1- How We Can Add new Index?

ازاي نضيف Index جديد في Entity Framework Core؟

عندنا كذا طريقة:

1️⃣ من خلال الـ Data Annotations (في الكلاس):

لو عايزين نضيف Index على عمود معين، ممكن نستخدم الـ Attribute الجاهزة:

```
using Microsoft.EntityFrameworkCore;

[Index(nameof(Name), IsUnique = true)]   // هنا هيتعمل Index على العمود Name
public class Product
{
    public int ProductId { get; set; }
    public string Name { get; set; }
}
```

- **`IsUnique = true` معناها إن الـ Index هيمنع تكرار القيم (Unique Index).**

- **لو مش محتاجين يكون Unique، نسيبها فاضية.**

2️⃣ من خلال الـ Fluent API (في OnModelCreating داخل DbContext)

الطريقة دي بتديلنا مرونة أكبر:

```
protected override void
OnModelCreating(ModelBuilder modelBuilder)
```

```
{
    modelBuilder.Entity<Product>()
        .HasIndex(p => p.Name)
        .IsUnique();
}
```

---

3 بعد ما نضيف التعديلات

نعمل Migration جديدة:

```
Add-Migration AddIndexToProductName
```

بعد كده ننفذها:

```
Update-Database
```

هيتعمل Index جديد في قاعدة البيانات على العمود اللي اخترناه.
لو عملنا `IsUnique` → هيمنع تكرار القيم.
لو سيبناها من غير `IsUnique` هيبقى Index عادي لتسريع البحث.

2-How We Can Add new Sequence?

يعني إيه Sequence؟

- الـ Sequence هو مولّد أرقام مستقل عن أي جدول.
- بيطلع قيم متسلسلة (1, 2, 3 ...) أو حتى ممكن تعمليه يبدأ من رقم مختلف وبخطوة مختلفة (مثلاً 100، 200، 300 ...).
- زي الـ Identity، لكن الفرق إنه مش مربوط بجدول واحد، ممكن كذا جدول يستخدمه.

إزاي نضيف Sequence في EF Core؟

1 باستخدام الـ Fluent API

داخل OnModelCreating:

```
protected override void
OnModelCreating(ModelBuilder modelBuilder)
{
    // إنشاء Sequence جديدة
    modelBuilder.HasSequence<int>("OrderNumbers",
schema: "shared")
        .StartsAt(1000)   // يبدأ من 1000
        .IncrementsBy(5); // يزيد كل مرة بـ 5
}
```

- بتبدأ من 1000، وكل مرة تزيد 5.

---

2 ربط العمود بالـ Sequence

لو عايزين عمود في جدول يستخدم الـ Sequence ده:

```
modelBuilder.Entity<Order>()
    .Property(o => o.OrderNumber)
    .HasDefaultValueSql("NEXT VALUE FOR shared.OrderNumbers");
```

كده العمود `OrderNumber` في جدول `Orders` هياخد القيمة من الـ Sequence تلقائيًا.

3️⃣ بعد كده

نعمل Migration جديدة:

```
Add-Migration AddOrderNumberSequence
```

وننفذها:

```
Update-Database
```

النتيجة

- **هيتعمل Sequence في قاعدة البيانات.**
- **كل مرة نضيف صف جديد، الـ Column اللي مربوط بيه هياخد قيمة من الـ Sequence بشكل أوتوماتيكي.**

3- How do we deal with inheritance processes?

طرق التعامل مع الوراثة في EF Core

1️⃣ TPH – Table per Hierarchy (جدول واحد لكل الوراثة)

- كل الكلاسات الموروثة بتتخزن في جدول واحد.
- EF Core بيضيف عمود اسمه "Discriminator" يحدد نوع الكيان.

مثال:

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


public class Teacher : Person
{
    public decimal Salary { get; set; }
}
```

- EF Core هيعمل جدول واحد اسمه People فيه:

```
Id | Name | School | Salary | Discriminator
```

- `Discriminator` يوضح إذا كانت Student أو Teacher.

ده الديفولت في EF Core.

2️⃣ TPT – Table per Type (جدول لكل نوع)

- كل Class ليه جدول خاص بيه.
- الجدول بتاع الـ Child بيعمل علاقة بالـ Parent.

🔹 في OnModelCreating:

```
protected override void
OnModelCreating(ModelBuilder modelBuilder)

{


modelBuilder.Entity<Student>().ToTable("Student
s");


modelBuilder.Entity<Teacher>().ToTable("Teacher
s");

}
```

🔹 النتيجة:

- جدول People فيه الأعمدة المشتركة (Id, Name).
- جدول Students فيه الأعمدة الخاصة بيهم (School) + FK للـ Id.
- جدول Teachers فيه الأعمدة الخاصة بيهم (Salary) + FK للـ **Id.**

3️⃣ TPC – Table per Concrete Class (جدول لكل كلاس موروث)

- كل كلاس Child بياخد جدول كامل لوحده (من غير جدول مشترك).
- الأعمدة المشتركة بتتكرر في كل جدول.

🔹 **من EF Core 7 بقى مدعوم رسميًا:**

```
protected override void
OnModelCreating(ModelBuilder modelBuilder)

{


modelBuilder.Entity<Student>().UseTpcMappingStr
ategy();


modelBuilder.Entity<Teacher>().UseTpcMappingStr
ategy();

}
```

🔹 **النتيجة:**

- جدول Students: Id, Name, School
- جدول Teachers: Id, Name, Salary

الخلاصة

- TPH → جدول واحد + Discriminator (أبسط وأكتر استخدامًا).
- TPT → فصل البيانات لكل نوع (أوضح لكن ممكن يبطّأ الاستعلامات).
- TPC → كل جدول مستقل (أسرع في القراءة لكن فيه تكرار بيانات).