< Previous

Next >

What is an Entity in Entity Framework?

An entity in Entity Framework is a class that maps to a database table. This class must be included as a `DbSet<TEntity>` type property in the `DbContext` class. EF API maps each entity to a table and each property of an entity to a column in the database.

For example, the following `Student`, and `Grade` are domain classes in the school application.

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public DateTime? DateOfBirth { get; set; }
    public byte[]  Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { 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; }
}
```

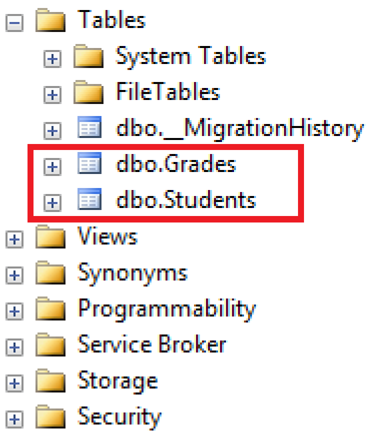
The above classes become entities when they are included as `DbSet<TEntity>` properties in a context class (the class which derives from `DbContext`), as shown below.

```
public class SchoolContext : DbContext
{
    public SchoolContext()
    {

    }

    public DbSet<Student> Students { get; set; }
    public DbSet<Grade> Grades { get; set; }
}
```

In the above context class, `Students`, and `Grades` properties of type `DbSet<TEntity>` are called entity sets. The `Student`, and `Grade` are entities. EF API will create the `Students` and `Grades` tables in the database, as shown below.



An Entity can include two types of properties: Scalar Properties and Navigation Properties.

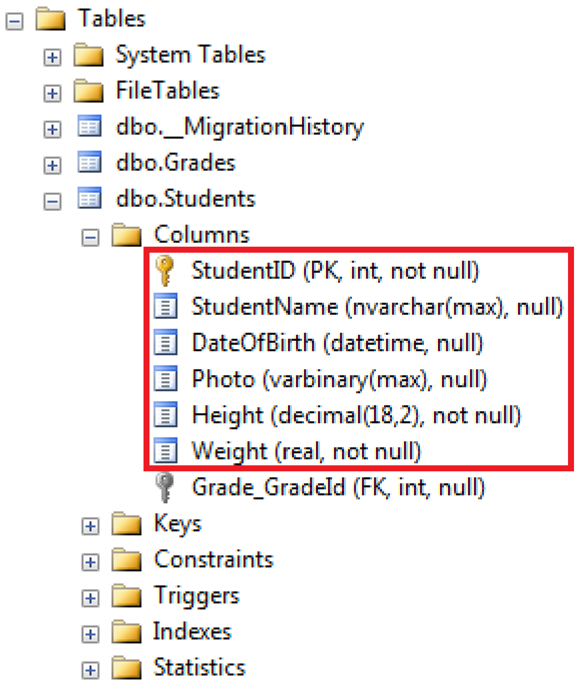
Scalar Property

The primitive type properties are called scalar properties. Each scalar property maps to a column in the database table which stores an actual data. For example, `StudentID, StudentName, DateOfBirth, Photo, Height, Weight` are the scalar properties in the `Student` entity class.

```
public class Student
{
    // scalar properties
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public DateTime? DateOfBirth { get; set; }
    public byte[]  Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    //reference navigation properties
    public Grade Grade { get; set; }
}
```

EF API will create a column in the database table for each scalar property, as shown below.



```
⊟ Tables
    ⊞ System Tables
    ⊞ FileTables
    ⊞ dbo.__MigrationHistory
    ⊞ dbo.Grades
    ⊟ dbo.Students
        ⊟ Columns
              StudentID (PK, int, not null)
              StudentName (nvarchar(max), null)
              DateOfBirth (datetime, null)
              Photo (varbinary(max), null)
              Height (decimal(18,2), not null)
              Weight (real, not null)
              Grade_GradeId (FK, int, null)
        ⊞ Keys
        ⊞ Constraints
        ⊞ Triggers
        ⊞ Indexes
        ⊞ Statistics
```

Navigation Property

The navigation property represents a relationship to another entity.

There are two types of navigation properties: Reference Navigation and Collection Navigation

Reference Navigation Property

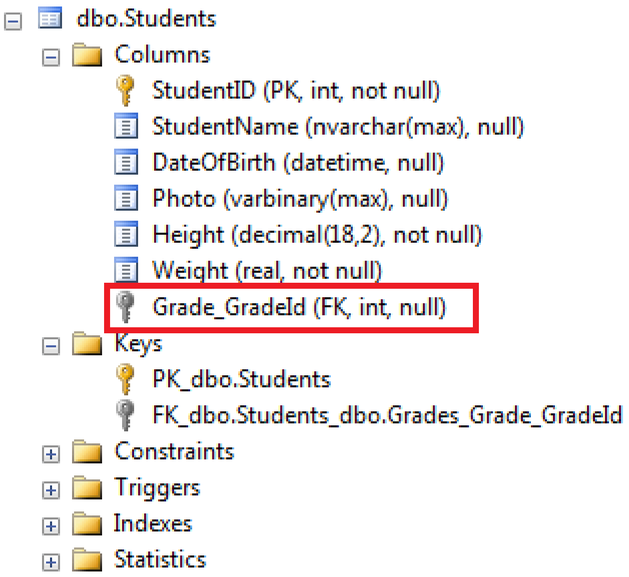
If an entity includes a property of another entity type, it is called a Reference Navigation Property. It points to a single entity and represents multiplicity of one (1) in the entity relationships.

EF API will create a ForeignKey column in the table for the navigation properties that points to a PrimaryKey of another table in the database. For example, `Grade` are reference navigation properties in the following `Student` entity class.

```
public class Student
{
    // scalar properties
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public DateTime? DateOfBirth { get; set; }
    public byte[]  Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

    //reference navigation property
    public Grade Grade { get; set; }
}
```

In the database, EF API will create a ForeignKey `Grade_GradeId` in the `Students` table, as shown below.

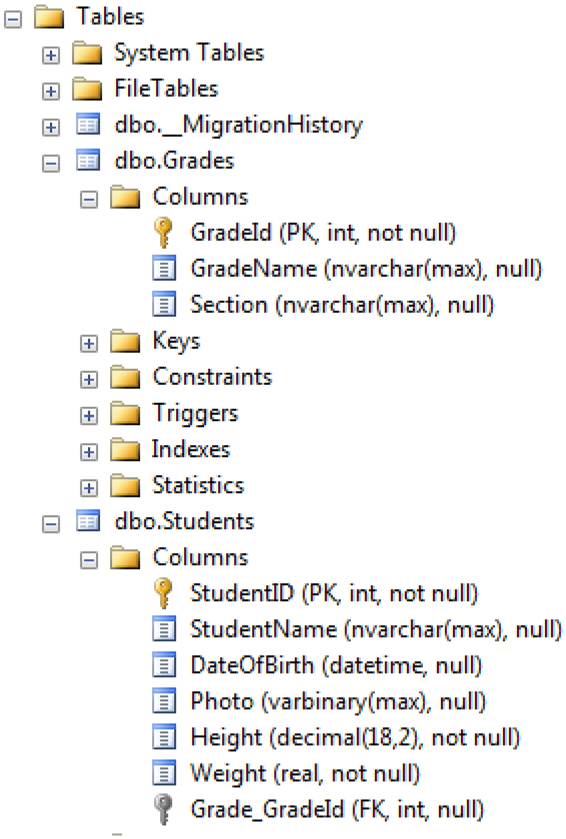


```
dbo.Students
    Columns
        StudentID (PK, int, not null)
        StudentName (nvarchar(max), null)
        DateOfBirth (datetime, null)
        Photo (varbinary(max), null)
        Height (decimal(18,2), not null)
        Weight (real, not null)
        Grade_GradeId (FK, int, null)
    Keys
        PK_dbo.Students
        FK_dbo.Students_dbo.Grades_Grade_GradeId
    Constraints
    Triggers
    Indexes
    Statistics
```

Collection Navigation Property

If an entity includes a property of generic collection of an entity type, it is called a collection navigation property. It represents multiplicity of many (*).

EF API does not create any column for the collection navigation property in the related table of an entity, but it creates a column in the table of an entity of generic collection. For example, the following `Grade` entity contains a generic collection navigation property `ICollection<Student>`. Here, the `Student` entity is specified as generic type, so EF API will create a column `Grade_GradeId` in the `Students` table in the database.



```
⊟ Tables
    ⊞ System Tables
    ⊞ FileTables
    ⊞ dbo.__MigrationHistory
    ⊟ dbo.Grades
        ⊟ Columns
              GradeId (PK, int, not null)
              GradeName (nvarchar(max), null)
              Section (nvarchar(max), null)
        ⊞ Keys
        ⊞ Constraints
        ⊞ Triggers
        ⊞ Indexes
        ⊞ Statistics
    ⊟ dbo.Students
        ⊟ Columns
              StudentID (PK, int, not null)
              StudentName (nvarchar(max), null)
              DateOfBirth (datetime, null)
              Photo (varbinary(max), null)
              Height (decimal(18,2), not null)
              Weight (real, not null)
              Grade_GradeId (FK, int, null)
```

Learn more about how the navigation properties plays an important role in [defining entity relationships](#).

< 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

- EF Basics
- EF Core
- EF 6 DB-First
- EF 6 Code-First

E-MAIL LIST

HOME PRIVACY POLICY ADVERTISE WITH US