# Search Task

## 1. How We Can Add new Index?

**ال** Index هو حاجة بنضيفها على عمود أو أكتر في الجدول علشان تسرّع عمليات البحث والاستعلام.. يعني لو عندك جدول فيه آلاف الصفوف، وال Index معمول على عمود معين ف الاستعلام اللي بيستخدم العمود ده هيبقى أسرع بكتير عن اي عمود تاني مش بيستخدم ال Index.

**وممكن اعملها بطريقتين :**

### Data Annotation:

```
[Index(nameof(Email))]
 public class User
 {
     public int Id { get; set; }
     public string Email { get; set; }

 }
```

```
[Index(nameof(Url))]
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
}
```

### Fluent API:

```
modelBuilder.Entity <Customer>().HasIndex(u => u.Email);
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>()
        .HasIndex(b => b.Url);
}
```

ومن هنا نبدأ نتعرف على انواع ال Index اللي عندنا ونعرف كمان بتتكتب ازاي

**ال (Composite Index):** ده لو عايز أعمل Index على أكتر من عمود تعالى نشوف ازاي بنكتبه بطريقتين

### Data Annotation:

```
[Index(nameof(FirstName), nameof(LastName))]

    public class Person
{
    public int PersonId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
        }
```

**Fluent API:**

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
 {
     modelBuilder.Entity<Person>()
         .HasIndex(p => new { p.FirstName, p.LastName });
         }
```

2_ ال **(Index uniqueness)** :معناه الفهرس المتميز يعني فريد من نوعه بمعنى انه ممنوع يتكرر.. تعالى نشوف بيتكتب ازاي

**Data Annotation:**

```
    [Index(nameof(Url), IsUnique = true)]
    public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
        }
```

**Fluent API:**

```
  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
      modelBuilder.Entity<Blog>()
          .HasIndex(b => b.Url)
          .IsUnique();
         }
```

وكمان ممكن نضيف بعض التفاصيل زي اني ممكن احدد ترتيب الاعمده اللي كتبت عندها ال Index سواء تصاعدي او تنازلي وكده كده الافتراضي عنده هو (Ascending) ...لكن في معظم الحالات لو الفهرس على عمود واحد بس ف الترتيب مش بيفرق قوي..لكن لو الفهرسComposite (يعني على أكتر من عمود) ف الترتيب بيبقى مهم جدًا علشان الأداء وعلشان الاستعلامات تستخدم ال Index فعلاً.

## How We Can Add new Sequence?

ببساطة كده، ال **(Sequence) يعني التسلسل**.. بمعنى ان هو زي ماكينة بتطلعلك أرقام ورا بعضها زي 1، 2، 3، 4 ...بس الميزة إنه مش مربوط بجدول معين، يعني مش زي الـ Identity اللي بيشتغل جوه جدول واحد بس.

يعني ممكن تعمل Sequence اسمه مثلًا `InvoiceSequence` وتخلي جدول الفواتير ياخد منه رقم الفاتورة وفي نفس الوقت جدول الطلبات ياخد منه رقم الطلب وكل واحد بياخد رقم جديد من نفس المصدر.

بالشكل ده هنا بنقول يبدأ من 1000 ويزيد كل مره ب 5

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
      {
          modelBuilder.HasSequence<int>("OrderNumbers", schema: "shared")
              .StartsAt(1000)
              .IncrementsBy(5);
```

نيجي مثلا نفهم ونجرب المثال ده

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
      {
          modelBuilder.HasSequence<int>("OrderNumbers");

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

يعني كل مرة تضيف Orderجديد في رقم الطلب بيتعمل تلقائي من التسلسل اللي عملته .

بس في حاجه لازم ناخد بالنا منها .. الكود ده شغال كويس على SQL Server لأنه بيدعم الدالة اللي اسمها NEXT VALUE FOR لكن لو بتستخدم قاعدة بيانات تانية زي PostgreSQL أوSQLite الكود ده ممكن ما يشتغلش ولازم ترجع لل documentation قاعدة البيانات اللي بتستخدمها وتشوف الطريقة المناسبة فيها.

## 2. How We deal with inheritance processes?

لما تكتب كود في C# فيه وراثة يعني مثلًا عندك كلاس رئيسي اسمه `Parent` وكلاس تاني اسمه `Chield` بيورث منه ال EF Core يقدر يفهم العلاقة دي ويخزنها في قاعدة البيانات بشكل منظم ويعمل queries عليها كأنها Class واحد.

يعني تقدر تشتغل بكلاسات فيها وراثة عادي جدًا و EF Coreهيترجمها لهيكل مناسب في قاعدة البيانات من غير ما تتعب نفسك.

ولكن في نقطه مهمه ال EF Core مش بيكتشف الوراثة لوحده بمعنى لو عندك كلاس رئيسي فيه وراثة لازم تقول لـ EF Core إنك عايز تضيف الكلاسات الموروثة دي في ال Modelبتاعك لو عملت `DbSet<Parent>` بس ال EFمش هيضيف `Chield` تلقائي إلا لو كتبتها .

تعالى نشوف ازاي

```
internal class MyContext : DbContext
        {
            public DbSet<Parent> Parents { get; set; }
            public DbSet<Chield> Chields { get; set; }
        }

        public class Parent
        {
            public int ParentId { get; set; }
            public string Url { get; set; }
        }

        public class Chield : Parent
        {
            public string Url { get; set; }
```

EF Core بيدعم 3 طرق لتخزينهم في قاعدة البيانات:

1- TPH (Table Per Hierarchy)

2- TPT (Table Per Type)

3- TPC (Table Per Concrete Type)

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

        public class Dog : Animal
        {
            public string Breed { get; set; }
        }

        modelBuilder.Entity<Animal>()
    .HasDiscriminator<string>("AnimalType")
    .HasValue<Animal>("Base")

    .HasValue<Dog>("Dog");
```

**1-** TPHجدول واحد فيه كل الأنواع : دي الطريقة الBy Default، كل الكلاسات بتتخزن في جدول واحد، وفيه عمود بيميز النوع .

2- TPT كل نوع في جدول منفصل : يعني كل كلاس ليه جدول خاص بيه، وبيكون فيه علاقة بينهم.

```
modelBuilder.Entity<Animal>().ToTable("Animals");
modelBuilder.Entity<Dog>().ToTable("Dogs");
```

3- TPC يعني كل كلاس موروث بيتخزن في جدول مستقل، ومفيش جدول للكلاس الأساسي.

```
modelBuilder.Entity<Animal>().UseTpcMappingStrategy()
;
```