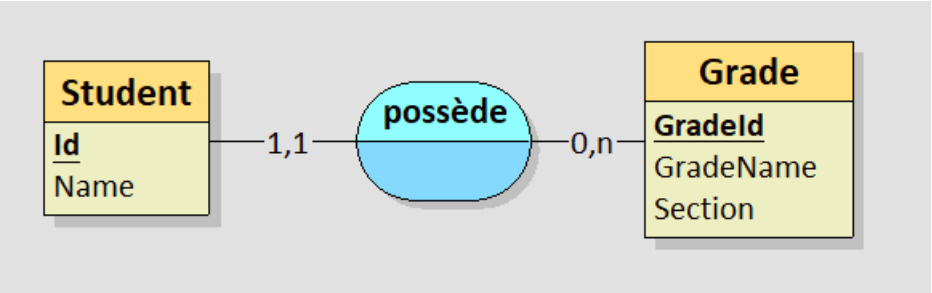
# Entity Framework : Configuration des relations entre les tables

Pour que le modèle d'Entity Framework fonctionne correctement, il faut lui indiquer les relations entre les tables, ainsi que les clés primaires et étrangères si elles ne sont pas notées de la même façon que les attributs dans les objets.

Dans cette présentation, pour simplifier les modelbuilder, on partira du principe que les noms des colonnes de la table sont identiques aux noms des propriétés des classes.

## Relation un à plusieurs (One-to-many)

MCD correspondant



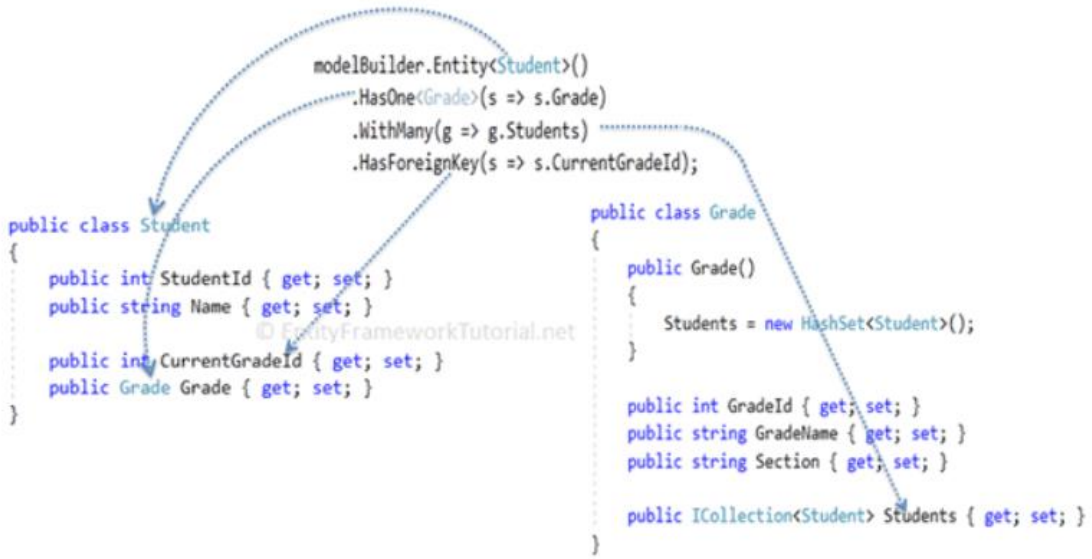

MLD Correspondant


Objets correspondants

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }

    public int CurrentGradeId { get; set; }
    public Grade Grade { get; set; }
}

public class Grade
{
    public int GradeId { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

    public ICollection<Student> Students { get; set; }
}
```

Configuration du ModelBuilder

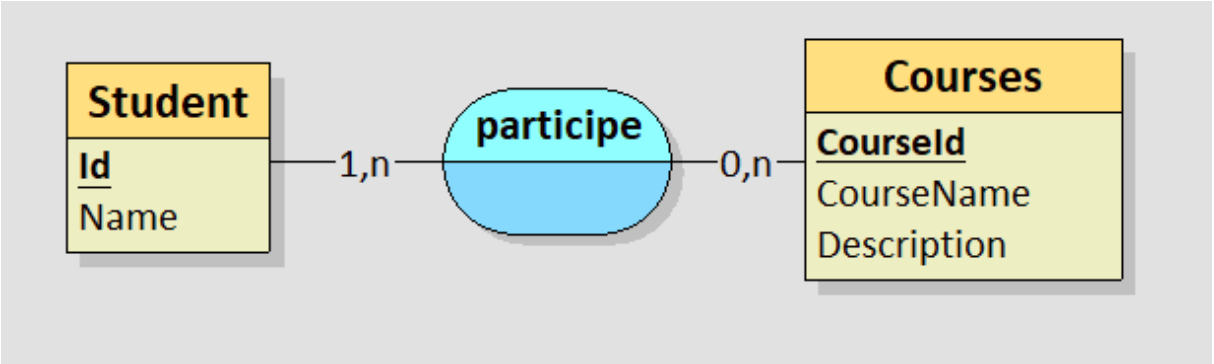
```
modelBuilder.Entity<Student>()
    .HasOne<Grade>(s => s.Grade)
    .WithMany(g => g.Students)
    .HasForeignKey(s => s.CurrentGradeId);
```



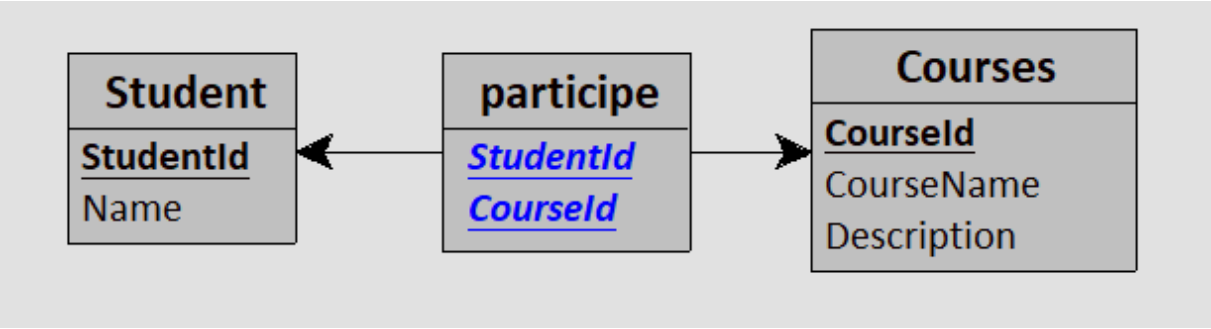

« Traduction » : Un étudiant possède un grade (stocké dans la propriété Student.Grade) qui est en relation avec plusieurs étudiants (stocké dans la propriété Grade.Students) selon la clé étrangère (CurrentGradeId)

## Relation plusieurs à plusieurs (Many-to-many)

MCD correspondant




MLD correspondant




Objets correspondants

```
public class Student
{
    public int StudentId { get; set; }
    public string Name { get; set; }

    public IList<StudentCourse> StudentCourses { get; set; }
}

public class Course
{
    public int CourseId { get; set; }
    public string CourseName { get; set; }
    public string Description { get; set; }

    public IList<StudentCourse> StudentCourses { get; set; }
}
```

Configuration du ModelBuilder

Si la table associative ne contient pas de clé unique (c'est-à-dire que c'est toujours une clé primaire double), il faut le déclarer à Entity Framework

```
        modelBuilder.Entity<StudentCourse>().HasKey(sc => new { sc.StudentId, sc.CourseId });
```

La relation Many-to-many correspond à 2 relations One-to-many

On crée donc 3 DbSet:

```
public DbSet<Student> Students { get; set; }
public DbSet<Course> Courses { get; set; }
public DbSet<StudentCourse> StudentCourses { get; set; }
```

```
modelBuilder.Entity<StudentCourse>()
    .HasOne<Student>(sc => sc.Student)
    .WithMany(s => s.StudentCourses)
    .HasForeignKey(sc => sc.SId);


modelBuilder.Entity<StudentCourse>()
    .HasOne<Course>(sc => sc.Course)
    .WithMany(s => s.StudentCourses)
    .HasForeignKey(sc => sc.CId);
```

## Utilisation d'Entity Framework scaffold pour générer les relations entre les tables

- Mettre en place une base de données avec les tables et les contraintes de clés étrangères
- Créer un projet
- Ajouter les packages nécessaires
- Ajouter l'automapper dans startup
- Mettre à jour appsettings.json avec la ConnectionStrings
- Lancer le scaffold sur la base
- Dans le fichier Data/Models/ un fichier context a été créé avec les relations entre les tables et les contraintes sur les colonnes.
- Ajouter le context au startup
- Créer les DTOs en entrée et en sortie
  Ajouter en plus des DTOS partiels pour les tables associatives (ex AvecEntité1, AvecEntité2, AvecEntité1EtEntité2)
- Créer les Profiles correspondants
- Créer les services (utilisation du snippet)
- Ajouter les services au startup
- Créer les controllers (utilisation du snippet)
- Modifier les services (getAll et getById) en incluant les tables liées

Ajouter les `.Include("NomDeAttributInclus")` pour les relations One-To-Many
Ajouter un niveau de plus pour les tables associatives

```
.Include("StudentsCourses.Course")
```