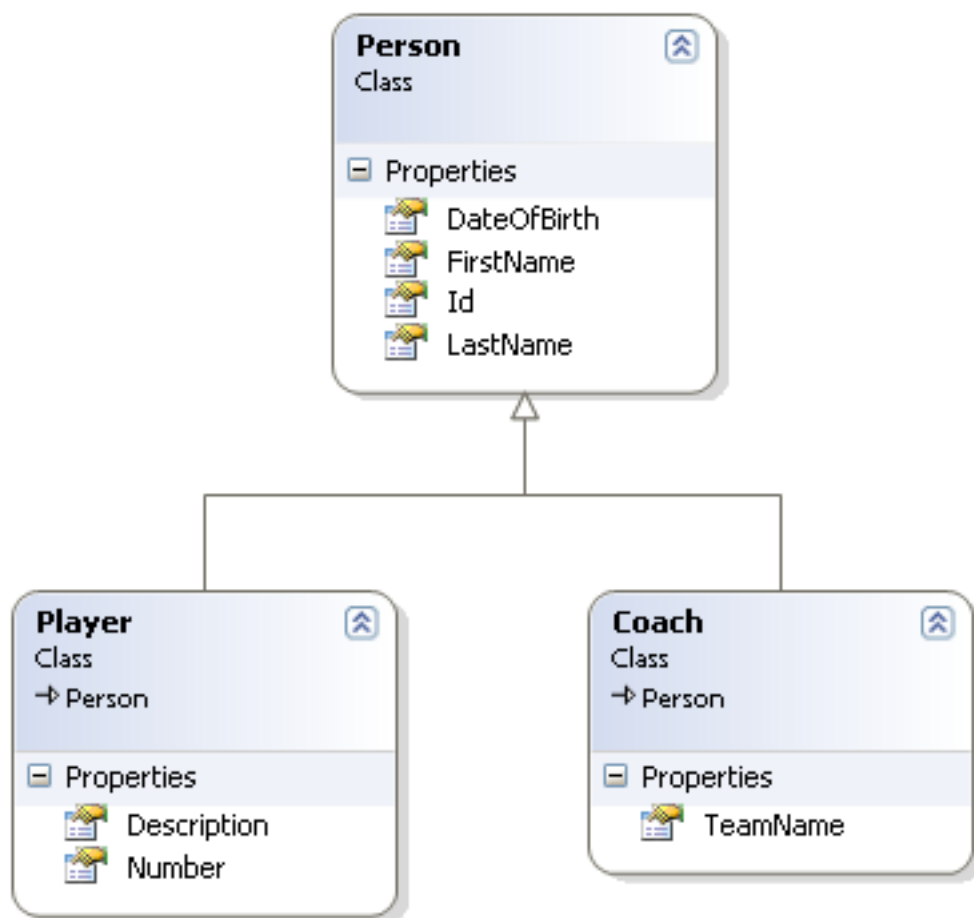


تنظیمات ارث بری کلاس‌ها در EF Code first

بانک‌های اطلاعاتی مبتنی بر SQL، تنها روابطی از نوع «has a» یا «دارای» را پشتیبانی می‌کنند؛ اما در دنیای شیء‌گرا روابطی مانند «is a» یا «هست» نیز قابل تعریف هستند. برای توضیحات بیشتر به مدل‌های زیر دقت نمایید:



```

using System;

namespace EF_Sample05.DomainClasses.Models
{
    public abstract class Person
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public DateTime DateOfBirth { get; set; }
    }
}
  
```

```
namespace EF_Sample05.DomainClasses.Models
{
    public class Coach : Person
    {
        public string TeamName { set; get; }
    }
}
```

```
namespace EF_Sample05.DomainClasses.Models
{
    public class Player : Person
    {
        public int Number { get; set; }
        public string Description { get; set; }
    }
}
```

در این مدل‌ها که بر اساس ارث بری از کلاس شخص، تهیه شده‌اند؛ بازیکن، یک شخص است. مربی نیز یک شخص است؛ و به این ترتیب خوانده می‌شوند:

```
Coach "is a" Person
Player "is a" Person
```

در EF Code first سه روش جهت کار با این نوع کلاس‌ها و کلا ارث بری وجود دارد که در ادامه به آن‌ها خواهیم پرداخت:

الف) TPH یا Table per Hierarchy



همانطور که از نام آن نیز پیدا است، کل سلسله مراتبی را که توسط ارث بری تعریف شده است، تبدیل به یک جدول در بانک اطلاعاتی می‌کند. این حالت، شیوه برخورد پیش فرض EF Code first با ارث بری کلاس‌ها است و نیاز به هیچگونه تنظیم خاصی ندارد.

برای آزمایش این مساله، کلاس Context را به نحو زیر تعریف نمائید و سپس اجازه دهید تا EF بانک اطلاعاتی معادل آن را تولید کند:

```
using System.Data.Entity;
using EF_Sample05.DomainClasses.Models;

namespace EF_Sample05.DataLayer.Context
{
    public class Sample05Context : DbContext
    {
        public DbSet<Person> People { set; get; }
    }
}
```

ساختار جدول تولید شده آن همانند تصویر زیر است:

People			
	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	FirstName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	LastName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	DateOfBirth	datetime	<input type="checkbox"/>
	Number	int	<input checked="" type="checkbox"/>
	Description	nvarchar(MAX)	<input checked="" type="checkbox"/>
	TeamName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Discriminator	nvarchar(128)	<input type="checkbox"/>
			<input type="checkbox"/>

همانطور که ملاحظه می‌کنید، تمام کلاس‌های مشتق شده از کلاس شخص را تبدیل به یک جدول کرده است؛ به علاوه یک فیلد جدید را هم به نام Discriminator به این جدول اضافه نموده است. برای درک بهتر عملکرد این فیلد، چند رکورد را توسط برنامه به بانک اطلاعاتی اضافه می‌کنیم. حاصل آن به شکل زیر خواهد بود:

Results

Messages

	Id	FirstName	LastName	DateOfBirth	Number	Description	TeamName	Discriminator
1	1	Coach F1	Coach L1	1962-05-11 10:59:25.853	NULL	NULL	Team A	Coach
2	2	Coach F2	Coach L2	1962-05-11 10:59:29.883	NULL	NULL	Team B	Coach
3	3	Coach F1	Coach L1	1962-05-11 10:59:29.883	1	...	NULL	Player

از فیلد Discriminator جهت ثبت نام کلاس‌های متناظر با هر رکورد، استفاده شده است. به این ترتیب EF حین کار با اشیاء دقیقاً می‌داند که چگونه باید خواص متناظر با کلاس‌های مختلف را مقدار دهی کند. به علاوه اگر به ساختار جدول تهیه شده دقت کنید، مشخص است که در حالت TPH، نیاز است فیلدهای متناظر با کلاس‌های مشتق شده از کلاس پایه، همگی null پذیر باشند. برای نمونه فیلد Number که از نوع int تعریف شده، در سمت بانک اطلاعاتی نال پذیر تعریف شده است. و برای کوئری نوشتن در این حالت می‌توان از متد الحاقی ofType جهت فیلتر کردن اطلاعات بر اساس کلاسی خاص، کمک گرفت:

```
db.People.OfType<Coach>().FirstOrDefault(x => x.LastName == "Coach L1")
```

سفارشی سازی نحوه نگاشت TPH

همانطور که عنوان شد، TPH نیاز به تنظیمات خاصی ندارد و حالت پیش فرض است؛ اما برای مثال می‌توان بر روی مقادیر و نوع ستون Discriminator تولیدی، کنترل داشت. برای این منظور باید از Fluent API به نحو زیر استفاده کرد:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample05.DomainClasses.Models;

namespace EF_Sample05.DataLayer.Mappings
{
    public class CoachConfig : EntityTypeConfiguration<Coach>
    {
        public CoachConfig()
        {
            // For TPH
            this.Map(m => m.Requires(discriminator: "PersonType").HasValue(1));
        }
    }
}
```

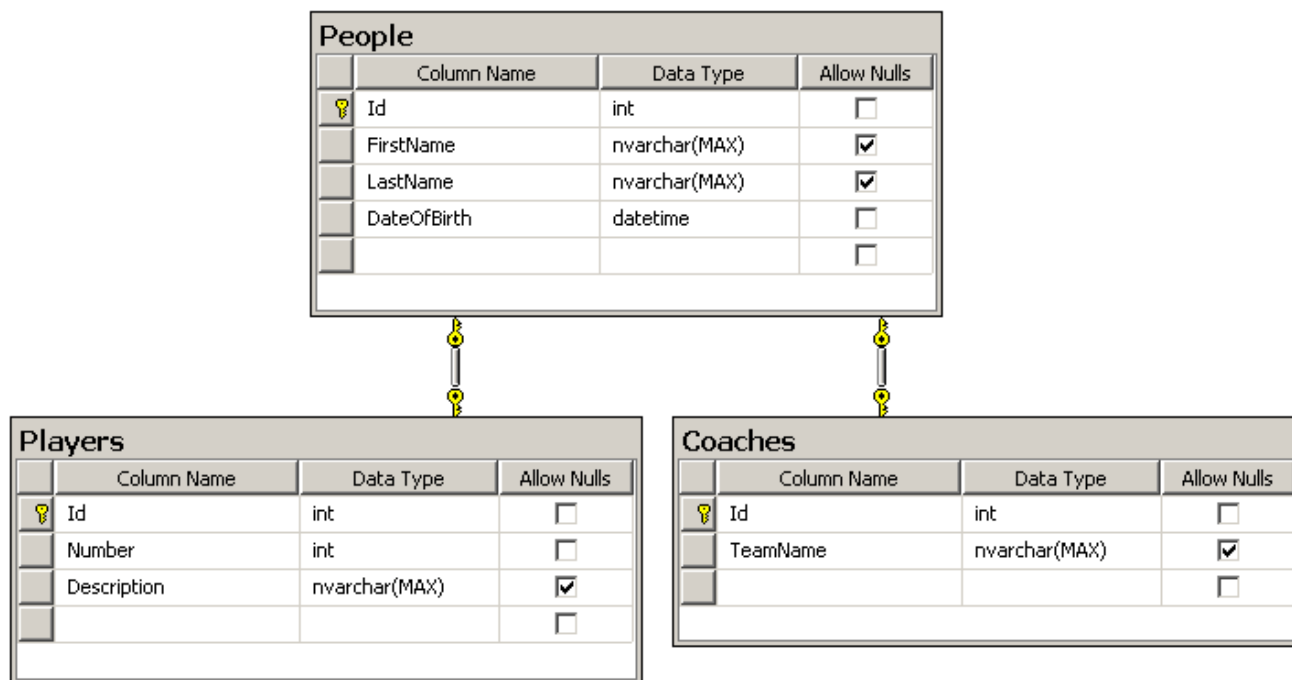
```
using System.Data.Entity.ModelConfiguration;
using EF_Sample05.DomainClasses.Models;

namespace EF_Sample05.DataLayer.Mappings
{
    public class PlayerConfig : EntityTypeConfiguration<Player>
    {
        public PlayerConfig()
        {
            // For TPH
            this.Map(m => m.Requires(discriminator: "PersonType").HasValue(2));
        }
    }
}
```

در اینجا توسط متد Map، نام فیلد discriminator به PersonType تغییر کرده. همچنین چون مقدار پیش فرض تعیین شده توسط متد HasValue عددی است، نوع این فیلد در سمت بانک اطلاعاتی به int null تغییر می‌کند.

TPT یا Table per Type (ب)

در حالت TPT، به ازای هر کلاس موجود در سلسله مراتب تعیین شده، یک جدول در سمت بانک اطلاعاتی تشکیل می‌گردد. در جداول متناظر با Sub classes، تنها همان فیلدهایی وجود خواهند داشت که در کلاس‌های هم نام وجود دارد و فیلدهای کلاس پایه در آن‌ها ذکر نخواهد گردید. همچنین این جداول دارای یک Primary key نیز خواهند بود (که دقیقاً همان کلید اصلی جدول پایه است که به آن Shared primary key هم گفته می‌شود). این کلید اصلی، به عنوان کلید خارجی اشاره کننده به کلاس یا جدول پایه نیز تنظیم می‌گردد:



برای تنظیم این نوع ارث بری، تنها کافی است ویژگی Table را بر روی Sub classes قرار داد:

```
using System.ComponentModel.DataAnnotations;

namespace EF_Sample05.DomainClasses.Models
{
    [Table("Coaches")]
    public class Coach : Person
    {
        public string TeamName { get; set; }
    }
}
```

```
using System.ComponentModel.DataAnnotations;

namespace EF_Sample05.DomainClasses.Models
{
    [Table("Players")]
    public class Player : Person
    {
        public int Number { get; set; }
        public string Description { get; set; }
    }
}
```

یا اگر حالت Fluent API را ترجیح می‌دهید، همانطور که در قسمت‌های قبل نیز ذکر شد، معادل ویژگی Table در اینجا، متد ToTable است.

ج) Table per Concrete type یا TPC

در تعاریف ارث بری که تاکنون بررسی کردیم، مرسوم است کلاس پایه را از نوع abstract تعریف کنند. به این ترتیب هدف اصلی،

Sub classes تعریف شده خواهند بود؛ چون نمی‌توان مستقیماً وهله‌ای را از کلاس abstract تعریف شده ایجاد کرد. در حالت TPC، به ازای هر sub class غیر abstract، یک جدول ایجاد می‌شود. هر جدول نیز حاوی فیلدهای کلاس پایه می‌باشد (برخلاف حالت TPT که جداول متناظر با کلاس‌های مشتق شده، تنها حاوی همان خواص و فیلدهای کلاس‌های متناظر بودند و نه بیشتر). به این ترتیب عملاً جداول تشکیل شده در بانک اطلاعاتی، از وجود ارث بری در سمت کدهای ما بی‌خبر خواهند بود.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	FirstName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	LastName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	DateOfBirth	datetime	<input type="checkbox"/>
	TeamName	nvarchar(MAX)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	FirstName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	LastName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	DateOfBirth	datetime	<input type="checkbox"/>
	Number	int	<input type="checkbox"/>
	Description	nvarchar(MAX)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

برای پیاده سازی TPC نیاز است از Fluent API استفاده شود:

```
using System.ComponentModel.DataAnnotations;
using System.Data.Entity.ModelConfiguration;
using EF_Sample05.DomainClasses.Models;

namespace EF_Sample05.DataLayer.Mappings
{
    public class PersonConfig : EntityTypeConfiguration<Person>
    {
        public PersonConfig()
        {
            // for TPC
            this.Property(x => x.Id).HasDatabaseGeneratedOption(DatabaseGeneratedOption.None);
        }
    }
}
```

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample05.DomainClasses.Models;

namespace EF_Sample05.DataLayer.Mappings
{
    public class CoachConfig : EntityTypeConfiguration<Coach>
    {
        public CoachConfig()
        {
            // For TPH
            //this.Map(m => m.Requires(discriminator: "PersonType").HasValue(1));

            // for TPT
            //this.ToTable("Coaches");

            //for TPC
            this.Map(m =>
            {
                m.MapInheritedProperties();
                m.ToTable("Coaches");
            });
        }
    }
}
```

```

using System.Data.Entity.ModelConfiguration;
using EF_Sample05.DomainClasses.Models;

namespace EF_Sample05.DataLayer.Mappings
{
    public class PlayerConfig : EntityTypeConfiguration<Player>
    {
        public PlayerConfig()
        {
            // For TPH
            //this.Map(m => m.Requires(discriminator: "PersonType").HasValue(2));

            // for TPT
            //this.ToTable("Players");

            //for TPC
            this.Map(m =>
            {
                m.MapInheritedProperties();
                m.ToTable("Players");
            });
        }
    }
}

```

ابتدا نوع فیلد Id از حالت Identity خارج شده است. این مورد جهت کار با TPC ضروری است در غیراینصورت EF هنگام ثبت، به مشکل بر می خورد، از این لحاظ که برای دو شیء، به یک Id خواهد رسید و امکان ثبت را نخواهد داد. بنابراین در یک چنین حالتی استفاده از نوع Guid برای تعریف primary key شاید بهتر باشد. بدیهی است در این حالت باید Id را به صورت دستی مقدار دهی نمود.

در ادامه توسط متد MapInheritedProperties، به همان مقصود لحاظ کردن تمام فیلدهای ارث بری شده در جدول حاصل، خواهیم رسید. همچنین نام جداول متناظر نیز ذکر گردیده است.

سؤال : از این بین، بهتر است از کدامیک استفاده شود؟

- برای حالت‌های ساده از TPH استفاده کنید. برای مثال یک بانک اطلاعاتی قدیمی دارید که هر جدول آن 200 تا یا شاید بیشتر فیلد دارد! امکان تغییر طراحی آن هم وجود ندارد. برای اینکه بتوان به حس بهتری حین کارکردن با این نوع سیستم‌های قدیمی رسید، می‌شود از ترکیب TPH و ComplexTypes (که در قسمت‌های قبل در مورد آن بحث شد) برای مدیریت بهتر این نوع جداول در سمت کدهای برنامه استفاده کرد.
- اگر علاقمند به استفاده از روابط پلی‌مرفیک هستید (برای مثال در کلاسی دیگر، ارجاعی به کلاس پایه Person وجود دارد) و sub classes دارای تعداد فیلدهای کمی هستند، از TPH استفاده کنید.
- اگر تعداد فیلدهای sub classes زیاد است و بسیار بیشتر است از کلاس پایه، از روش TPT استفاده کنید.
- اگر عمق ارث بری و تعداد سطوح تعریف شده بالا است، بهتر است از TPC استفاده کنید. حالت TPT از join استفاده می‌کند و حالت TPC از union برای تشکیل کوئری‌ها کمک خواهد گرفت

نظرات خوانندگان

نویسنده: ایلیا اکبری فرد
تاریخ: ۲۰:۱۴ ۱۳۹۱/۰۷/۳۰

آقای نصیری با سلام.
من 4 تا Entity دارم و یک Entity پایه. استراژی پیاده سازی اون هم TPT است. جدول پایه حدود 100 فیلد دارد. موجودیت‌های فرزند هم هرکدام حدود 30 ستون دارند. وقتی با EF Code First داده‌ها را واکنشی میکنم کوئری بسیار سنگین را در SQL Server Profiler ایجاد می‌شود که وقتی آنرا در NotePad کپی میکنم چیزی حدود 60 کیلوبایت می‌شود. ضمناً به تمام ستون‌های جداول نیاز دارم تا آنها را در گرید نمایش دهم.
در صورت امکان راهنمایی کنید. [heavyQuery.txt](#)

نویسنده: وحید نصیری
تاریخ: ۲۱:۴۱ ۱۳۹۱/۰۷/۳۰

سنگینی یا سبکی یک کوئری بر اساس execution plan آن محاسبه می‌شود و نه حجم کوئری در notepad. بررسی این مورد هم نیاز به جزئیات دقیق کار شما دارد مانند ساختار کلاس‌ها و کوئری LINQ نوشته شده. (ضمن اینکه جویین 5 جدول با هم، با هر ابزاری کند است. این مورد ربطی به EF ندارد. باید طراحی کار خودتون رو بررسی و تصمیم گیری یا اصلاح کنید)

نویسنده: حسین
تاریخ: ۱۵:۵۵ ۱۳۹۲/۰۲/۰۷

سلام. میشه در حالت TPH برای فیلد discriminator یک پروپرتی تعریف کنیم؟ یعنی اگر بخواهیم به مقدار فیلد discriminator از طریق کد دسترسی داشته باشیم چکار باید کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۶:۳۲ ۱۳۹۲/۰۲/۰۷

- این فیلد ویژه به صورت خودکار توسط ساز و کار داخلی EF مدیریت می‌شود و امکان نوشتن یا خواندن مستقیم آن وجود ندارد. (مگر اینکه مستقیماً SQL بنویسید و توسط SqlQuery آنرا اجرا کنید)
- اما ... برای دسترسی به آن جهت یافتن نوعی خاص، فقط کافی هست بنویسید:

```
db.People.OfType<Coach>() // .Where ...
```

OfType مطابق نوع آرگومان جنریکی که دریافت می‌کنه، اطلاعات را بر اساس فیلد discriminator متناظر فیلتر خواهد کرد. مثلاً در اینجا افرادی از نوع مربی فیلتر شدن.

نویسنده: مهدی فرهانی
تاریخ: ۲۲:۲۷ ۱۳۹۲/۰۲/۱۰

سلام
زمانی که از TPT استفاده میکنم و نیاز دارم که یکسری اطلاعات را از جدول پایه فراخوانی کنم بدون اینکه به جداول دیگه نیاز داشته باشم کوئری عجیب غریبی میسازه.
آیا روشی برای اصلاح این نوع کوئری‌ها هست؟ شاید هم من اشتباه استفاده کردم!

این یک تیکه از کوئری ساخته شده است که در آخر هم همه جداول رو با هم جویین میکند.

```
CASE WHEN ((CASE WHEN ([Extent1].[Discriminator] = N'Person') THEN cast(1 as bit) ELSE cast(0 as bit)
END) = 1) THEN CAST(NULL AS varchar(1)) WHEN ((([Project6].[C1] = 1) AND ([Project6].[C1] IS NOT NULL)
AND ( NOT ((([Project6].[C2] = 1) AND ([Project6].[C2] IS NOT NULL)))) AND ( NOT ((([Project6].[C3] = 1)
AND ([Project6].[C3] IS NOT NULL)))) AND ( NOT ((([Project6].[C4] = 1) AND ([Project6].[C4] IS NOT
```



```

NULL)))) THEN [Project6].[Name] WHEN (([Project6].[C2] = 1) AND ([Project6].[C2] IS NOT NULL)) THEN
[Project6].[Name] WHEN (([Project1].[C1] = 1) AND ([Project1].[C1] IS NOT NULL)) THEN CAST(NULL AS
varchar(1)) WHEN (([Project6].[C4] = 1) AND ([Project6].[C4] IS NOT NULL)) THEN [Project6].[Name] WHEN
(([Project2].[C1] = 1) AND ([Project2].[C1] IS NOT NULL)) THEN CAST(NULL AS varchar(1)) WHEN
(([Project6].[C3] = 1) AND ([Project6].[C3] IS NOT NULL)) THEN [Project6].[Name] END AS [C2],
CASE WHEN ((CASE WHEN ([Extent1].[Discriminator] = N'Person') THEN cast(1 as bit) ELSE cast(0 as bit)
END) = 1) THEN CAST(NULL AS varchar(1)) WHEN (([Project6].[C1] = 1) AND ([Project6].[C1] IS NOT NULL)
AND ( NOT (([Project6].[C2] = 1) AND ([Project6].[C2] IS NOT NULL))) AND ( NOT (([Project6].[C3] = 1)
AND ([Project6].[C3] IS NOT NULL))) AND ( NOT (([Project6].[C4] = 1) AND ([Project6].[C4] IS NOT
NULL)))) THEN [Project6].[Family] WHEN (([Project6].[C2] = 1) AND ([Project6].[C2] IS NOT NULL)) THEN
[Project6].[Family] WHEN (([Project1].[C1] = 1) AND ([Project1].[C1] IS NOT NULL)) THEN CAST(NULL AS
varchar(1)) WHEN (([Project6].[C4] = 1) AND ([Project6].[C4] IS NOT NULL)) THEN [Project6].[Family]
WHEN (([Project2].[C1] = 1) AND ([Project2].[C1] IS NOT NULL)) THEN CAST(NULL AS varchar(1)) WHEN
(([Project6].[C3] = 1) AND ([Project6].[C3] IS NOT NULL)) THEN [Project6].[Family] END AS [C3],
CASE WHEN ((CASE WHEN ([Extent1].[Discriminator] = N'Person') THEN cast(1 as bit) ELSE cast(0 as bit)
END) = 1) THEN CAST(NULL AS datetime2) WHEN (([Project6].[C1] = 1) AND ([Project6].[C1] IS NOT NULL)
AND ( NOT (([Project6].[C2] = 1) AND ([Project6].[C2] IS NOT NULL))) AND ( NOT (([Project6].[C3] = 1)
AND ([Project6].[C3] IS NOT NULL))) AND ( NOT (([Project6].[C4] = 1) AND ([Project6].[C4] IS NOT
NULL)))) THEN [Project6].[DateOfBirth] WHEN (([Project6].[C2] = 1) AND ([Project6].[C2] IS NOT NULL))
THEN [Project6].[DateOfBirth] WHEN (([Project1].[C1] = 1) AND ([Project1].[C1] IS NOT NULL)) THEN
CAST(NULL AS datetime2) WHEN (([Project6].[C4] = 1) AND ([Project6].[C4] IS NOT NULL)) THEN
[Project6].[DateOfBirth] WHEN (([Project2].[C1] = 1) AND ([Project2].[C1] IS NOT NULL)) THEN CAST(NULL
AS datetime2) WHEN (([Project6].[C3] = 1) AND ([Project6].[C3] IS NOT NULL)) THEN
[Project6].[DateOfBirth] END AS [C4],

```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۰ ۲۳:۱

تنظیمات شما اشتباه است. وجود فیلد Discriminator در بانک اطلاعاتی به معنای استفاده از روش TPH است و نه TPT. در حالت TPH کل کلاس‌های مشتق شده از کلاس پایه، با آن یکی می‌شوند که نیاز به Discriminator برای تمایز قائل شدن بین آن‌ها وجود دارد (در یک جدول و نه در بیش از سه جدولی که در کوئری شما نامبرده شده). در کل نیاز به بررسی کدهای شما و روابط آن هست. شاید خاصیت ارتباطی اضافی وجود دارد، شاید روابط صحیح تنظیم نشدن.

نویسنده: مهدی فرهانی
تاریخ: ۱۳۹۲/۰۲/۱۰ ۲۳:۲۷

```

public class Person:BaseEntity
{
    public int PersonId { get; set; }
    [StringLength(100)]
    public string Title { get; set; }
    public PersonType PersonType { get; set; }
    public virtual ICollection<PrivacyPolicy> PrivacyPolicies { get; set; }

    public override string ToString()
    {
        return Title;
    }
}

```

کلاس دوم

```

[Table("Organs")]
public class Organ:Person
{
    [StringLength(100)]
    public string FullName { get; set; }
    [StringLength(1000)]
    public string Address { get; set; }

    public override string ToString()
    {
        return FullName;
    }
    public Organ()
    {
        PersonType = PersonType.Organ;
    }
}

```

تو این مثال از Discriminator استفاده کرده ولی بعضی وقتها کوئری‌ها به این شکل ایجاد میکنه

```
CASE WHEN (( NOT (([UnionAll12].[C7] = 1) AND ([UnionAll12].[C7] IS NOT NULL))) AND ( NOT
([UnionAll12].[C8] = 1) AND ([UnionAll12].[C8] IS NOT NULL))) AND ( NOT (([UnionAll12].[C9] = 1) AND
([UnionAll12].[C9] IS NOT NULL))) THEN CAST(NULL AS varchar(1)) WHEN (([UnionAll12].[C9] = 1) AND
([UnionAll12].[C9] IS NOT NULL)) THEN [UnionAll12].[C2] WHEN (([UnionAll12].[C8] = 1) AND
([UnionAll12].[C8] IS NOT NULL)) THEN CAST(NULL AS varchar(1)) END AS [C2],
CASE WHEN (( NOT (([UnionAll12].[C7] = 1) AND ([UnionAll12].[C7] IS NOT NULL))) AND ( NOT
([UnionAll12].[C8] = 1) AND ([UnionAll12].[C8] IS NOT NULL))) AND ( NOT (([UnionAll12].[C9] = 1) AND
([UnionAll12].[C9] IS NOT NULL))) THEN CAST(NULL AS varchar(1)) WHEN (([UnionAll12].[C9] = 1) AND
([UnionAll12].[C9] IS NOT NULL)) THEN CAST(NULL AS varchar(1)) WHEN (([UnionAll12].[C8] = 1) AND
([UnionAll12].[C8] IS NOT NULL)) THEN [UnionAll12].[C3] END AS [C3],
CASE WHEN (( NOT (([UnionAll12].[C7] = 1) AND ([UnionAll12].[C7] IS NOT NULL))) AND ( NOT
([UnionAll12].[C8] = 1) AND ([UnionAll12].[C8] IS NOT NULL))) AND ( NOT (([UnionAll12].[C9] = 1) AND
([UnionAll12].[C9] IS NOT NULL))) THEN CAST(NULL AS int) WHEN (([UnionAll12].[C9] = 1) AND
([UnionAll12].[C9] IS NOT NULL)) THEN CAST(NULL AS int) WHEN (([UnionAll12].[C8] = 1) AND
([UnionAll12].[C8] IS NOT NULL)) THEN [UnionAll12].[C4] END AS [C4],
```

زمان اجرای کوئری پایین هست ، ولی حجم کد تولید شده بالا هست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۱

بهتر هست این مسایل رو در انجمن‌ها پیگیری کنید. کوئری قبلی شما در مورد پروژه و DateOfBirth بود، کلاس‌هایی که ارائه دادید در مورد شخص و ارگان است با یک سری فیلد دیگر. صحبت از TPT بود بعد فیلد Discriminator داشتید. کار می‌کنه سیستم؟ همین خوبه. دستی هم بخواهید با بیش از سه جدول با هم کار کنید باید جوین بنویسید. موفق باشید

نویسنده: مهدی فرهانی
تاریخ: ۱۳۹۲/۰۲/۱۱

مهندس جان سوء تفاهم شده ، کوئری که گذاشتم قسمتی از کوئری بود ، من یک کلاس پایه دارم به نام Person و یکسری کلاس مثل Role, User, University, Organ و..... که از Person ارث بری میکنند . Discriminator هم فالت خودم بود که برای یکی از کلاس‌ها فراموش کرده بودم با Table مزینش کنم. بحث اصلی من سر کوئری حجیمی هست که تولید میشه.

تو این مسئله من نیازی به جوین ندارم و فقط می‌خواهم اطلاعات پایه خونده بشه نه بقیه کلاس‌ها ، که این مشکل را با پروجیکشن حل کردم . ولی می‌خواهم بدونم چرا همچین کوئری میسازه زمانی که از TPT استفاده میکنم . اونم با این همه Case When و Union.

```
SELECT
CASE WHEN (( NOT (([Project7].[C1] = 1) AND ([Project7].[C1] IS NOT NULL))) AND ( NOT (([Project3].[C1]
= 1) AND ([Project3].[C1] IS NOT NULL))) AND ( NOT (([Project2].[C1] = 1) AND ([Project2].[C1] IS NOT
NULL))) AND ( NOT (([Project1].[C1] = 1) AND ([Project1].[C1] IS NOT NULL))) THEN '0X' WHEN
([Project7].[C1] = 1) AND ([Project7].[C1] IS NOT NULL) AND ( NOT (([Project7].[C2] = 1) AND
([Project7].[C2] IS NOT NULL))) AND ( NOT (([Project7].[C3] = 1) AND ([Project7].[C3] IS NOT NULL)))
AND ( NOT (([Project7].[C4] = 1) AND ([Project7].[C4] IS NOT NULL))) THEN '0X0X' WHEN
([Project7].[C2] = 1) AND ([Project7].[C2] IS NOT NULL) THEN '0X0X0X' WHEN (([Project2].[C1] = 1) AND
([Project2].[C1] IS NOT NULL)) THEN '0X1X' WHEN (([Project7].[C4] = 1) AND ([Project7].[C4] IS NOT
NULL)) THEN '0X0X1X' WHEN (([Project3].[C1] = 1) AND ([Project3].[C1] IS NOT NULL)) THEN '0X2X' WHEN
([Project7].[C3] = 1) AND ([Project7].[C3] IS NOT NULL) THEN '0X0X2X' ELSE '0X3X' END AS [C1],
[Extent1].[PersonId] AS [PersonId],
[Extent1].[Title] AS [Title],
[Extent1].[PersonType] AS [PersonType],
```

آیا واقعاً این همه Case لازم داره ، یا باز من اشتباه کردم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۱

- کارآیی یک کوئری بر اساس execution plan آن بررسی می‌شود و نه حجم حاصل از فیلدهای درگیر در آن.
- هرگونه مشکلات و ناراحتی‌های خودتون رو در مورد طراحی EF [در اینجا](#) و یا [اینجا](#) ارسال کنید.

نویسنده: وحید م
تاریخ: ۱۷:۲۴ ۱۳۹۲/۰۹/۲۹

با سلام یک کلاس abstrac بنام person که یک سری خصوصیات دارد و یکسری کلاس از آن مشتق شده حال کلاس person چگونه می‌تواند به فیلدهای کلاس مشتق شده دسترسی داشته باشد مثل حالت tph که ما فقط به dbset مان person رادادیم و آن توانست جدولی با تمام فیلدهای کلاس مشتق شده برایمان درست نماید ممنون. این روند فقط در codefirst موجود است یا خاصیت oop است

نویسنده: وحید نصیری
تاریخ: ۱۸:۳۷ ۱۳۹۲/۰۹/۲۹

از متد الحاقی [OfType](#) برای دسترسی به خواص زیرکلاس‌ها استفاده کنید. [مثالش](#) در متن هست.

نویسنده: محمد رضا خزائی
تاریخ: ۱۴:۱۵ ۱۳۹۲/۱۲/۱۷

با سلام
در روش TPT اگر بخواهیم فقط اطلاعات جدول پایه (پدر) را select بزنیم، متأسفانه تمامی جداول مشتق شده با هم Union شده و بعد با جدول پایه Join می‌خورد.
آیا راهی وجود دارد که فقط از جدول پایه Select زده شود؟
مرسی

نویسنده: محمد رضا خزائی
تاریخ: ۱۶:۱۴ ۱۳۹۲/۱۲/۱۷

پیدا کردم
باید کلاس پایه رو از حالت Abstract خارج کنیم.