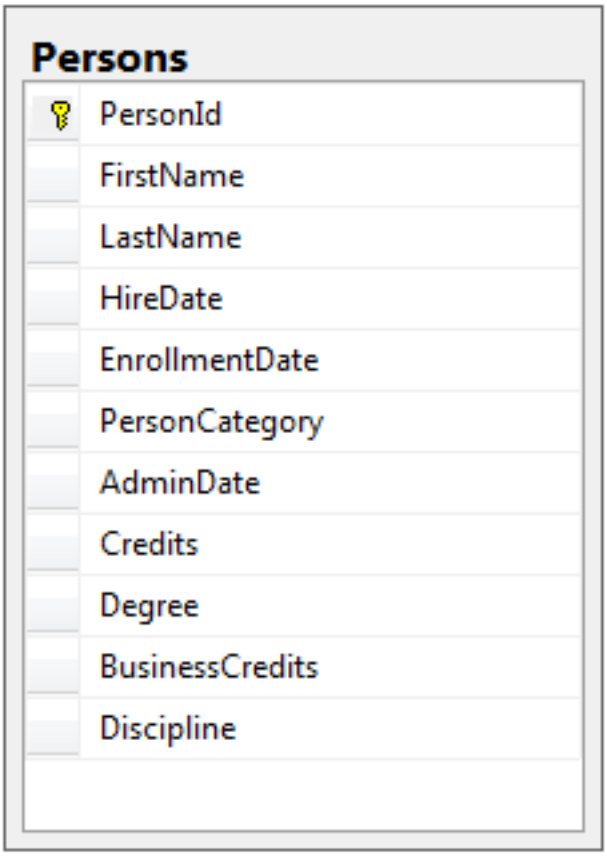
عنوان: **روش های ارث بری در Entity Framework - قسمت دوم**
نویسنده: بهروز راد
تاریخ: ۱۳۹۲/۰۲/۲۱ ۸:۴۰
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework, Inheritance Strategies, Table per hierarchy, TPH

در قسمت اول در مورد روش TPT خواندید. در این قسمت به روش TPH می‌پردازیم.

روش TPH

در این روش، ارث بری از طریق فقط یک جدول ایجاد می‌شود و زیر مجموعه‌ها بر اساس مقدار یک فیلد از یکدیگر متمایز می‌شوند. پس اگر جدولی دارید که برای متمایز کردن رکوردهای آن از یک فیلد استفاده می‌کنید، روش TPH مناسب شما است. با روش TPH نیز می‌توانید به همان مدلی که در روش TPT دارید برسید، تنها تفاوت این هست که در روش TPH، تمامی داده‌ها در یک جدول قرار دارند و یک فیلد برای متمایز کردن رکوردها استفاده می‌شود. همه چیز با مثال عملی واضح‌تر است. پس کار خود را با یک مثال ادامه می‌دهیم. جدول مثال ما در شکل زیر مشخص است.



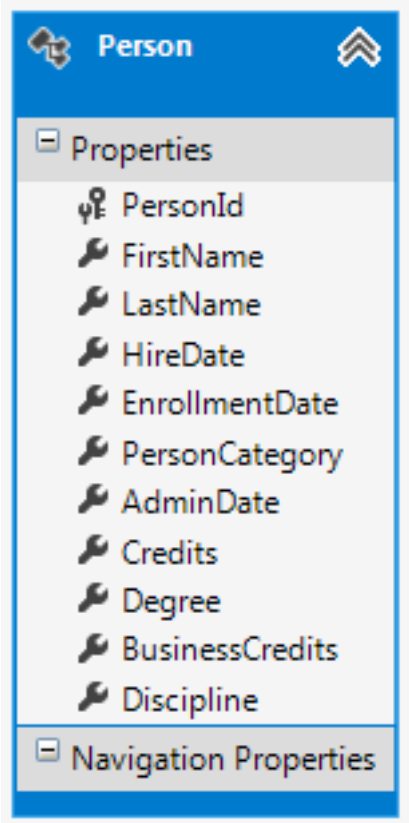
به نظر می‌رسد که این جدول با جداول قسمت قبل شباهتی دارد. بله! فیلدهای جداول مثال قبل در این جدول آمده اند.
فیلدهای FirstName و LastName از جدول Persons
فیلد HireDate از جدول Instructors
فیلد EnrollmentDate، Credits و Degree از جدول Students
فیلد AdminDate از جدول Admins
فیلدهای BusinessCredits و Discipline از جدول BusinessStudents

یک فیلد با نام PersonCategory نیز اضافه شده است که «مقداری عددی» می‌پذیرد و برای «متمایز کردن رکوردها» استفاده

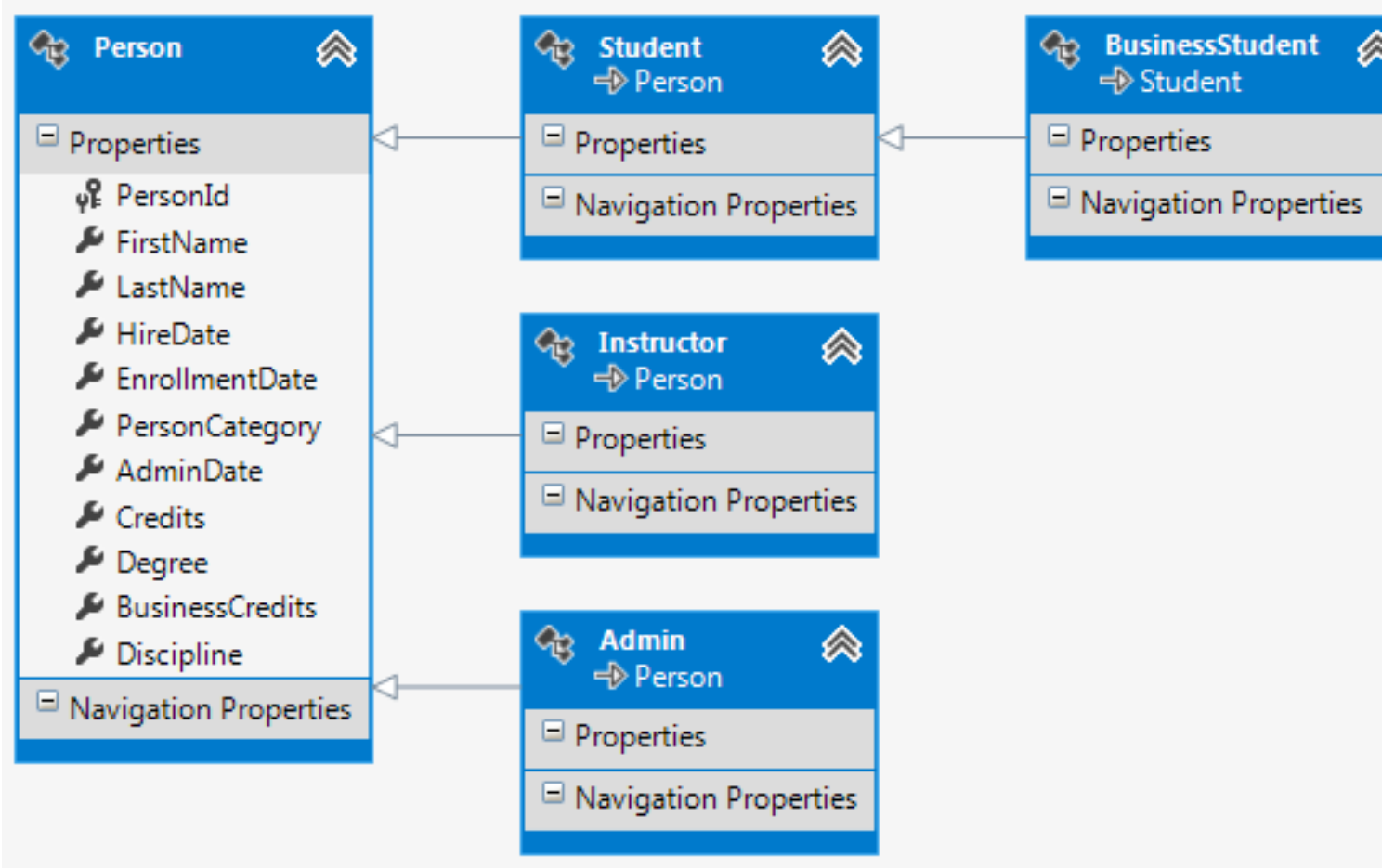
می‌شود:

1 ، نمایانگر Student
2 ، نمایانگر Instructor
3 ، نمایانگر Admin
4 ، نمایانگر BusinessStudent

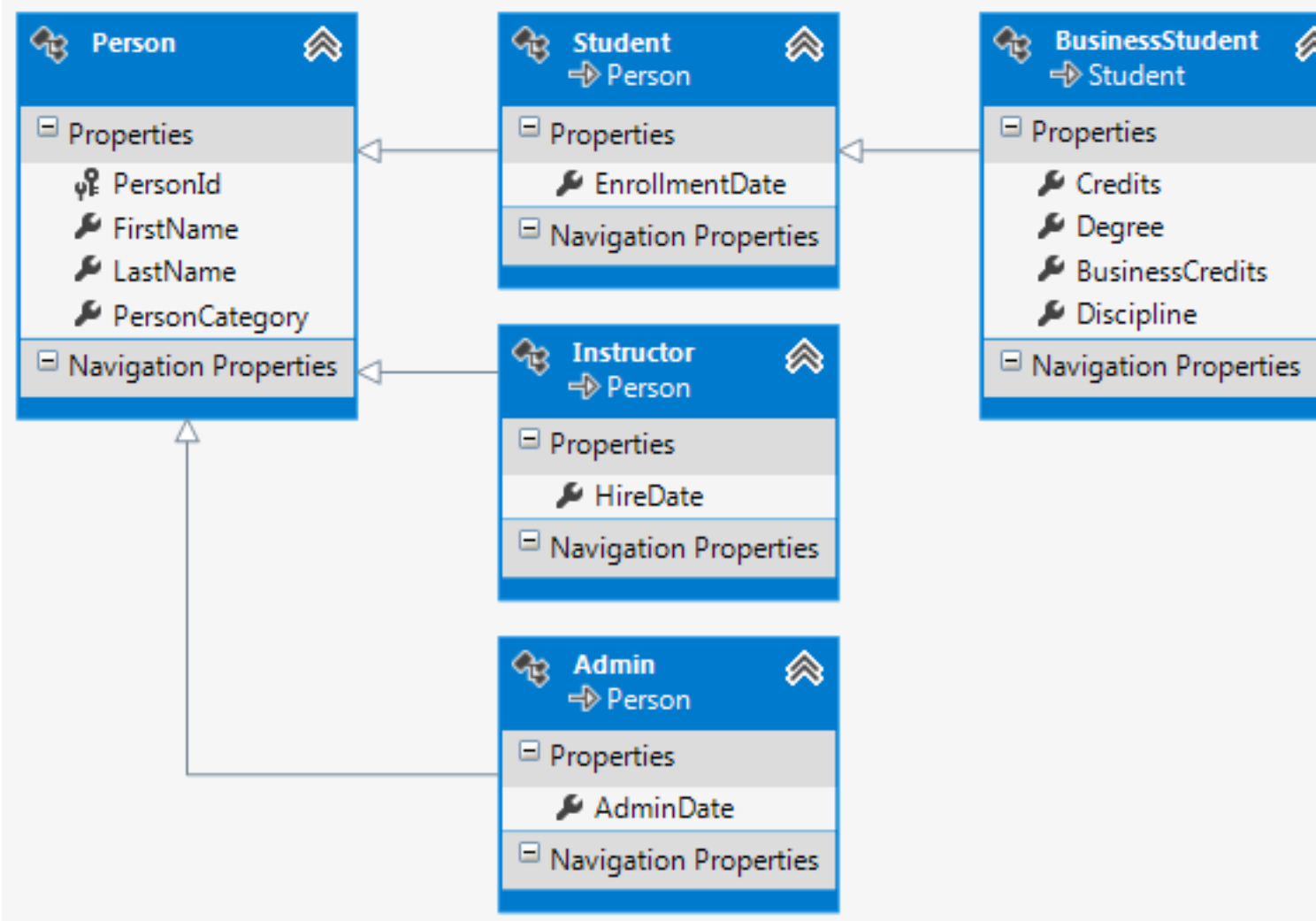
از این جدول می‌خواهیم به مدل قسمت اول برسیم اما این بار با استفاده از روش TPH. در شکل زیر، جدول Persons به صورت مدل شده در برنامه نشان داده شده است.



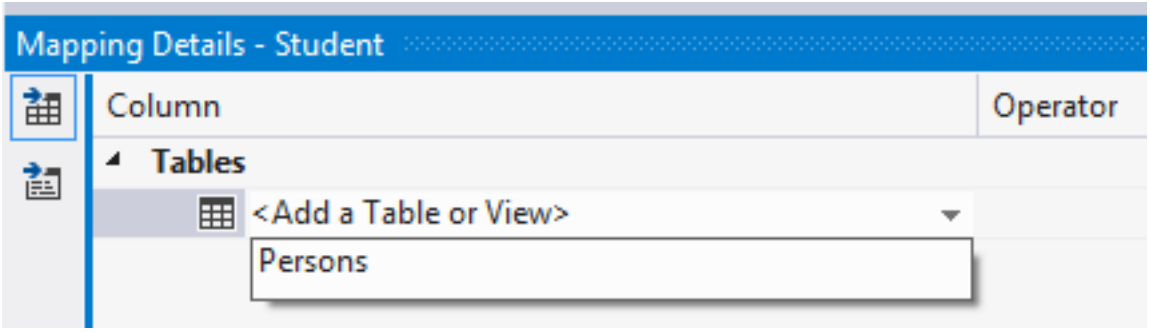
حال باید چهار موجودیت مشتق شده را به مدل اضافه کنیم. موجودیت‌های Student، Instructor و Admin را با کلیک راست بر روی EDM Designer و انتخاب گزینه‌ی Entity از منوی Add New ایجاد کنید. در فرمی که باز می‌شود، از قسمت Base type، Person را انتخاب کنید. موجودیت BusinessStudent را نیز ایجاد و موجودیت پایه‌ی آن را موجودیت Student در نظر بگیرید. مدل ایجاد شده تا اینجای کار در شکل زیر مشخص است.



حال باید خصیصه‌های موجودیت Person را به موجودیت‌های مشتق شده منتقل کرد. بدین منظور، هر خصیصه از موجودیت Person را انتخاب، کلیدهای Ctrl+X را فشار دهید، سپس بر روی قسمت Properties موجودیت مشتق شده‌ی مورد نظر رفته و کلیدهای Ctrl+V را فشار دهید. نتیجه در شکل زیر نشان داده شده است.

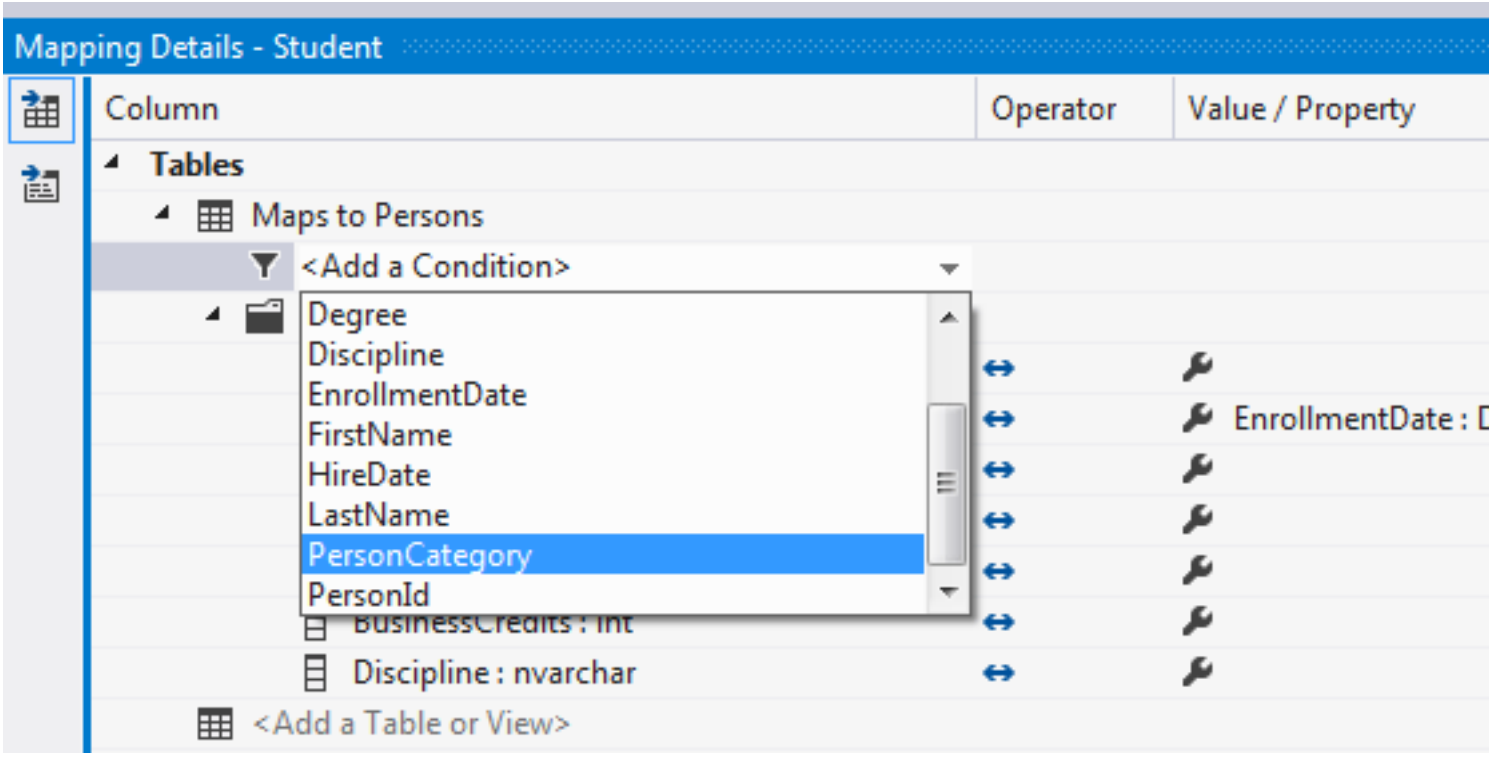


اکنون زمان آن رسیده است تا جدول متناظر با هر یک از موجودیت‌های مشتق شده را معرفی کنیم. تمامی موجودیت‌های مشتق شده از جدول Persons استفاده می‌کنند. بر روی هر یک از آنها کلیک راست کرده و گزینه‌ی Table Mapping را انتخاب کنید.
پنجره ی Mapping Details نشان داده می‌شود. ابتدا بر روی عبارت Add a Table or View و سپس بر روی نشانگر رو به پایینی که کنار آن ظاهر می‌شود کلیک کنید (شکل زیر).

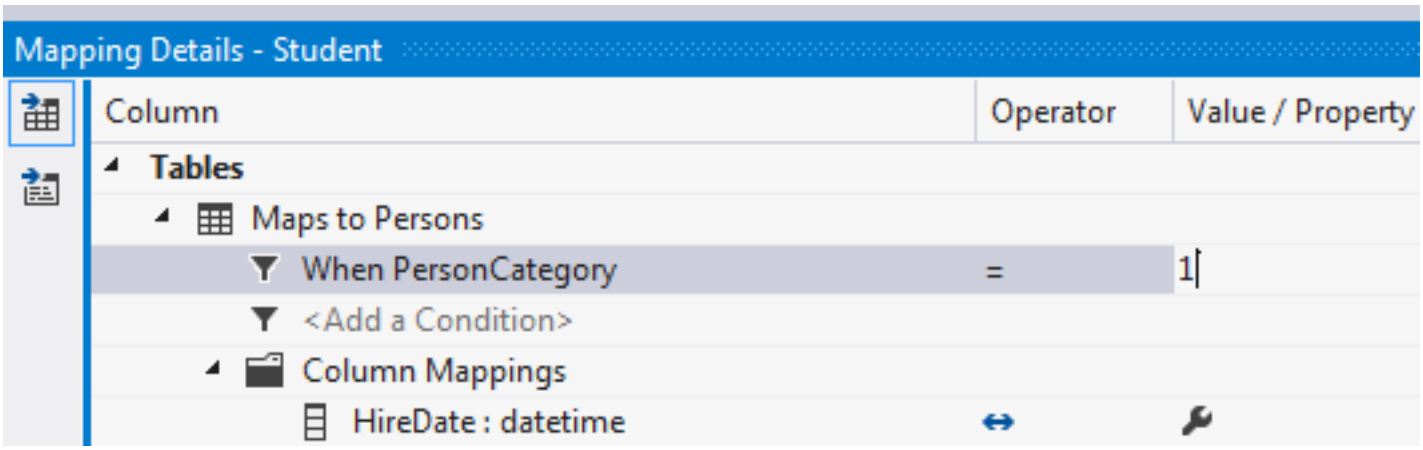


آیتم Persons را انتخاب کنید. اکنون باید فیلد تفکیک کننده‌ی رکوردها را مشخص کنیم. برای این حالت باید یک شرط ایجاد نمود.
در همان پنجره‌ی Mapping Details، عبارتی با عنوان Add a Condition وجود دارد. بر روی آن کلیک و در لیستی که ظاهر می‌شود،

آیتم PersonCategory را انتخاب کنید (شکل زیر).



سپس در ستون Value/Property، مقدار آن را "1" قرار دهید (شکل زیر).



تناظر میان موجودیت و جدول Persons و مقداردهی مناسب به فیلد متمایز کننده را برای تمامی موجودیت‌های مشتق شده انجام دهید. دلیل این کار این است که EF بداند هر رکورد در چه زمانی باید به چه موجودیتی تبدیل شود. دقت کنید که پیشتر به مقدار فیلد متمایز کننده برای هر موجودیت اشاره کردیم. نکته‌ی مهم اینکه یک شرط نیز باید برای موجودیت Person ایجاد و مقدار فیلد متمایز کننده‌ی آن را "صفر" تعریف کنید.
مثال ما آماده است. آن را امتحان می‌کنیم.

```
using (PersonDbEntities context = new PersonDbEntities())
{

    var people = from p in context.Persons
                 select p;

    foreach (Person person in people)
    {
        Console.WriteLine("{0}, {1}",
            person.LastName,
            person.FirstName);

        if (person is Student)
            Console.WriteLine("    Degree: {0}",
                ((Student)person).Degree);

        if (person is BusinessStudent)
            Console.WriteLine("    Discipline: {0}",
                ((BusinessStudent)person).Discipline);
    }

    Console.ReadLine();
}
```

چه کدهای آشنایی! بله، این کدها همان کدهایی هستند که برای مثال روش TPT استفاده کردیم و بدون کوچکترین تغییری در اینجا نیز قابل استفاده هستند.

مزایای روش TPH

سرعت بالای عملیات CRUD، به دلیل وجود تمامی داده‌ها در یک جدول

تعداد جداول در پایگاه داده، کم و مدیریت آنها آسان‌تر است

معایب روش TPH

افزونگی داده ها. مقادیر برخی ستون‌ها برای بعضی از رکوردها، حاوی مقدار NULL است و تعداد این ستون‌ها به تعداد زیر مجموعه‌ها ارتباط دارد

عیب اول، باعث می‌شود تا در صورتی که داده‌ها به صورت دستی تغییر پیدا کنند، جامعیت داده‌ها از بین برود

افزایش بی دلیل حجم داده ها

اضافه و حذف کردن موجودیت‌ها به مدل، عملی زمانبر و پیچیده است

سومین روش، TPC است که در EF Designer، پشتیبانی از پیش تعبیه شده برای آن وجود ندارد و باید از طریق ویرایش فایل edmx. به آن رسید. استفاده از این روش توصیه نمی‌شود.

نظرات خوانندگان

نویسنده: محمد فریدونی
تاریخ: ۱۳۹۲/۰۵/۱۹ ۱۳:۲۰

سلام؛ با توجه به دو مدلی که در ارث بری می‌تونیم بکار ببریم ، روش مناسب و بهتر، با توجه مزایا و معایبی که فرمودید کدوم روش است ؟

روش TPT ؟

یا روش

روش TPH ؟

نویسنده: محمد پهلوان
تاریخ: ۱۳۹۲/۰۸/۲۱ ۱۳:۴۶

روش‌های ذکر شده در اینجا نیز بررسی شده است