بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

پیاده سازی وراثت با EF6 در یک برنامه ی MVC

مدرس : مهندس افشین رفوآ

دوره آموزش MVC

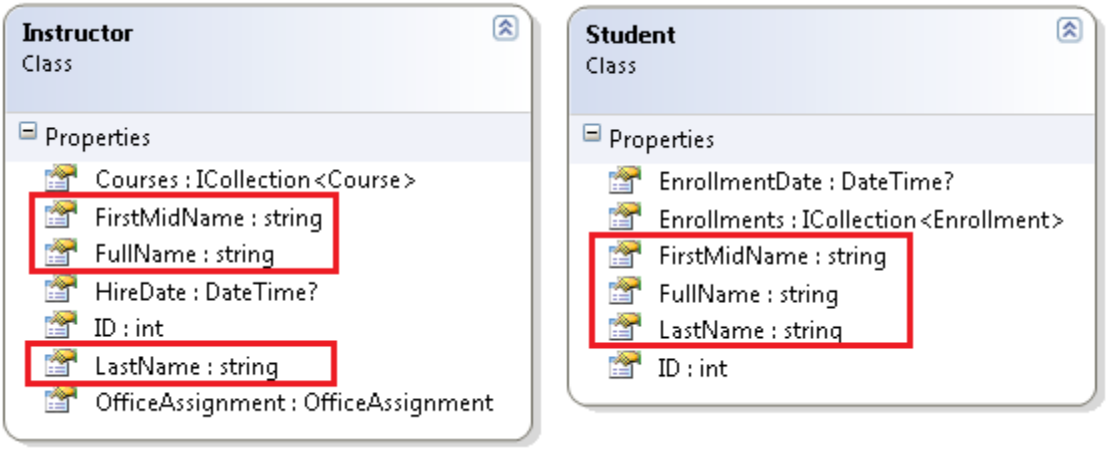
پیاده سازی وراثت با EF6 در یک برنامه ی MVC

در درس قبلی مدیریت خطاهای مربوط به همروندی (به انگلیسی concurrency) را آموختیم. آموزش حاضر نحوه ی پیاده سازی وراثت در data model را آموزش می دهد.

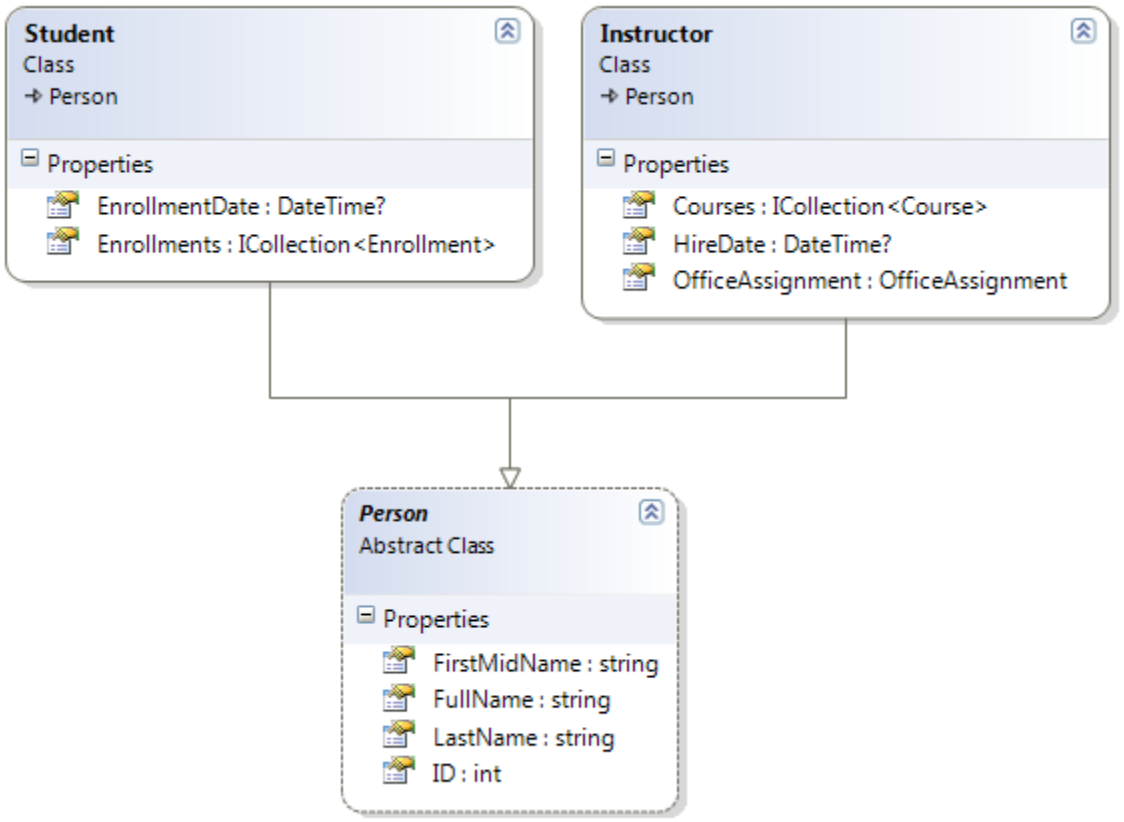
در برنامه نویسی شی گرا، با بهره گیری از ارث بری، می توان استفاده ی مجدد از کد (code reuse) را سهولت بخشید. در این مبحث کلاس های Instructor و Student را طوری تنظیم خواهیم کرد که از کلاس پایه ی Person ارث بری داشته باشند. این کلاس پایه دربردارنده ی خاصیت هایی (property) همچون LastName می باشد که بین هر دو کلاس instructors و students مشترک می باشد. در این درس هیچ صفحه ی وبی را ویرایش نخواهیم کرد، ولی برخی از کدها را تغییر می دهیم. این اصلاحات به صورت خودکار در پایگاه داده منعکس (اعمال) خواهد شد.

روش های نگاشت وراثت به جداول پایگاه داده

در این بخش یک data model به نام School داریم. این data model دارای دو کلاس به نام های Student و Instructor می باشد. دو کلاس یاد شده هر کدام دارای خواص مشترک و یکسان متعددی هستند که در تصویر زیر مشاهده می کنید:

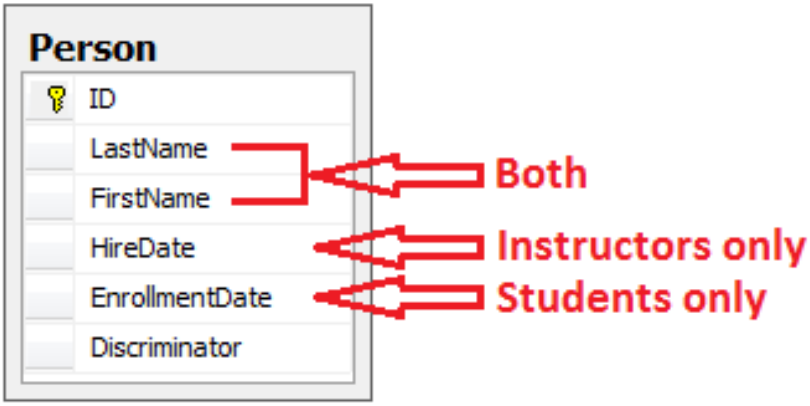


فرض کنید می خواهیم کد اضافه بر سازمان که متعلق به خواص مشترک بین دو موجودیت است را حذف کنیم و یا کد سرویسی را بنویسیم که اسم ها را صرف نظر از اینکه از کدام کلاس گرفته شده، فرمت کند. برای این منظور می توان یک کلاس پایه به نام Person ایجاد کرد که فقط دربردارنده ی آن خواص مشترک باشد، سپس کاری کنید که موجودیت های Student و Instructor از کلاس پایه ی نام برده property های خود را به ارث ببرند، همان طور که در این تصویر مشاهده می کنید:



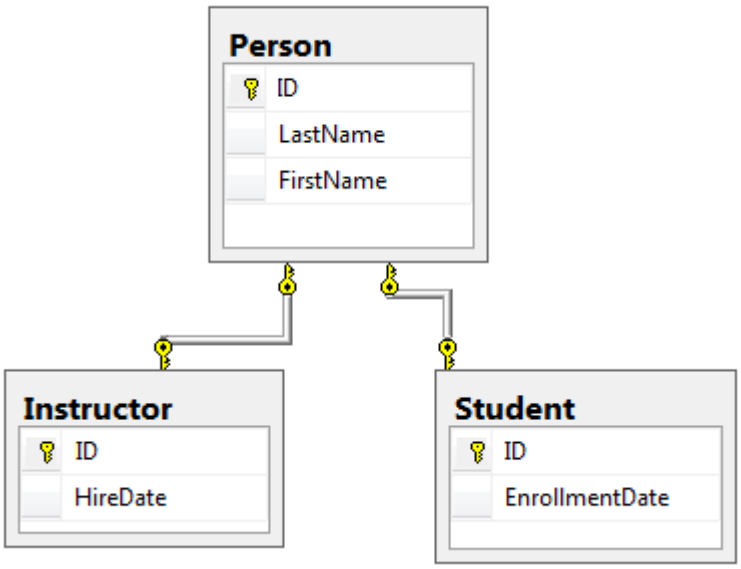
راه های مختلفی برای نمایش و ارائه ی این ساختار ارث بری در پایگاه داده ی مورد نظر وجود دارد. می توان این کار را از طریق یک جدول Person انجام داد که اطلاعات مربوط به کلاس های student و instructor را در یک صفحه ی واحد می گنجاند، به گونه ای که برخی از ستون ها فقط به instructors (HireDate) و برخی دیگر فقط به students (EnrollmentDate) و برخی هم به هر دو اعمال شود (LastName و FirstName).

برای این که مشخص شود هر سطر نشانگر و نماینده ی کدام نوع (type) است، اغلب یک ستون discriminator (تفکیک گر) در نظر می گیریم. به عنوان مثال، ستون discriminator می تواند "Instructor" را برای instructors و "Student" را برای students داشته باشد.



این الگو یا مدل ایجاد ساختار ارث بری موجودیت از یک جدول پایگاه داده را ارث بری به روش TPH (table-per-hierarchy) می نامند. 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.

روش دیگر این است که کاری کنید پایگاه داده بیشتر شبیه ساختار ارث بری باشد. برای مثال، فیلد های name را در جدول person داشت و برای فیلد های date جداول جداگانه ی student و Instructor را در نظر گرفت.



به این الگو یا روش ارث بری که در آن به ازای هر کلاس entity یک جدول ایجاد می شود، TPT (table per type) گفته می شود. بنابراین در این روش خاصیت های مشترک در جدول پایه قرار دارند و به ازای هر زیر مجموعه نیز یک جدول مجزا ایجاد می‌شود.

در کل الگوهای ارث بری TPC و TPH کارایی بهتری را نسبت به مدل TPT در EF ارائه می دهند، زیرا الگوی TPT ناچارا به join query های پیچیده منتهی می شود که افت کارایی را به دنبال دارد.

در این آموزش الگوی ارث بری TPH را پیاده سازی خواهیم کرد. TPH در واقع الگوی ارث بری پش فرض در EF می باشد، بنابراین تنها کاری که می بایست انجام دهید، ایجاد کلاس Person، ویرایش کلاس های Student و Instructor به گونه ای که از کلاس پایه ی person ارث بری داشته باشند، افزودن کلاس جدید به DbContext و ایجاد یک migrations است.

ایجاد کردن کلاس Person

داخل پوشه ی Models، فایل Person.cs را ایجاد کرده و کد زیر را جایگزین template code نمایید:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContosoUniversity.Models
{
  public abstract class Person
```

```
  {
    public int ID { get; set; }

    [Required]
    [StringLength(50)]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }
    [Required]
    [StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
    [Column("FirstName")]
    [Display(Name = "First Name")]
    public string FirstMidName { get; set; }

    [Display(Name = "Full Name")]
    public string FullName
    {
      get
      {
        return LastName + ", " + FirstMidName;
      }
    }
  }
}
```

ارث بری کلاس های Student و Instructor از کلاس پایه ی Person

در پوشه ی Instructor.cs ، کلاس Instructor را از Person مشتق (ارث بری) کرده ولی کلید و فیلدهای name را از آن حذف کنید. پس از ویرایش کد بدین شکل خواهد بود:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContosoUniversity.Models
{
  public class Instructor : Person
  {
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    [Display(Name = "Hire Date")]
    public DateTime HireDate { get; set; }

    public virtual ICollection<Course> Courses { get; set; }
    public virtual OfficeAssignment OfficeAssignment { get; set; }
  }
}
```

تغییرات مشابه را به فایل Student.cs اعمال کنید. کلاس Student پس از اصلاح به صورت زیر درخواهد آمد:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContosoUniversity.Models
{
  public class Student : Person
  {
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    [Display(Name = "Enrollment Date")]
    public DateTime EnrollmentDate { get; set; }

    public virtual ICollection<Enrollment> Enrollments { get; set; }
  }
}
```

افزودن نوع Person Entity به Model

در فایل SchoolContext.cs، یک خاصیت DbSet برای نوع موجودیت person، اضافه کنید:

```
public DbSet<Person> People { get; set; }
```

این تمام آن چیزی است که EF برای پیکربندی و تنظیم الگوی ارث بری TPH به آن نیاز دارد. خواهید دید که پایگاه داده پس از بروز رسانی، بجای جداول Student و Instructor، جدول Person را خواهد داشت.

ایجاد و بروز آوری فایل Migrations

در پنجره ی PMC، دستور زیر را وارد کنید:

```
Add-Migration Inheritance
```

دستور Update-Database را در پنجره ی PMC اجرا کنید. در اینجا اجرای دستور با شکست مواجه می شود. دلیلش این است که داده هایی از پیش وجود دارند که Migrations نمی داند چگونه اداره کند (با آن ها چه کار کند). از این رو یک پیغام خطا دریافت می کنید که متن آن به شرح زیر است:

Could not drop object 'dbo.Instructor' because it is referenced by a FOREIGN KEY constraint.

"از آنجایی که یک محدودیت کلید خارجی (FOREIGN KEY constraint) به شی 'dbo.Instructor' اشاره دارد، حذف آن امکان پذیر نیست."

فایل Migrations\<timestamp>_Inheritance.cs را باز کرده و کد زیر را جایگزین کد زیر کنید:

```
public override void Up()
{
  // Drop foreign keys and indexes that point to tables we're going to drop.
  DropForeignKey("dbo.Enrollment", "StudentID", "dbo.Student");
  DropIndex("dbo.Enrollment", new[] { "StudentID" });

  RenameTable(name: "dbo.Instructor", newName: "Person");
  AddColumn("dbo.Person", "EnrollmentDate", c => c.DateTime());
  AddColumn("dbo.Person", "Discriminator", c => c.String(nullable: false, maxLength: 128, defaultValue:
"Instructor"));
  AlterColumn("dbo.Person", "HireDate", c => c.DateTime());
  AddColumn("dbo.Person", "OldId", c => c.Int(nullable: true));

  // Copy existing Student data into new Person table.
  Sql("INSERT INTO dbo.Person (LastName, FirstName, HireDate, EnrollmentDate, Discriminator, OldId) SELECT
LastName, FirstName, null AS HireDate, EnrollmentDate, 'Student' AS Discriminator, ID AS OldId FROM
dbo.Student");

  // Fix up existing relationships to match new PK's.
  Sql("UPDATE dbo.Enrollment SET StudentId = (SELECT ID FROM dbo.Person WHERE OldId =
Enrollment.StudentId AND Discriminator = 'Student')");

  // Remove temporary key
  DropColumn("dbo.Person", "OldId");

  DropTable("dbo.Student");

  // Re-create foreign keys and indexes pointing to new table.
  AddForeignKey("dbo.Enrollment", "StudentID", "dbo.Person", "ID", cascadeDelete: true);
  CreateIndex("dbo.Enrollment", "StudentID");
}
```

کد زیر عملیات بروز رسانی زیر را انجام می دهد:

1. محدودیت های کلید خارجی (foreign key constraint) و اندیس هایی را که به جدول Student اشاره دارد را حذف می کند.

2. اسم جدول Instructor را به Person تغییر داده و اصلاحاتی را که لازم است تا بتواند داده های Student را ذخیره کند، اعمال می کند:

1. یک فیلد EnrollmentDate برای students ایجاد می کند که null پذیر (nullable) می باشد.
2. یک ستون discriminator (تفکیک گر) اضافه می کند که مشخص می کند آیا آن سطر مربوط (متعلق) به یک student می باشد یا instructor.
3. فیلدهای HireDate را null پذیر می کند زیرا سطر student (دارنده ی اطاعات مربوط به داشنجویان) دارای تاریخ استخدام (hire date) نخواهد بود.
4. یک فیلد موقتی ایجاد می کند که از آن برای بروز رسانی کلیدهای خارجی (foreign key) استفاده می شود. این کلیدها به دانشجویان اشاره می کند. زمانی که شما students را در جدول Person جای گذاری می کنید، مقادیر کلید اصلی جدید را می گیرد.

3. داده ها را از جدول Student کپی کرده و در جدول Person جای گذاری می کند. این باعث می شود مقادیر اصلی (primary key values) جدید به students تخصیص یابد.
4. مقادیر کلید خارجی که به students اشاره دارد را اصلاح می کند.
5. محدودیت های کلید خارجی (foreign key constraints) و اندیس ها را مجدد ایجاد کرده و کاری می کند که این بار به جدول person اشاره کنند.

(اگر نوع کلید اصلی را بجای integer (عدد صحیح)، GUID (شناسه ی منحصر بفرد سراسری) انتخاب می کردید، در آن صورت لزومی نداشت مقادیر کلید اصلی تغییر یافته و بسیاری از این گام ها حذف می شدند.)

دستور update-database را بار دیگر اجرا کنید.

(در یک سیستم production، اگر می خواستید به نسخه ی قبلی پایگاه داده برگردید بایستی تغییرات مربوطه را به متد Down نیز اعمال می کردید.)

نکته: احتمال دارد حین انتقال داده ها (migration) و ایجاد تغییراتی در schema، با پیام های خطایی دیگر مواجه شوید. اگر با خطاهای migration مواجه شدید که برطرف کردن آن برای شما امکان پذیر نیست، می توانید connection string را در فایل Web.config ویرایش کرده یا پایگاه داده را حذف کنید و آموزش را ادامه دهید. ساده ترین راه این است که اسم پایگاه داده را در فایل Web.config تغییر دهید. برای مثال، اسم پایگاه داده را به ContosoUniversity2 تغییر دهید، مانند نمونه ی زیر:

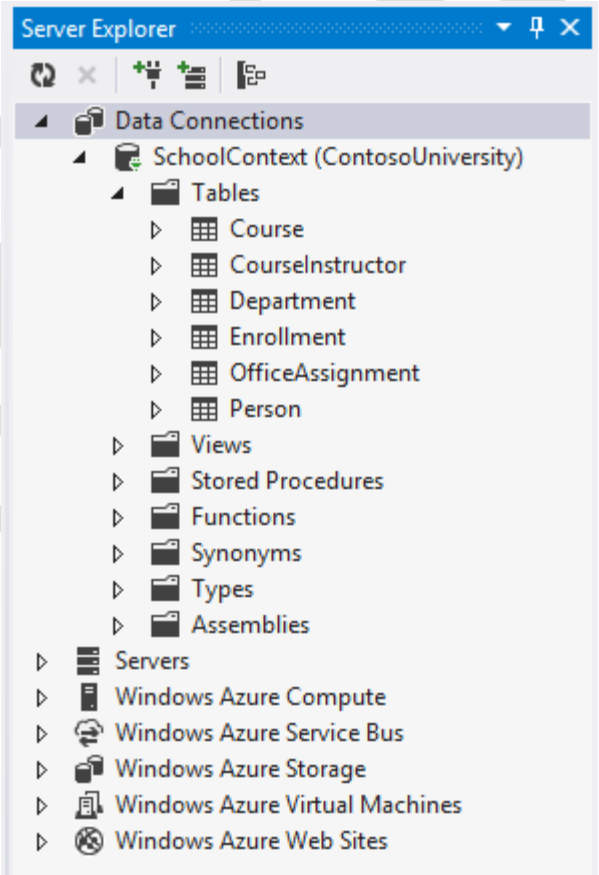
```
<add name="SchoolContext"
  connectionString="Data Source=(LocalDb)\v11.0;Initial Catalog=ContosoUniversity2;Integrated Security=SSPI;"

  providerName="System.Data.SqlClient" />
```

با یک پایگاه داده ی جدید دیگر داده ای برای انتقال وجود ندارد و همچنین دستور update-database با موفقیت و بدون رخداد خطا اجرا خواهد شد.

تست

سایت را راه اندازی کرده و صفحات متعددی از آن را تست کنید. همه چیز به درستی کار خواهد کرد.

در پنجره ی Server Explorer، Data Connections\SchoolContext و سپس Tables را باز کنید. خواهید دید که جدول Person جایگزین جداول Student و Instructor شده است. حال اگر جدول Person را باز کنید، می بینید که جدول ذکر شده تمامی ستون های موجود در جداول Student و Instructor را دربردارد.



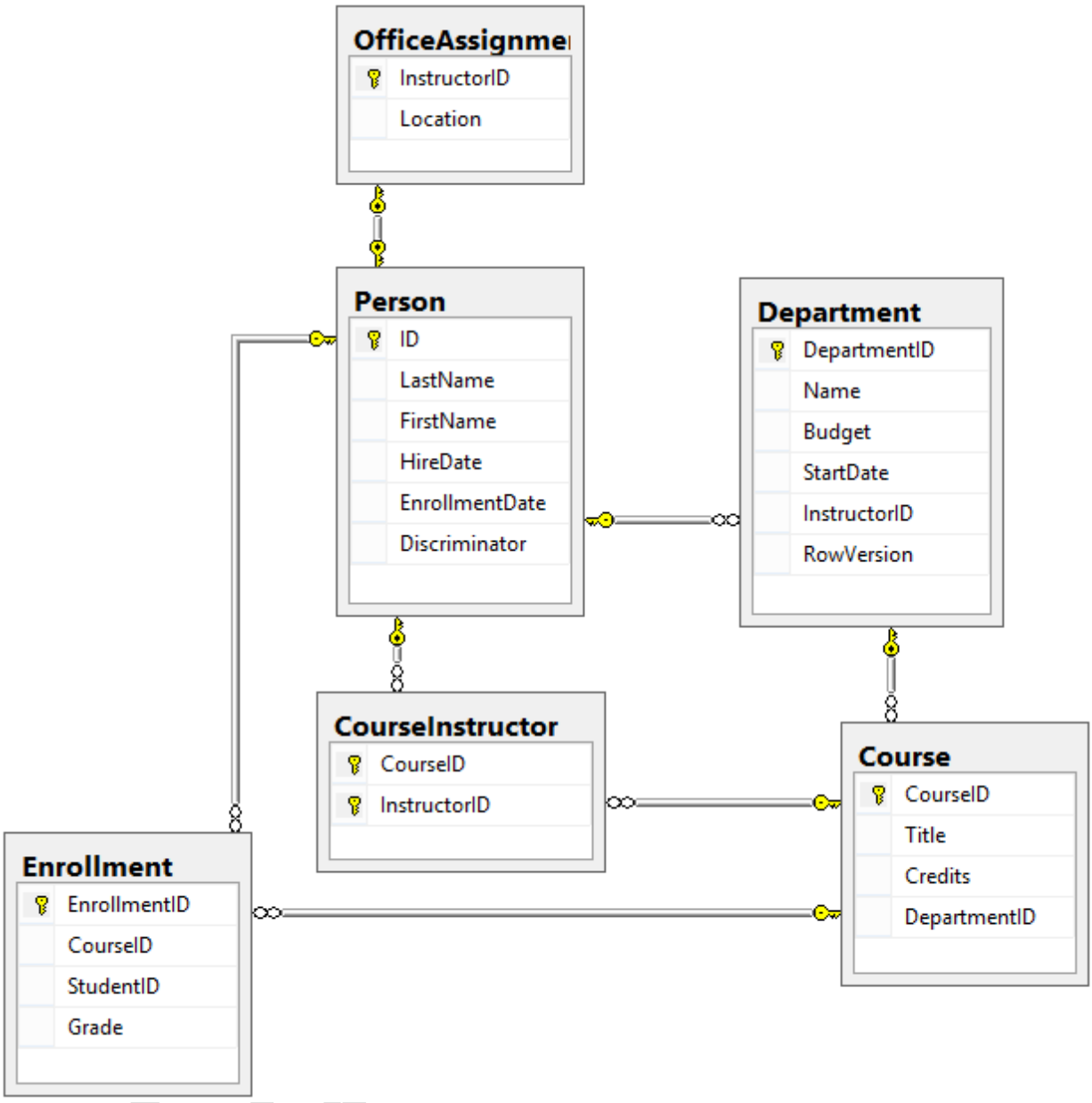
روی جدول Person راست کلیک کرده، سپس Show Table Data را کلیک کنید تا جدول مزبور همراه با ستون discriminator نمایش داده شود.

dbo.Person [Data]

Max Rows: 1000

	PersonID	LastName	FirstName	HireDate	EnrollmentDate	Discriminator
▶	1	Alexander	Carson	NULL	9/1/2010 12:00...	Student
	2	Alonso	Meredith	NULL	9/1/2012 12:00...	Student
	3	Anand	Arturo	NULL	9/1/2013 12:00...	Student
	4	Barzdukas	Gytis	NULL	9/1/2012 12:00...	Student
	5	Li	Yan	NULL	9/1/2012 12:00...	Student
	6	Justice	Peggy	NULL	9/1/2011 12:00...	Student
	7	Norman	Laura	NULL	9/1/2013 12:00...	Student
	8	Olivetto	Nino	NULL	9/1/2005 12:00...	Student
	9	Abercrombie	Kim	3/11/1995 1...	NULL	Instructor
	10	Fakhouri	Fadi	7/6/2002 12:...	NULL	Instructor
	11	Harui	Roger	7/1/1998 12:...	NULL	Instructor
	12	Kapoor	Candace	1/15/2001 1...	NULL	Instructor
	13	Zheng	Roger	2/12/2004 1...	NULL	Instructor
	14	Smith	Joe	NULL	5/23/2013 12:0...	Student
*	NULL	NULL	NULL	NULL	NULL	NULL

نمودار زیر ساختار جدید پایگاه داده School را نشان می دهد:



در این درس ارث بری به روش table-per-hierarchy را برای کلاس های Person، Student و Instructor پیاده سازی کردیم. در آموزش بعدی تعدادی سناریو بسیار پیچیده ی EF را در نظر خواهیم گرفت و نحوه ی مدیریت آن را تشریح خواهیم کرد.