Entity Framework

Yannick Mazières
Département d'informatique
Cégep Sainte-Foy

Entity framework est un ORM (object-relational mapping)

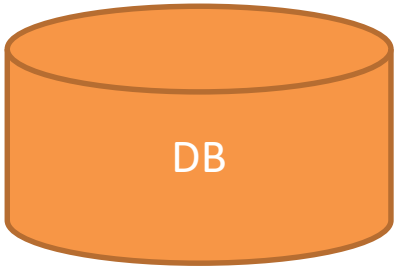
Qu’est-ce qu’un ORM ?

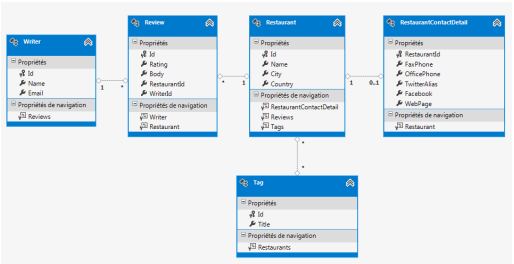
Définition ORM de Wikipédia:

> Un ORM est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé. On pourrait le désigner par « correspondance entre monde objet et monde relationnel ».

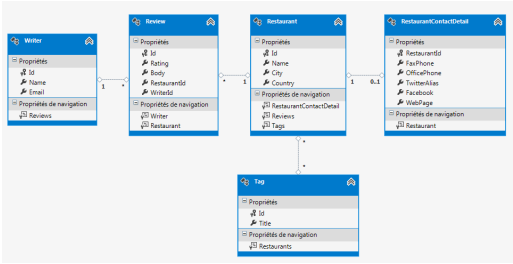
3 façons d'utiliser EF

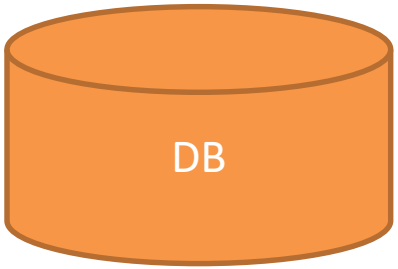
Database First





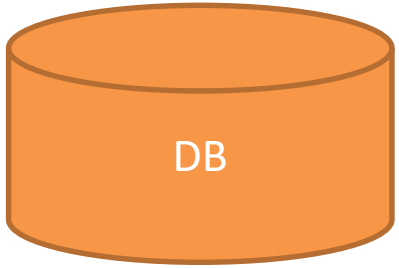
Model First



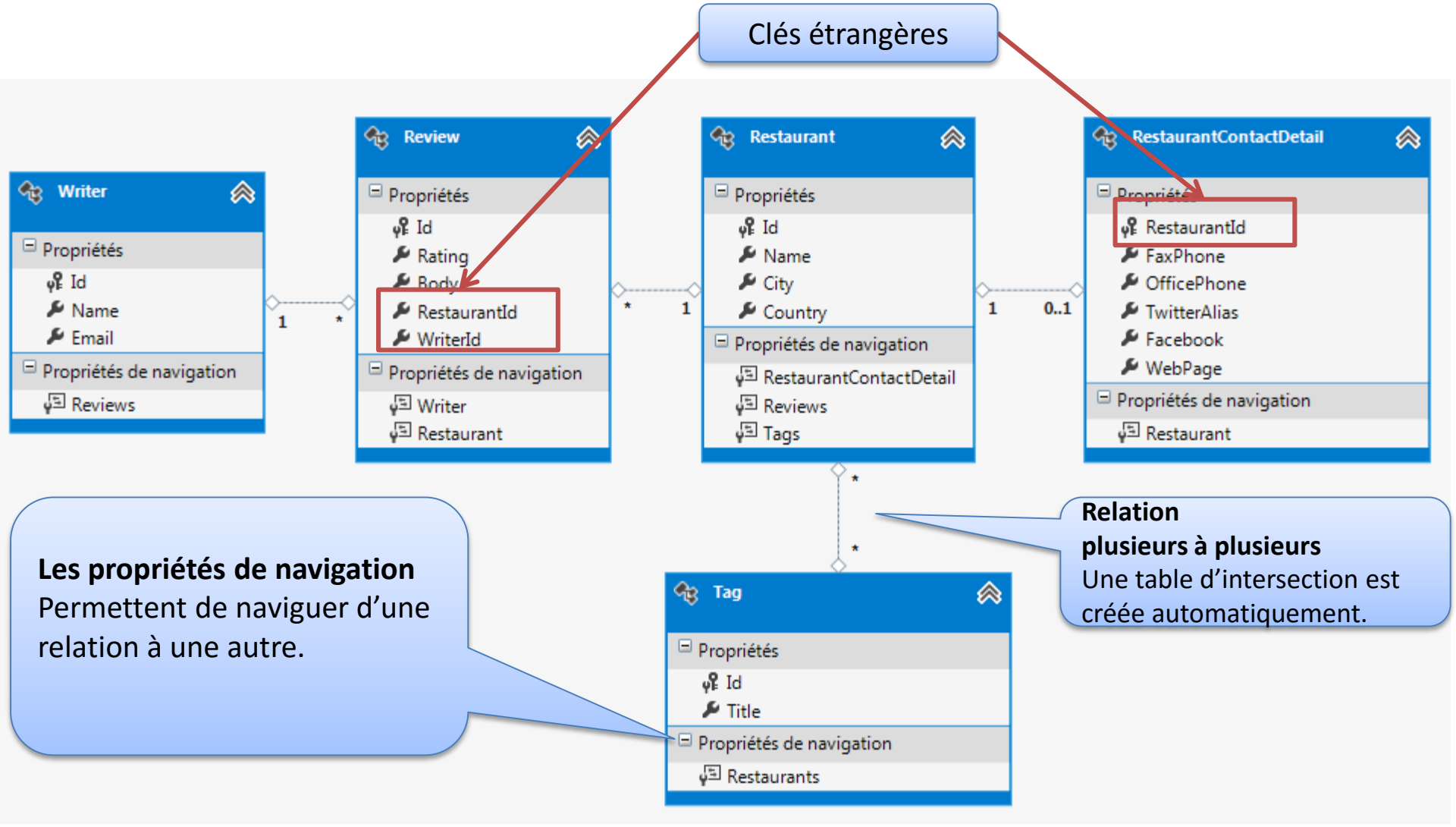


Code First

```
public class Restaurant
   {
       [Key]
       [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
       public int Id { get; set; }

       [Required]
       public string Name { get; set; }
       [Required]
       public string City { get; set; }
       [Required]
       public string Country { get; set; }

       //Navigation properties
       public virtual RestaurantContactDetail RestaurantContactDetail { get; set;}
       public virtual ICollection<Review> Reviews { get; set; }
       public virtual ICollection<Tag> Tags { get; set; }

       public Restaurant()
       {
           Reviews = new List<Review>();
           Tags = new List<Tag>();
       }
   }
```



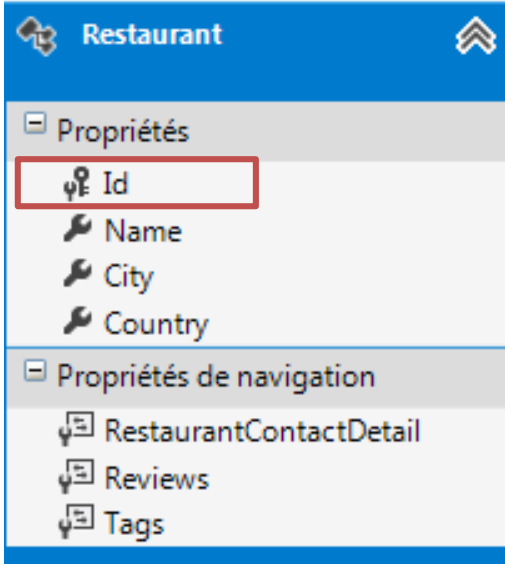
Représentation d'un modèle de données d'EF



Code first

La clé primaire

Annotation pour identifier la clé primaire



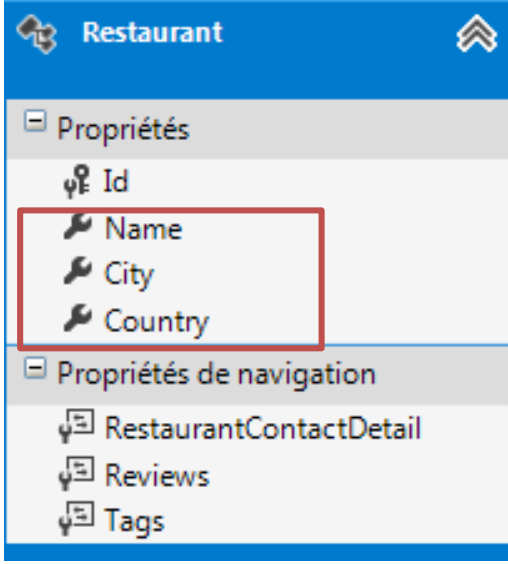

```
public class Restaurant
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [Required]
    public string Name { get; set; }
    [Required]
    public string City { get; set; }
    [Required]
    public string Country { get; set; }

    //Navigation properties
    public virtual RestaurantContactDetail RestaurantContactDetail { get; set;}
    public virtual ICollection<Review> Reviews { get; set; }
    public virtual ICollection<Tag> Tags { get; set; }

    public Restaurant()
    {
        Reviews = new List<Review>();
        Tags = new List<Tag>();
     }
}
```

Annotation qui indique que la clé sera générée par la BD

Code first

Les champs

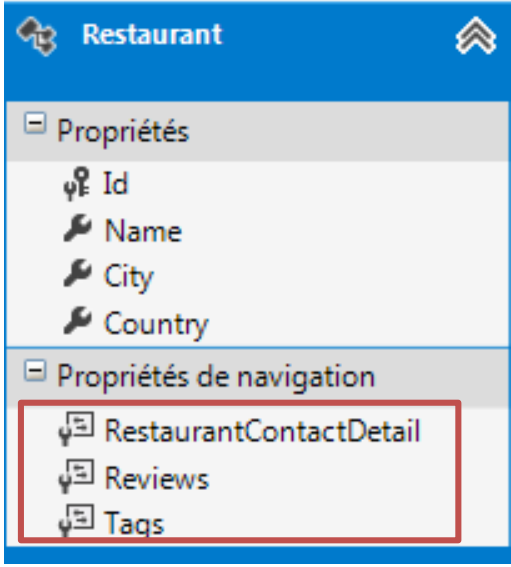
```
public class Restaurant
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [Required]
    public string Name { get; set; }
    [Required]
    public string City { get; set; }
    [Required]
    public string Country { get; set; }

    //Navigation properties
    public virtual RestaurantContactDetail RestaurantContactDetail { get; set;}
    public virtual ICollection<Review> Reviews { get; set; }
    public virtual ICollection<Tag> Tags { get; set; }

    public Restaurant()
    {
        Reviews = new List<Review>();
        Tags = new List<Tag>();
     }
}
```



L'annotation **[Required]** indique que le champs ne peut être à NULL dans la BD.

Code first

Les propriétés de navigation

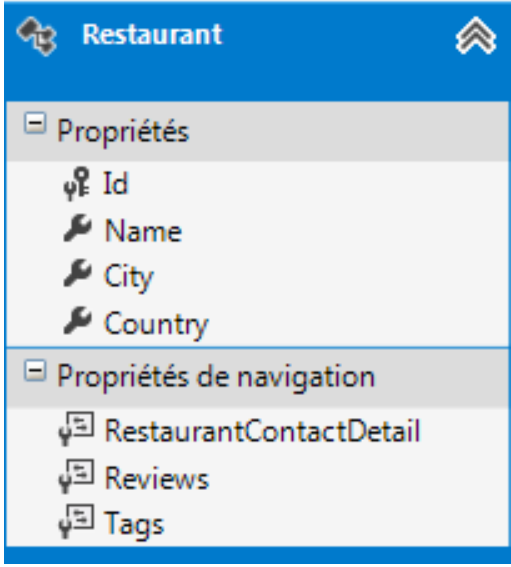



```
public class Restaurant
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [Required]
    public string Name { get; set; }
    [Required]
    public string City { get; set; }
    [Required]
    public string Country { get; set; }

    //Navigation properties
    public virtual RestaurantContactDetail RestaurantContactDetail { get; set;}
    public virtual ICollection<Review> Reviews { get; set; }
    public virtual ICollection<Tag> Tags { get; set; }

    public Restaurant()
    {
        Reviews = new List<Review>();
        Tags = new List<Tag>();
     }
}
```

Virtual est nécessaire pour le **lazy loading**

Code first

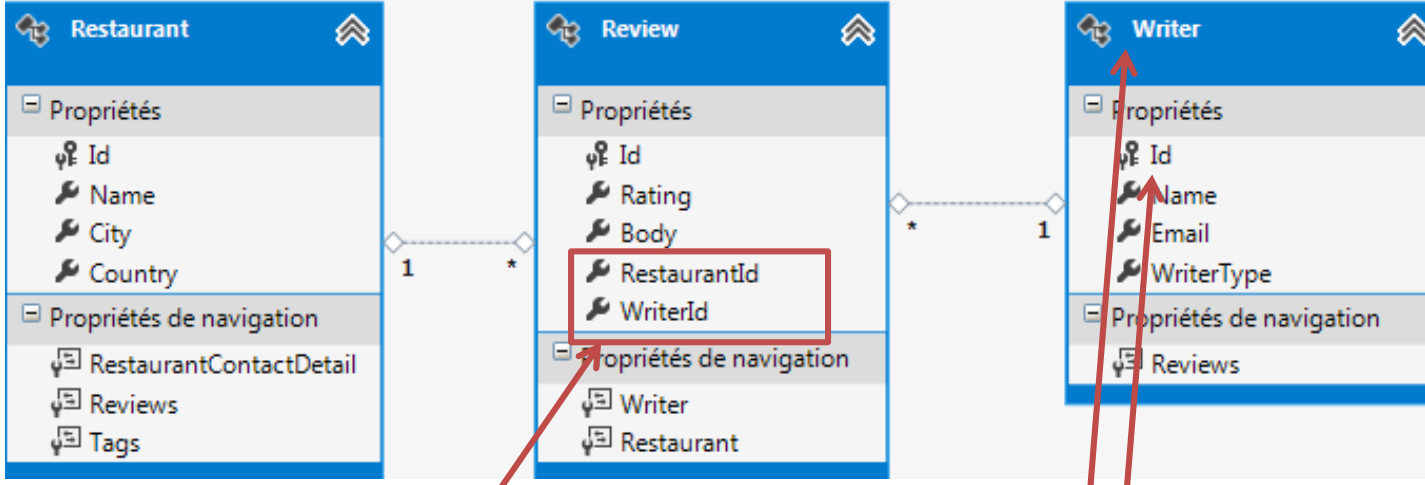
Initialisation des collections




```
public class Restaurant
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [Required]
    public string Name { get; set; }
    [Required]
    public string City { get; set; }
    [Required]
    public string Country { get; set; }

    //Navigation properties
    public virtual RestaurantContactDetail RestaurantContactDetail { get; set;}
    public virtual ICollection<Review> Reviews { get; set; }
    public virtual ICollection<Tag> Tags { get; set; }

    public Restaurant()
    {
        Reviews = new List<Review>();
        Tags = new List<Tag>();
     }
}
```

Code first

Les clés étrangères.




```
public class Review
{
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Range(1, 10)]
        [Required]
        public int Rating { get; set; }

        [StringLength(1024)]
        public string Body { get; set; }

        // Foreign key
        public int RestaurantId { get; set; }
        public int WriterId { get; set; }

        //Navigation properties
        public virtual Writer Writer { get; set; }
        public virtual Restaurant Restaurant { get; set; }
}
```

Par convention,
Le nom de la clé étrangère est:

le nom de l'entité

+

La clé primaire

Annotations

Exemples pour la validation

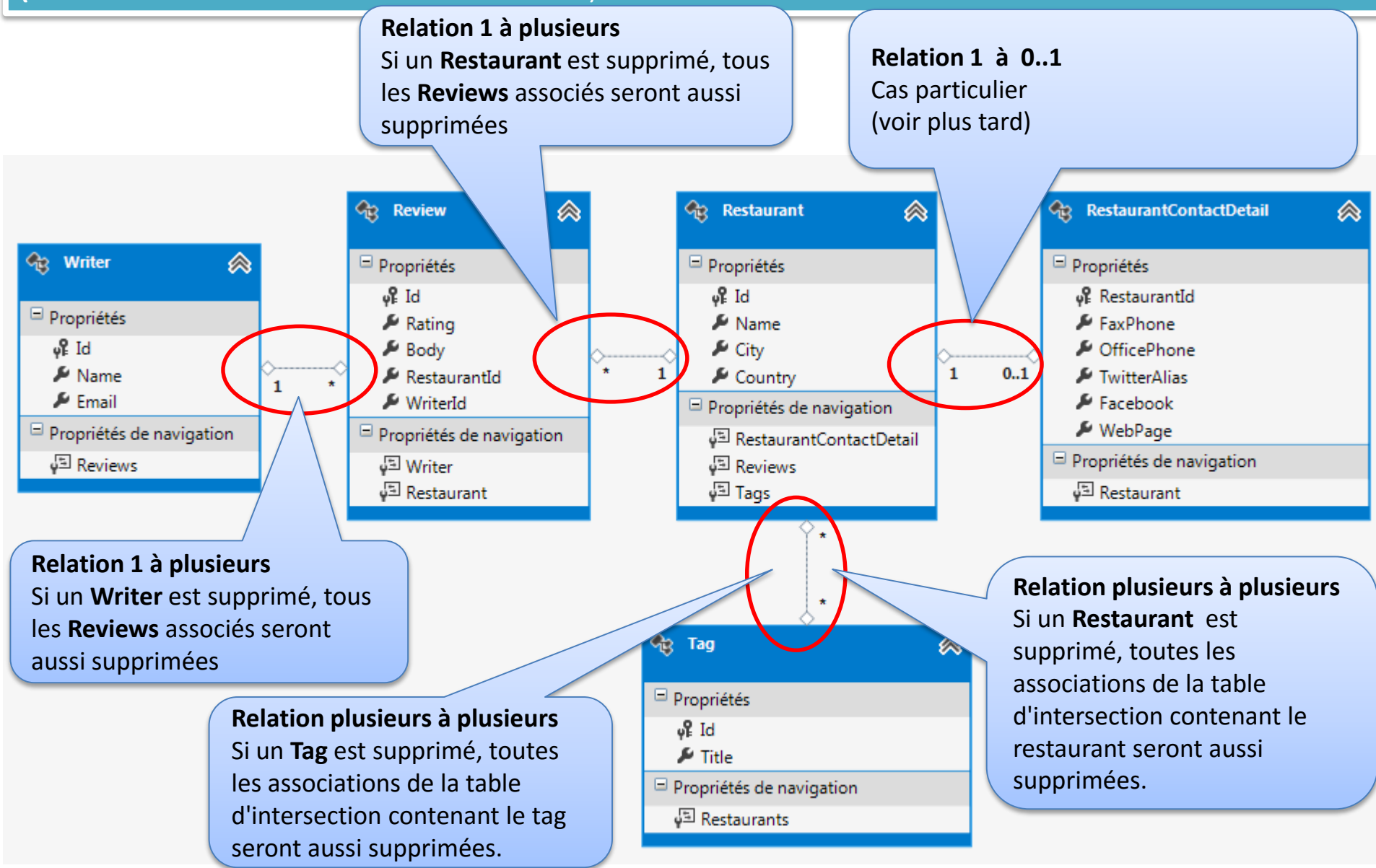
```
[Required]
[Range(1, 10)]
[StringLength(1024)]
[MinLength(5)]
[MaxLength(10)]
[MinLength(5), MaxLength(10)]
```

Il existe plusieurs autres types d'annotations. Exemples:

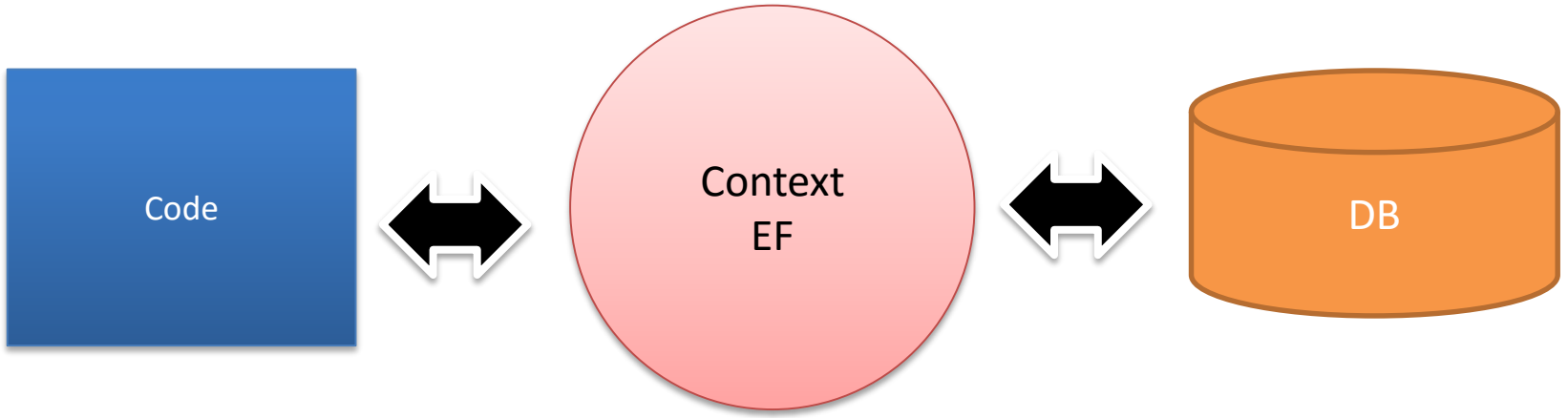
- Column
- Table
- ForeignKey
- DatabaseGenerated
- Etc.

La suppression en cascade

C'est lors de la création de la BD qu'EF indique les actions à prendre pour la suppression.
Par défaut, l'effacement en cascade est appliqué pour les relations **un à plusieurs** et **plusieurs à plusieurs** (avec la table d'intersection dans une relation).



La notion de *context*



EF communique avec la BD par l'intermédiaire d'un ***context***.

Créer l'objet *context*

```
public class MiamContext : DbContext
{
    public DbSet<Restaurant> Restaurants { get; set; }
    public DbSet<Review> Reviews { get; set; }
    public DbSet<Writer> Writers { get; set; }
    public DbSet<Tag> RestaurantTags { get; set; }
    public DbSet<RestaurantContactDetail> RestaurantContactDetails { get; set;}
}
```

Utiliser un *context*

Exemple: ajouter un restaurant dans la BD

```
var restaurant = new Restaurant()
                 {
                     City = "Québec",
                     Name = "La grenouille bouillie",
                     Country = "Canada"
                  };

var context = new MiamContext();
context.Restaurants.Add(restaurant);
context.SaveChanges();
```

Exemples de requêtes sur un *context*

Retourne le nombre de restaurants

```
var restautantCount = context.Restaurants.Count();
```

Retourne le nombre de restaurants qui respecte un critère

```
var restautantCount = context.Restaurants.Count(r => r.City.Contains("A"));
```

Retourne le seul restaurant ou lève une exception.

```
var restaurant = context.Restaurants.Single(r => r.Id == 10);
```

Retourne le premier élément d'une séquence ou lève une exception

```
var restaurant = context.Restaurants.First(r => r.City == "Québec")
```

Retourne le premier élément d'une séquence ou une valeur par défaut si ne contient rien.

```
var restaurant = context.Restaurants.FirstOrDefault(r => r.City == "Québec");
```

Retourne un ensemble (IQueryable) de restaurants répondant à la requête

```
var restaurants = context.Restaurants.Where(x => x.City.StartsWith("A"));
```

Exemples d'opérations sur un *context*

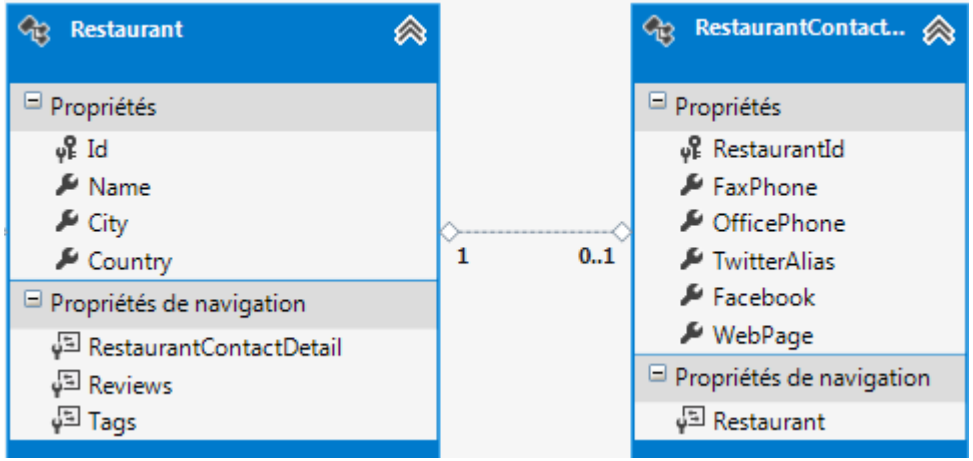
Mise à jour d'un restaurant dans la BD.

```
restaurant.Name = "La vache à Roger";
context.Restaurants.Attach(restaurant);
context.Entry(restaurant).State = System.Data.EntityState.Modified;
context.SaveChanges();
```

Suppression d'un restaurant dans la BD.

```
context.Restaurants.Attach(restaurant);
context.Restaurants.Remove(restaurant);
context.SaveChanges();
```

La suppression en cascade **Relation 1 – 0..1**



Contexte:

Dans la BD, un **RestaurantContactDetail** existe pour un **Restaurant**.

Problème:

Si on supprime un **Restaurant** une exception sera lancée par la BD (problème d'intégrité référentielle). La suppression du **RestaurantContactDetail** ne se fera pas automatiquement.

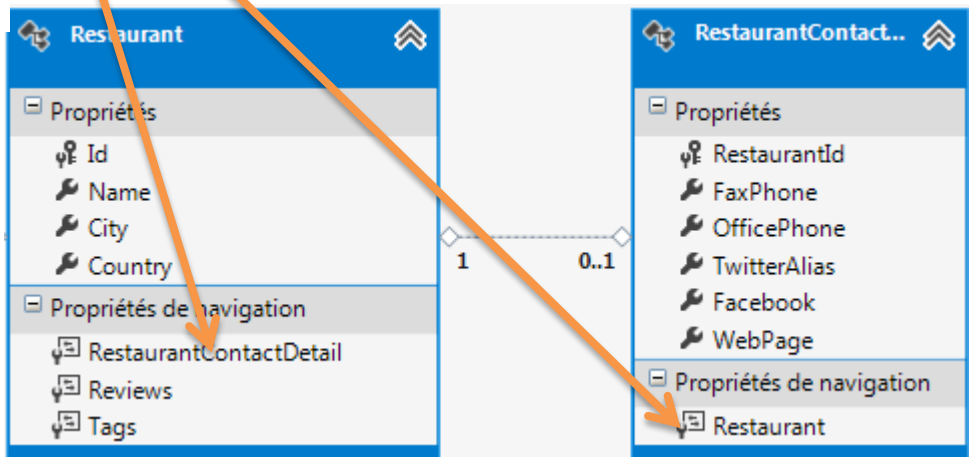
Dans ce cas particulier, on doit préciser à EF (avec **fluent API**) de faire la suppression du **RestaurantContatctDetail** associé au **Restaurant**. Voir page suivante.

Fluent API

Fluent API offre plus que les annotations.
Certains cas ne peuvent être gérés que par fluent api.

Exemple: Gestion de la suppression dans une relation 1 – 0..1

```
public class MiamContext : DbContext
{
  public DbSet<Restaurant> Restaurants { get; set; }
  public DbSet<Review> Reviews { get; set; }
  public DbSet<Writer> Writers { get; set; }
  public DbSet<Tag> RestaurantTags { get; set; }
  public DbSet<RestaurantContactDetail> RestaurantContactDetails { get; set;}

  protected override void OnModelCreating(DbModelBuilder modelBuilder)
  {
    modelBuilder.Entity<RestaurantContactDetail>().HasKey(c => c.RestaurantId);
    modelBuilder.Entity<RestaurantContactDetail>()
                .HasRequired(c => c.Restaurant)
                .WithOptional(r => r.RestaurantContactDetail)
                .WillCascadeOnDelete();
   base.OnModelCreating(modelBuilder);
 }
}
```



Configurer la chaine de connexion à la BD

Si aucun chaine de connexion n'est spécifiée pour un *context*, EF va créer une base de donnée par défaut.

Dans App.Config (ou web.config) , il est possible de préciser la chaine de connexion.
Exemple:

```
<connectionStrings>
   <add name="MiamContext" connectionString="
    Data Source=.\SQLEXPRESS;
    Initial Catalog=uneBaseDeDonnees;
    user id=ymazieres;
    password=ymazieres;"
    providerName="System.Data.SqlClient" />
 </connectionStrings>
```

Initialiser la base de données

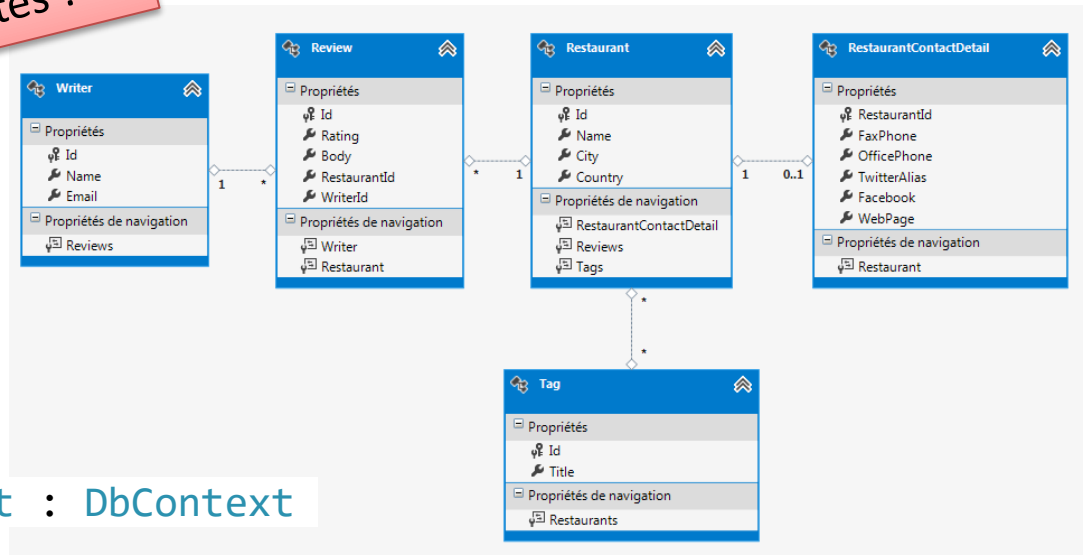
```
Database.SetInitializer(new DropCreateDatabaseIfModelChanges<MiamContext>());
```

Autres exemples: DropCreateDatabaseAlways, MigrateDatabaseToLatestVersion, CreateDatabaseIfNotExists, etc.

Entity Framework Powertool Beta 4

- Est une extension à ajouter à VS2013.
- Permet d'afficher la représentation visuelle des relations entre les entités (clic droit de la souris sur la classe qui hérite de DbContext).

Très utile pour valider les relations entre les entités !

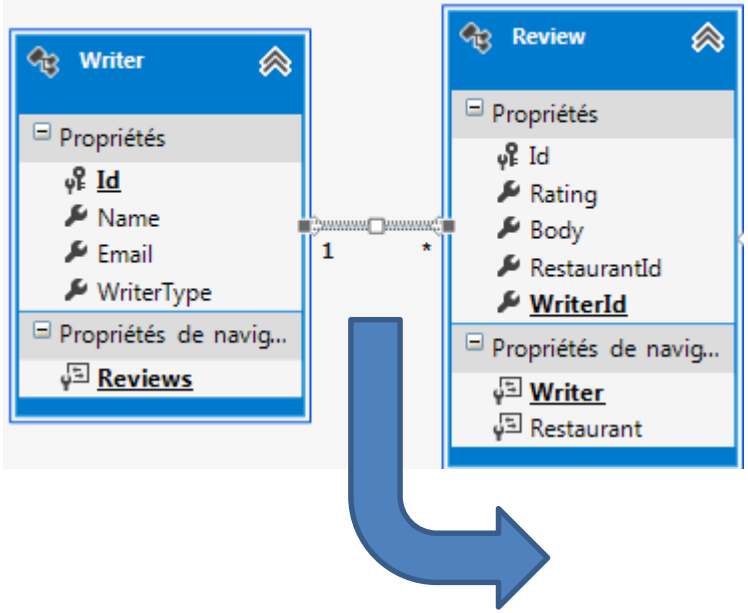


Miamcontext.cs

```
public class MiamContext : DbContext
{
    public DbSet<Restaurant> Restaurants { get; set; }
    public DbSet<Review> Reviews { get; set; }
    public DbSet<Writer> Writers { get; set; }
    public DbSet<Tag> RestaurantTags { get; set; }
    public DbSet<RestaurantContactDetail> RestaurantContactDetails { get; set;}
}
```

Entity Framework Powertool Beta 4

Permet de voir les actions à prendre lors de la suppression (cascade delete ou non)



Propriétés

Miam.DataLayer.Writer_Reviews Association

Contraintes	
Contrainte référentielle	Writer -> Review
Général	
Documentation	
Multiplicité End1	**1 (Un de Writer)**
Multiplicité End2	*** (Collection de Review)**
Nom	**Writer_Reviews**
Nom de l'ensemble d'associations	**Writer_Reviews**
Nom du rôle End1	**Writer_Reviews_Source**
Nom du rôle End2	**Writer_Reviews_Target**
OnDelete End1	**Cascade**
OnDelete End2	None
Propriété de navigation End1	**Reviews**
Propriété de navigation End2	**Writer**