Cách tạo Table liên kết với nhau trong Model (Visual)

// Để nối hai bảng lại với nhau thì nên chỉ có một đường nối duy nhất giữa các khóa chính và khóa ngoại tránh bị lỗi

**// note : khóa ngoại hay Class nối sau kiểu dữ liệu thêm dấu chấm ?**

**// trong Model hai class join vs nhau thì Class này chứa Navigation của class kia và ngược lại để có thể join table theo hai chiều.**

**// Navigation property = 1 thuộc tính trong model C# để EF Core biết và để bạn đi từ Class này sang Class kia để join vs nhau thông qua khóa chính và khóa ngoại**

**//Tên khóa ngoại : bằng tên biến Navigation property + Id**

**// Navigation Property : cho biết khóa ngoại tham chiếu đến bảng nào**

**// Quan hệ 1-1 hay 1-N : 1 và N nằm bên phải đại điện bảng tới được**

**Nối thông qua khóa chính và khóa ngoại**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Item.cs (1-1 với SerialNumber) // quan hệ 1- 1 chuẩn**

* Mỗi bản ghi ở bảng A chỉ liên kết với 1 bản ghi ở bảng B và ngược lại.

public class Item

{

[Key]

public int ItemId { get; set; } // Khóa chính

public string Name { get; set; }

public decimal Price { get; set; }

// Navigation property: 1 Item chỉ có 1 SerialNumber

public SerialNumber ? SerialNumber { get; set; }

}

**SerialNumber.cs**

public class SerialNumber

{

[Key]

public int SerialNumberId { get; set; } // Khóa chính

public string Code { get; set; } // Serial code

// Khóa ngoại

public int ItemId { get; set; }

// Navigation property

[ForeignKey("ItemId")]

public Item Item { get; set; }

}

**🔹 Giải thích**

* **Item chỉ có một SerialNumber** → mối quan hệ **1-1**
* **SerialNumber vẫn có ItemId** làm khóa ngoại → liên kết với Item

SQL (1-1)

CREATE TABLE Item (

ItemId INT PRIMARY KEY,

Name NVARCHAR(100),

Price DECIMAL(18,2)

);

CREATE TABLE SerialNumber (

SerialNumberId INT PRIMARY KEY,

Code NVARCHAR(50),

ItemId INT UNIQUE, -- UNIQUE để đảm bảo 1-1

FOREIGN KEY (ItemId) REFERENCES Item(ItemId)

);

A screenshot of a computer screen

AI-generated content may be incorrect.

Result join

**Var items = \_context .Item .Include (s=> s.SerialNumber)**

A screenshot of a number

AI-generated content may be incorrect.

-----------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------

**Item.cs (1 - nhiều )**

* **Một bản ghi ở bảng A có thể liên kết với nhiều bản ghi ở bảng B, nhưng mỗi bản ghi ở bảng B chỉ liên kết với 1 bản ghi ở bảng A.**

public class Item

{

[Key]

public int ItemId { get; set; } // Khóa chính

public string Name { get; set; }

public decimal Price { get; set; }

// Navigation property: 1 Item có nhiều SerialNumbers

public List<SerialNumber> ? SerialNumbers { get; set; } = new List<SerialNumber> () ;

}

public class SerialNumber

{

[Key]

public int SerialNumberId { get; set; } // Khóa chính

public string Code { get; set; } // Serial code

// Khóa ngoại

public int ItemId { get; set; }

// Navigation property

[ForeignKey("ItemId")]

public Item Item { get; set; }

}

SQL (1-nhiều )

CREATE TABLE Item (

ItemId INT PRIMARY KEY,

Name NVARCHAR(100),

Price DECIMAL(18,2)

);

CREATE TABLE SerialNumber (

SerialNumberId INT PRIMARY KEY,

Code NVARCHAR(50),

ItemId INT,

FOREIGN KEY (ItemId) REFERENCES Item(ItemId)

);

A screenshot of a computer

AI-generated content may be incorrect.

Many To Many (n-n) là sự kết hợp từ (1-n) và ( 1-n )

* Bảng A liên kết nhiều bản ghi với bảng B, và ngược lại.

public class Student

{

public int StudentId { get; set; }

public string Name { get; set; }

public List<Student\_Course>? Student\_Courses { get; set; } = new List<Student\_Course>();

}

public class Course

{

public int CourseId { get; set; }

public string Name { get; set; }

public List<Student\_Course>? Student\_Courses { get; set; } = new List<Student\_Course>();

}

public class Student\_Course {

public int StudentId { get; set; } // Khóa ngoại đến Student

public Student? Student { get; set; } // Navigation property đến Student

public int CourseId { get; set; } // Khóa ngoại đến Course

public Course? Course { get; set; } // Navigation property đến Course

}

SQL

INSERT INTO Students ( Name) VALUES

( 'Alice'),

( 'Bob'),

( 'Carol');

INSERT INTO Courses ( CourseId, Name) VALUES

(101, 'Toán'),

(102, 'Vật lý'),

(103, 'Hóa học');

INSERT INTO Student\_Courses (StudentId, CourseId) VALUES

(4, 101),

(5, 102),

(6, 101),

(6, 102),

(4, 103);

CREATE TABLE Student\_Course (

StudentId INT NOT NULL,

CourseId INT NOT NULL,

PRIMARY KEY (StudentId, CourseId),

FOREIGN KEY (StudentId) REFERENCES Student(StudentId),

FOREIGN KEY (CourseId) REFERENCES Course(CourseId)

);

Student

StudentId Name // khóa chính StudentId

1 Alice

2 Bob

3 Carol

Course

CourseId Name // khóa chính CourseId

101 Toán

102 Vật lý

103 Hóa học

Student\_Course // Table trung gian

StudentId CourseId // khóa chính (StudentId , CourseId )

1 101

1 102

2 101

3 102

3 103

Join Table

StudentId Name CourseId Name

1 Alice 101 Toán

1 Alice 102 Vật lý

2 Bob 101 Toán

3 Carol 102 Vật lý

3 Carol 103 Hóa học

Các quy tắc đặt khóa chính khóa ngoại trong Model

public class Item {

public int ItemId { get; set; } // PK quy tắc ClassChinh + Id = khóa chính

public string Name { get; set; }

public int SerialNumberId { get; set; } // FK (cột trong DB) ClassNavigationProperty + Id = khóa ngoại // có thể thêm : [ForeignKey (" SerialNumberId")] để chỉ rõ khóa ngoại

// 👉 Đây là Navigation Property : Navigation Property nói cho biết là khóa ngoại tham chiếu từ bảng nào

public SerialNumber SerialNumber { get; set; }

}

public class SerialNumber

{

public int serialNumberId { get; set; } // EF sẽ nhận đây là PK

public string Name { get; set; }

public int ItemId { get; set; } // FK

[ForeignKey ("ItemId")]

public Item Item { get; set; }

}

CREATE TABLE SerialNumber (

SerialNumberId INT PRIMARY KEY,

Code NVARCHAR(50),

ItemId INT,

FOREIGN KEY (ItemId) REFERENCES Item(ItemId) // ItemId trong Table SerialNumber được tham chiếu từ ItemId trong Table Item

);

A screenshot of a computer

AI-generated content may be incorrect.

SerialNumber.serialNumberId [ khóa chính ] = Item.serialNumberId [khóa ngoại]

( SerialNumber-> Item ).

Item.ItemId [khóa chính] = SerialNumber.ItemId [khóa ngoại]

( Item-> SerialNumber).

**Cách dùng lệnh Join trong LINQ để Join nhiều bảng lại với nhau**

**Note : cùng bậc thì ghi cái nào trước cũng được còn khác thì thì đi tuần tự**

**🗂️ Bảng trong database**

**Item**

| **ItemId** | **Name** | **Price** | **SerialNumberId** |
| --- | --- | --- | --- |
| 1 | Bút | 5000 | 10 |
| 2 | Thước | 3000 | NULL |
| 3 | Vở | 7000 | 12 |

**Class Item**

**Public int ItemId // khóa chính**

**Public string Name**

**Public Double Price**

**Public int SerialNumberId // khóa ngoại**

**Public SerialNumber SerialNumber // Navigation Property**

**SerialNumber**

| **SerialNumberId** | **Name** | **CategoryId** |
| --- | --- | --- |
| 10 | SN-BUT-001 | 100 |
| 12 | SN-VO-777 | 100 |

**Class SerialNumber**

**Public int SerialNumberId // khóa chính**

**Public string Name**

**Public int CategoryId // khóa ngoại**

**Public Category Category // Navigation Property**

**Category**

| **CategoryId** | **Name** |
| --- | --- |
| 100 | Học tập |

**Class Category**

**Public int CategoryId // khóa chính**

**Public string Name**

A piece of paper with writing on it

AI-generated content may be incorrect.

// xuất phát từ Items

var items = await \_context.Items

.Include(i => i.SerialNumber) // Item → SerialNumber

.ThenInclude(sn => sn.Category) // SerialNumber → Category

.ToListAsync();

//xuất phát từ serialNumber

var serialNumber = await \_context.SerialNumbers

.Include(sn => sn.Category) // SerialNumber → Category

.Include(sn => sn.Item) // SerialNumber → Item

.ToListAsync();

//xuất phát từ category

var categories = await \_context.Categories

.Include(c => c.SerialNumbers) // Category → SerialNumbers

.ThenInclude(sn => sn.Item) // SerialNumber → Item

.ToListAsync();

**Note : điều kiện Column được Null hay Not Null trong Model và Mapping tới Database trong mvc**

public class BrandModel

{

public string Name { get; set; } // NOT NULL trong database

public string Slug { get; set; } // NOT NULL trong database

public string Description { get; set; } // NOT NULL trong database

public int Status { get; set; } // NOT NULL trong database

}

BrandModel brand = new BrandModel {

Name = "dfsf",

Slug = "ttt",

Description = "good product"

// ❌ THIẾU Status - mặc định = 0 nhưng database có thể không chấp nhận

};

SQL

CREATE TABLE Brands (

Id INT PRIMARY KEY,

Name NVARCHAR(100) NOT NULL, -- Bắt buộc

Slug NVARCHAR(100) NOT NULL, -- Bắt buộc

Description NVARCHAR(MAX) NOT NULL, -- Bắt buộc

Status INT NOT NULL -- Bắt buộc, không cho phép NULL

)

public class BrandModel

{

public string Name { get; set; } ;

public string Slug { get; set; };

public string Description { get; set; };

public int? Status { get; set; } // ✅ Thêm dấu ? để cho phép null

}

CREATE TABLE BrandModels (

Id INT PRIMARY KEY IDENTITY(1,1), -- EF tự thêm Id làm PK

Name NVARCHAR(MAX) NOT NULL, -- NOT NULL (vì string không có ?)

Slug NVARCHAR(MAX) NOT NULL, -- NOT NULL

Description NVARCHAR(MAX) NOT NULL, -- NOT NULL

Status INT NULL -- NULL (vì có int?) Status có thể null

);

**❌ NOT NULL = BẮT BUỘC PHẢI CÓ GIÁ TRỊ**

**✅ NULL = CÓ THỂ ĐỂ TRỐNG/KHÔNG CÓ GIÁ TRỊ**

Bài tập minh họa join table trong asp.net mvc

A screenshot of a computer

AI-generated content may be incorrect.

Models

🧩 1. Bảng Category

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

// Quan hệ 1 - N: 1 Category có nhiều Product

public List<Product> Products { get; set; }

}

🧩 2. Bảng Product

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

// Khóa ngoại

public int CategoryId { get; set; }

public Category Category { get; set; }

// Quan hệ 1 - N với OrderDetail

public List<OrderDetail> OrderDetails { get; set; }

}

🧩 3. Bảng OrderDetail

public class OrderDetail

{

public int Id { get; set; }

public int Quantity { get; set; }

// Khóa ngoại

public int ProductId { get; set; }

public Product Product { get; set; }

}

Controllers

Th1 : Join từ Categories 🡪 Products 🡪 OrderDetails (1-n-n)

var result = \_context.Categories

.Include(c => c.Products) // JOIN Category → Product

.ThenInclude(p => p.OrderDetails) // JOIN Product → OrderDetail

Th2 Join từ OrderDetails 🡪 Products 🡪 Categories

var item = \_context.OrderDetails.Include(c => c.Product).ThenInclude(x => x.Category).ToList();

Th3 join từ Products 🡪 Categories && Product 🡪 OrderDetails

var result = \_context.Products

.Include(p => p.Category) // JOIN Product → Category

.Include(p => p.OrderDetails) // JOIN Product → OrderDetails

.ToList();

Views

Th1 : Join từ Categories --> Products --- > OrderDetails (1-n-n)

<table class="table table-bordered">

<thead>

<tr>

<th>Category</th>

<th>Product</th>

<th>OrderDetail Id</th>

<th>Quantity</th>

</tr>

</thead>

<tbody>

@foreach (var category in Model)

{

foreach (var product in category.Products)

{

foreach (var order in product.OrderDetails)

{

<tr>

<td>@category.Name</td>

<td>@product.Name</td>

<td>@order.Id</td>

<td>@order.Quantity</td>

</tr>

}

}

}

</tbody>

</table>

Th2 Join từ OrderDetails -> Products -> Categories

@model List<Check.Models.OrderDetail>

<table class="table table-bordered">

<thead>

<tr>

<th>OrderDetail Id</th>

<th>Quantity</th>

<th>Product</th>

<th>Category</th>

</tr>

</thead>

<tbody>

@foreach (var order in Model)

{

<tr>

<td>@order.Id</td>

<td>@order.Quantity</td>

<td>@order.Product?.Name</td>

<td>@order.Product?.Category?.Name</td>

</tr>

}

</tbody>

</table>

Th3 join từ Products 🡪 Categories && Product 🡪 OrderDetails

@model List<Product>

<table class="table table-bordered">

<thead>

<tr>

<th>Product</th>

<th>Category</th>

<th>OrderDetail Id</th>

<th>Quantity</th>

</tr>

</thead>

<tbody>

@foreach (var product in Model)

{

foreach (var order in product.OrderDetails)

{

<tr>

<td>@product.Name</td>

<td>@product.Category?.Name</td>

<td>@order.Id</td>

<td>@order.Quantity</td>

</tr>

}

}

</tbody>

</table>

Result

Th1 : Join từ Categories --> Products --- > OrderDetails (1-n-n)

A screenshot of a computer

AI-generated content may be incorrect.

Th2 Join từ OrderDetails -> Products -> Categories

A screenshot of a computer

AI-generated content may be incorrect.

Th3 join từ Products 🡪 Categories && Product 🡪 OrderDetails

A screenshot of a computer

AI-generated content may be incorrect.