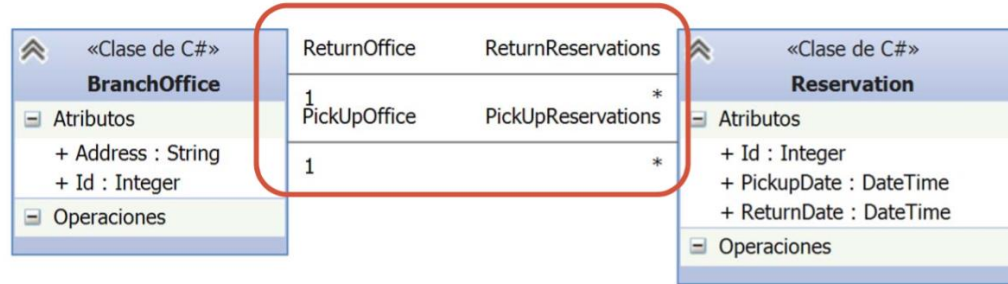
    public string City {get; set;}

    public string PostalCode {get; set;}

    public DateTime DateDriverLicense {get;set;}

    public virtual ICollection<Reservation> Reservations {get; set;}
```

# Exemple de Data Annotations



```
public partial class BranchOffice
{
    public string Address {get;set;}

    public int Id {get;set;}

    public virtual ICollection<Reservation>
    PickUpReservations {get; set;}

    public virtual ICollection<Reservation>
    ReturnReservations {get; set;}
}
```

```
public partial class Reservation
{
    public DateTime PickupDate {get;set;}

    public DateTime ReturnDate {get; set;}

    public int Id {get; set;}

    [InverseProperty("PickUpReservations")]
    public virtual BranchOffice PickUpOffice {get;set;}

    [InverseProperty("ReturnReservations")]
    public virtual BranchOffice ReturnOffice {get;set;}

    public virtual Category Category {get;set;}

    public virtual ICollection<Person> Drivers {get;set;}

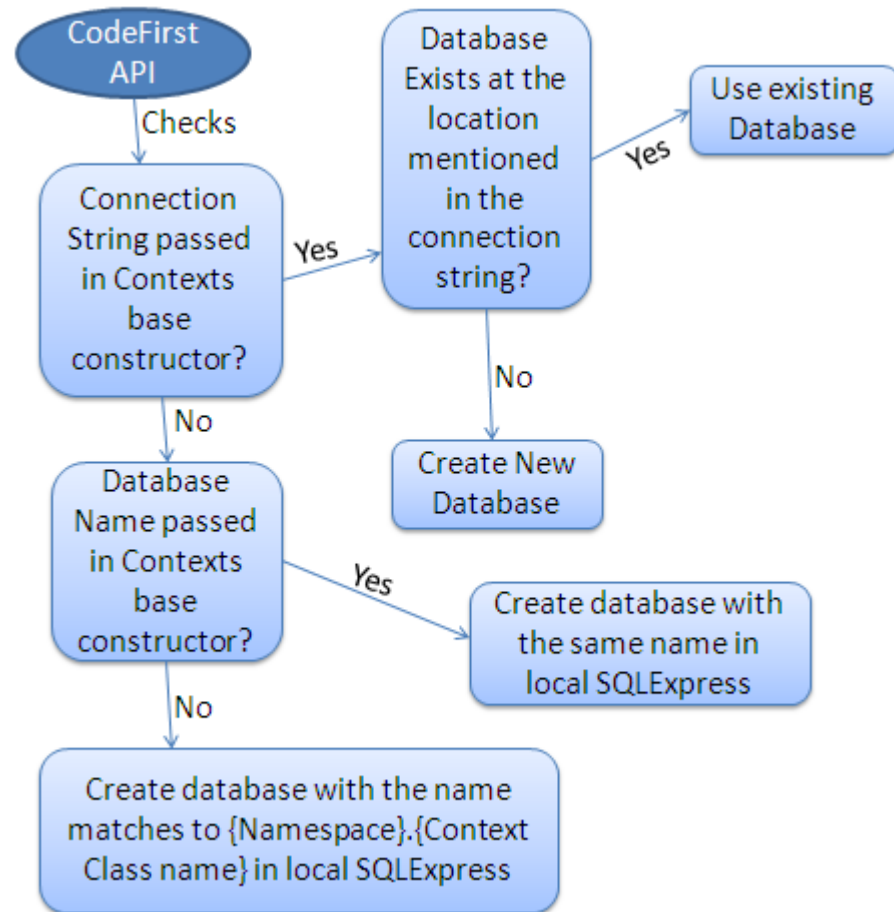
    public virtual Customer Customer {get;set;}
}
```

# INICIALITZACIÓ DE LA BASE DE DADES

---

## 5. Inicialització de la Base de Dades

- *Code First* crea una base de dades automàticament d'acord al següent diagrama:



# Exemple d'inicialització de la BD

- Cadena de connexió especificada en el constructor de la classe de context (derivada de DbContext)

```
public VehicleRentalDbContext() :  
base("Name=VehicleRentalDbConnection"){}
```

- Cadena de connexió definida en el arxiu de configuració **App.config** (o Web.config)

```
<connectionStrings>  
  <clear />  
  <add name="VehicleRentalDbConnection"  
connectionString="Server=(localdb)\mssqllocaldb;  
Database=VehicleRental;Trusted_Connection=True;MultipleActive  
ResultSets=true" providerName="System.Data.SqlClient" />  
</connectionStrings>
```

# OPERACIONS D'ACCÉS A DADES AMB ENTITY FRAMEWORK

---

## Operacions de la Base de Dades:

- **IDbSet** ofereix mètodes per afegir, esborrar i recuperar objectes
- Permet expressar i executar consultes.
- Recupera els resultats de la consulta de la base de dades i els transforma en instàncies del nostre model de classes.
- Pot fer un seguiment dels canvis en les entitat, incloent adició i esborrat; i desencadenar la creació dels comandos de inserció, actualització y esborrat que són enviats a la base de dades baix demanda



# Exemple d'operacions EF

```
VehicleRentalDbContext context = new  
VehicleRentalDbContext();
```

```
context.Categories.Add(new Category("luxury",  
23, 12, 2, 32, 14));  
context.People.Add(new Person("222222222A", ...);  
Person p = context.People.Find("12345678A");  
context.People.Remove(p);  
context.SaveChanges();
```

```
context.People.Where(person => person.Id ==  
"123456789A")
```

# Resum de característiques clau de EF

- EF es un framework de correspondència objecte-relacional (**Object-Relational Mapping**, ORM)
- EF automatitza les operacions **CRUD** estàndard (Create, Read, Update & Delete) per a que el desenvolupador no necessite escriure-les de forma manual
- **Code-First** permet centrar la atenció en las classes de domini i crear la base de dades a partir d'estes.
- Code-First crea la base de dades i realitza el mapeo entre las classes del domini i la base de dades utilitzant convencions i configuració mitjançant l'ús d'anotacions de dades o Fluent API
- **DbContext** permet expressar i executar consultes, manté el seguiment de canvis i materialitza objectes

# Tasques...

- Quins són els beneficis de tenir una capa/interfície intermèdia (com la IDAL) quan es treballa amb EntityFramework?
- Explorar *VehicleRentalApp* e identificar els diferents elements EF (context de la BD, entitats persistides, inicialització de la BD, cadena de connexió)
- Quin patró d'accés a dades s'utilitza en EF i com/on s'implementa en l'aplicació *VehicleRental*?

# Tasques...

- Entendre el significat de las diferents anotacions de dades:

[http://www.tutorialspoint.com/entity\\_framework/entity\\_framework\\_data\\_annotations.htm](http://www.tutorialspoint.com/entity_framework/entity_framework_data_annotations.htm)

- Tasca Avançada. Entendre com funciona Fluent API

[http://www.tutorialspoint.com/entity\\_framework/entity\\_framework\\_fluent\\_api.htm](http://www.tutorialspoint.com/entity_framework/entity_framework_fluent_api.htm)

# Bibliografia i referències

- Hirani, Z., et al. Entity Framework 6 Recipes 2013.
- Entity Framework Documentation (MSDN)
  - [Entity Framework Code First Conventions](#)
  - [Entity Framework Code First Data Annotations](#)
  - [Entity Framework Fluent API – Relationships](#)
  - [Entity Framework Fluent API - Configuring and Mapping Properties and Types](#)
  - [Entity Framework Loading Related Entities](#)
- Tutorials Online
  - <http://www.entityframeworktutorial.net>
  - [http://www.tutorialspoint.com/entity framework/](http://www.tutorialspoint.com/entity_framework/)