Previous Next

One-to-Many Relationship Conventions in Entity Framework Core

In the previous chapter, you learned about the EF conventions which map entities to different objects of the database. Here, you will learn about the relationship conventions between two entity classes that result in one-to-many relationships between corresponding tables in the database.

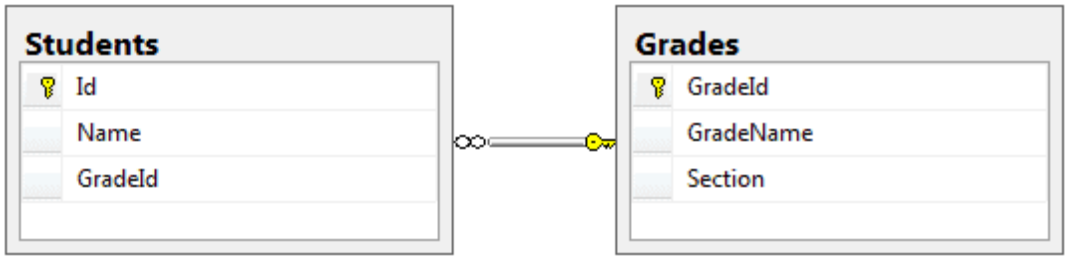
Entity Framework Core follows the same convention as Entity Framework 6.x conventions for one-to-many relationship. The only difference is that EF Core creates a foreign key column with the same name as navigation property name and not as `<NavigationPropertyName>_<PrimaryKeyPropertyName>`

Let's look at the different conventions which automatically configure a one-to-many relationship between the following `Student` and `Grade` entities.

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

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

After applying the conventions for one-to-many relationship in the entities above, the database tables for `Student` and `Grade` entities will look like below, where the `Students` table includes a foreign key `GradeId`.



Convention 1

We want to establish a one-to-many relationship where many students are associated with one grade. This can be achieved by including a reference navigation property in the dependent entity as shown below. (here, the `Student` entity is the dependent entity and the `Grade` entity is the principal entity).

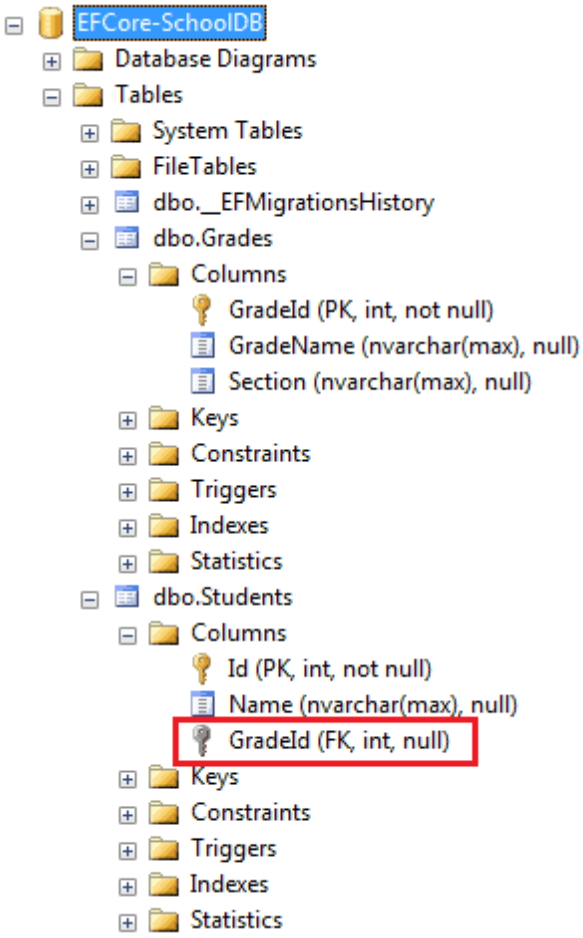
```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }

    public Grade Grade { get; set; }
}

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

In the example above, the `Student` entity class includes a reference navigation property of `Grade` type. This allows us to link the same `Grade` to many different `Student` entities, which creates a one-to-many relationship between them. This will produce a one-to-many relationship between the `Students` and `Grades` tables in the database, where `Students` table includes a nullable foreign key `GradeId`, as shown

below. EF Core will create a shadow property for the foreign key named `GradeId` in the conceptual model, which will be mapped to the `GradeId` foreign key column in the `Students` table.



Note: The reference property `Grade` is nullable, so it creates a nullable ForeignKey `GradeId` in the `Students` table. You can configure NotNull foreign keys using fluent API.

Convention 2

Another convention is to include a collection navigation property in the principal entity as shown below.

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

In the example above, the `Grade` entity includes a collection navigation property of type `ICollection<student>`. This will allow us to add multiple `Student` entities to a `Grade` entity, which results in a one-to-many relationship between `Students` and `Grades` tables in the database, same as in convention 1.

Convention 3

Another EF convention for the one-to-many relationship is to include navigation property at both ends, which will also result in a one-to-many relationship (convention 1 + convention 2).

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

    public Grade Grade { get; set; }
}

public class Grade
{
    public int GradeID { get; set; }
    public string GradeName { get; set; }

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

In the example above, the `Student` entity includes a reference navigation property of `Grade` type and the `Grade` entity class includes a collection navigation property `ICollection<Student>`, which results in a one-to-many relationship between corresponding database tables `Students` and `Grades`, same as in convention 1.

Convention 4

Defining the relationship fully at both ends with the foreign key property in the dependent entity creates a one-to-many relationship.

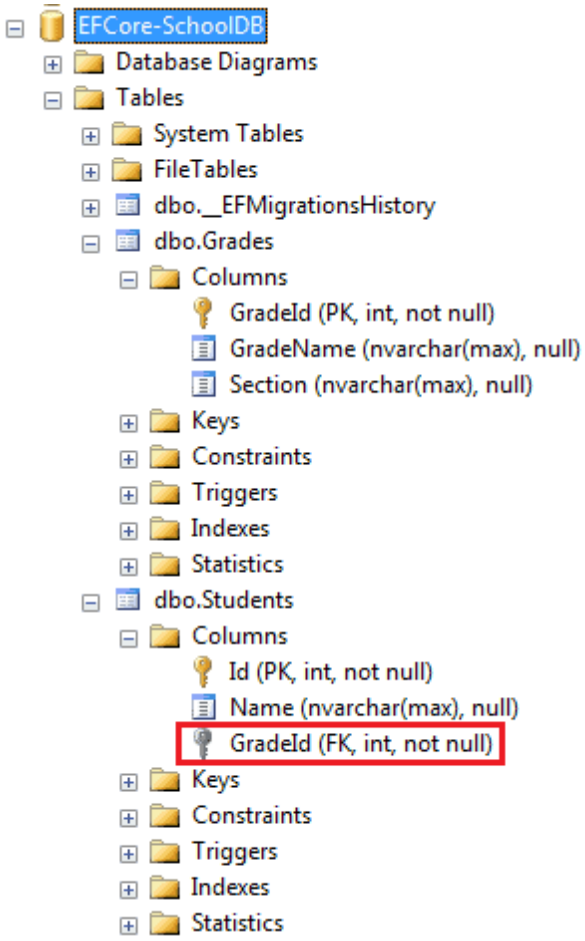
```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }

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

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

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

In the above example, the `Student` entity includes a foreign key property `GradeId` of type `int` and its reference navigation property `Grade`. At the other end, the `Grade` entity also includes a collection navigation property `ICollection<Student>`. This will create a one-to-many relationship with the NotNull foreign key column in the `Students` table, as shown below.



If you want to make the foreign key `GradeId` as nullable, then use nullable int data type (`Nullable<int> or int?`), as shown below.

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

    public int? GradeId { get; set; }
    public Grade Grade { get; set; }
}
```

Therefore, these are the conventions which automatically create a one-to-many relationship in the corresponding database tables. If entities do not follow the above conventions, then you can use Fluent API to configure the one-to-many relationship.

Further Reading

- Configure one-to-many relationship using Fluent API
- Configure one-to-one relationship using Fluent API

- Configure many-to-many relationship using Fluent API

Previous Next

ENTITYFRAMEWORKTUTORIAL

Learn Entity Framework using simple yet practical examples on EntityFrameworkTutorial.net for free. Learn Entity Framework DB-First, Code-First and EF Core step by step. While using this site, you agree to have read and accepted our terms of use and privacy policy.

feedback@entityframeworktutorial.net

TUTORIALS

E-MAIL LIST

Subscribe to EntityFrameworkTutorial email list and get EF 6 and EF Core Cheat Sheets, latest updates, tips & tricks about Entity Framework to your inbox.