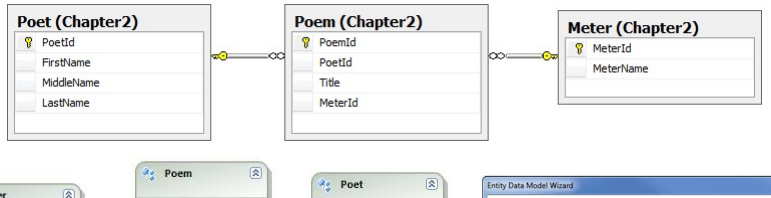
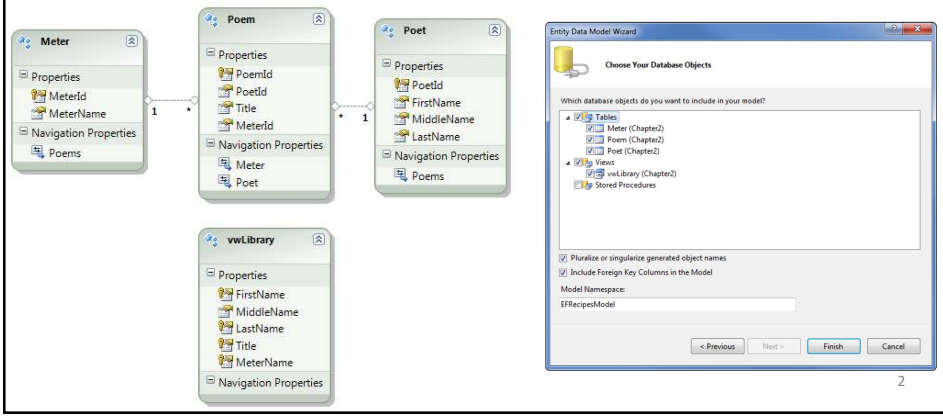
EDM primeri

Formiranje EDM na osnovu postojeće baze





Primer 1

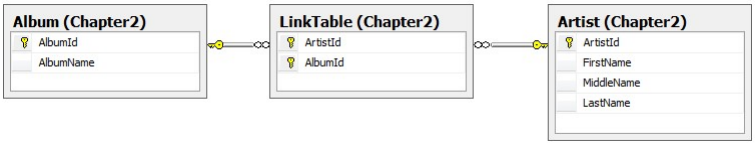
```
using(var ctx = new EFVezbanjeEntities())
{

    efPrimeri.Meter meter = new Meter { MeterName = "nazivMeter1" };
    meter = new Meter { MeterName = "nazivMeter2" };

    efPrimeri.Poet poet = new efPrimeri.Poet { FirstName = "Vojislav",
        LastName = "Ilic" };
    efPrimeri.Poem poem = new efPrimeri.Poem { Title = "Jesen" };
    poem.Meter = meter;
    poem.Poet = poet;
    ctx.Poem.Add(poem);
```

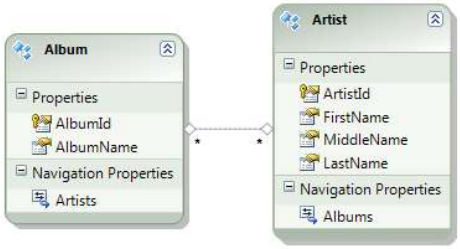
3

```
 poem = new efPrimeri.Poem { Title = "Vece" };
    poem.Meter = meter;
    poem.Poet = poet;
    ctx.Poem.Add(poem);

    poet = new efPrimeri.Poet { FirstName = "Jovan", LastName = "Ducic" };
    poem = new Poem { Title = "Susret" };
    poem.Meter = meter;
    poem.Poet = poet;
    ctx.Poem.Add(poem);

    ctx.SaveChanges();
}
```

4

Mapiranje tabele za povezivanje: bez dodatnih polja: više na više





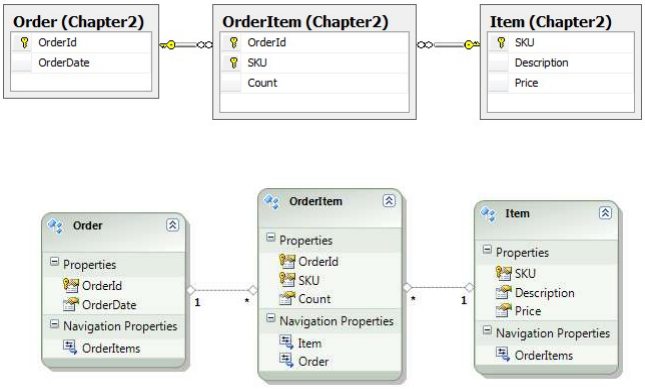
5

Primer 2

```
using (var ctx = new EFVezbanjeEntities()) // primer sa relacijom vise n
{
    // dodavanje jednog zapisa u Artist za dva zapisa u Album
    var artist = new Artist { FirstName = "Djordje", MiddleName="N", Las
};
    var album1 = new Album { AlbumName = "Devedesete" };
    var album2 = new Album { AlbumName = "Rani mraz" };
    artist.Album.Add(album1);
    artist.Album.Add(album2);
    ctx.Artist.Add(artist);

    // dodavanje jednog albuma za dva izvodjaca
    var artist1 = new Artist { FirstName = "Momcilo", MiddleName="M", La
    var artist2 = new Artist { FirstName = "Dejan", MiddleName = "M", La
    var album = new Album { AlbumName = "Pozitivna geografija" };
    album.Artist.Add(artist1);
    album.Artist.Add(artist2);
    ctx.Album.Add(album);

    ctx.SaveChanges();
}
```

6

Mapiranje tabele za povezivanje: sa dodatnim poljima: više na više

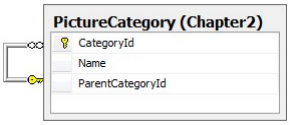


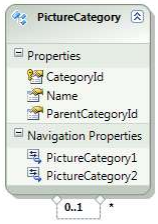
7

```
using (var context = new EFRecipesEntities())
{
    var order = new Order { OrderId = 1,
                            OrderDate = new DateTime(2010, 1, 18) };
    var item = new Item { SKU = 1729, Description = "Backpack",
                          Price = 29.97M };
    var oi = new OrderItem { Order = order, Item = item, Count = 1 };
    item = new Item { SKU = 2929, Description = "Water Filter",
                      Price = 13.97M };
    oi = new OrderItem { Order = order, Item = item, Count = 3 };
    item = new Item { SKU = 1847, Description = "Camp Stove",
                      Price = 43.99M };
    oi = new OrderItem { Order = order, Item = item, Count = 1 };
    context.Orders.AddObject(order);
    context.SaveChanges();
}
```

8

Modelovanje tabela koje imaju referencu na samu sebe



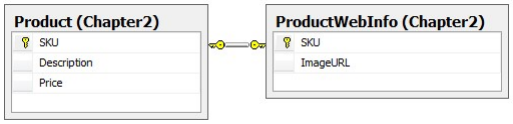


9

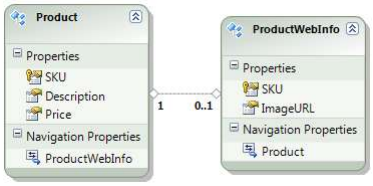
- Formiraju se dva navigaciona svojstva. Jedno se odnosi na roditeljsku kategoriju i ima 0..1 stranu relacije. Drugo se odnosi na dete i ima * uz relaciju. Preporučuje se promena imena imena ovih relacija.

10

Modelovanje tabela koje dele isti primarni ključ u jedan entitet



U bazi

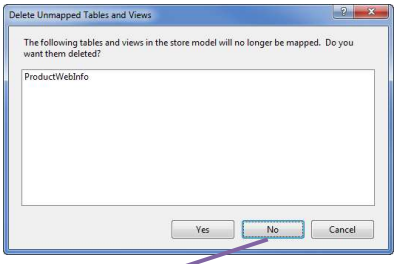


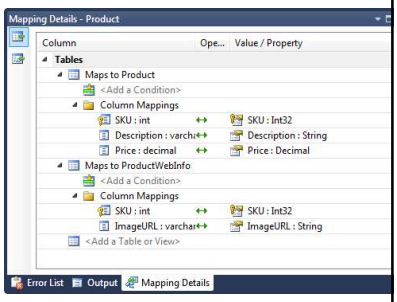
Nakon importovanja

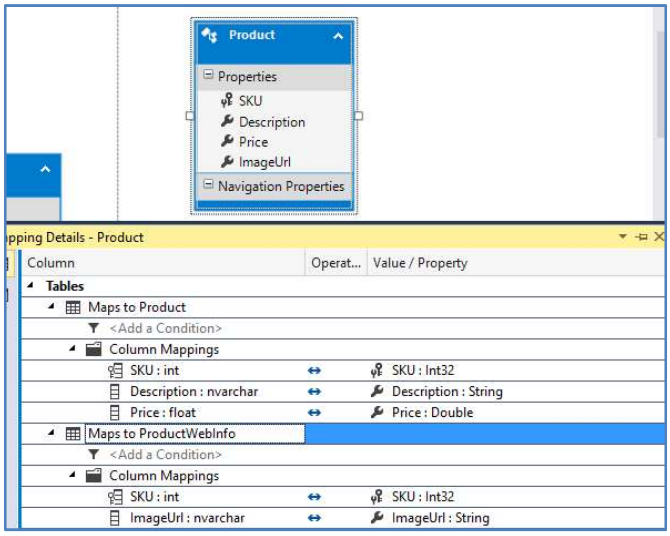
11

Postupak spajanja tabela:

- 1. kopirati *ImageURL* iz tabele ProductWebInfo u Product
- 2.Desni klik na ProductWebInfo, izabrati brisanje, nakon što se pojavi dijalog za brisanje entiteta, **odabrati No, čime se čuva definicija u Store modelu**
- 3. otvoriti prozor za editovanje mapiranja na entitetu Product. Odabrati dodatno ProductWebInfo tabelu za mapiranje







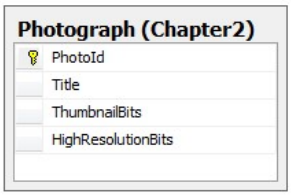
13

Primer 3

```
using (var ctx = new EFVezbanjeEntities())
{
    var product = new Product { SKU = 148, Description = "Opis1",
                          Price = 79.9, ImageUrl="/slika1.jpg" };
    ctx.Product.Add(product);

    product = new Product { SKU = 151, Description = "Opis2",
                        Price = 49.9, ImageUrl = "/slika2.jpg" };
    ctx.Product.Add(product);

    ctx.SaveChanges();
}
```

14

Podela jedne tabele na više entiteta

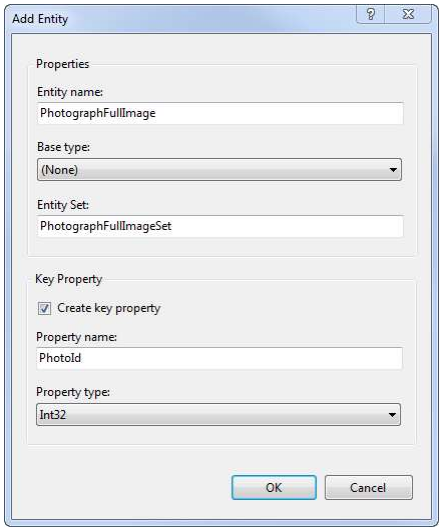


Na primer:

- Ako imamo tabelu sa nekoliko često korišćenih polja i sa nekoliko velikih, ali retko potrebnih polja, ona se deli na više entiteta.

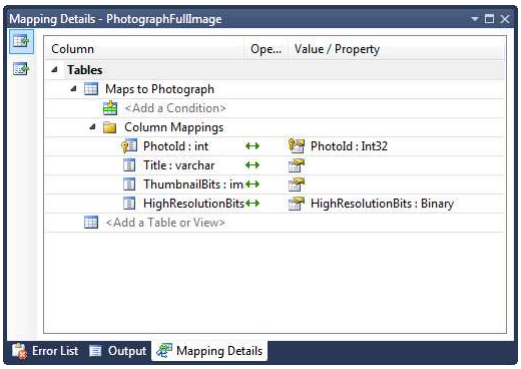
15

- Dodaje se novi entitet sa ključem i poljem koje se dobija kopiranjem iz postojeće tabele.
- Prebaciti polje *HighResolutionbits* u novi entitet



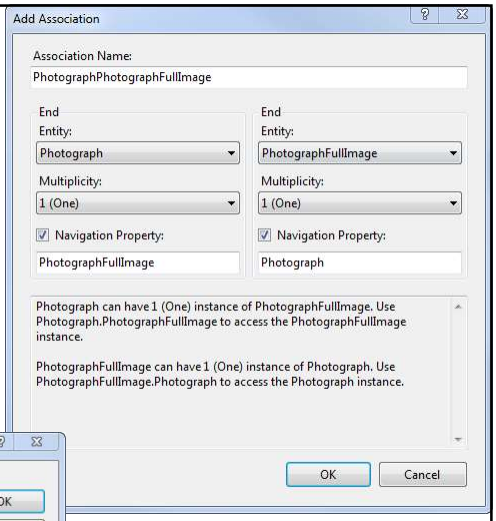
16

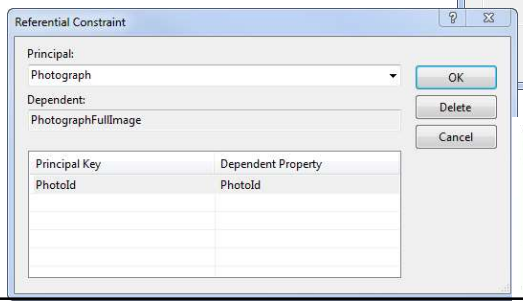
- Izvrši se mapiranje

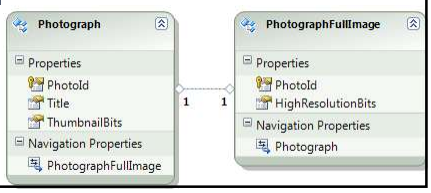


17

- Dodaje se asocijacija.
- Zatim se podesi svojstvo dodate asocijacije, desnim klikom na nju, *Referential Constraint*





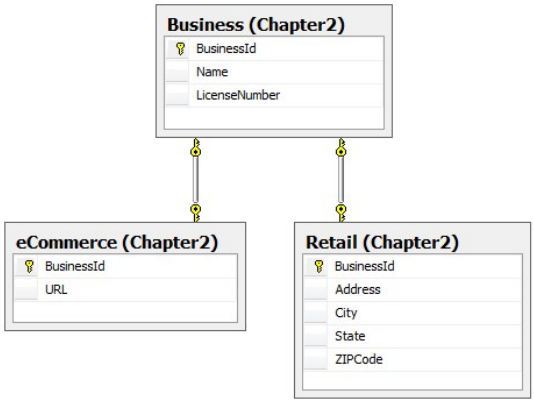


Primer 4

```
byte[] manjaSlika = new byte[400];
byte[] vecaSlika = new byte[2500];
using (var ctx = new EFVezbanjeEntities())
{
    var photo = new Photograph { PhotoId=2, Title="kuca1",
                                 ThumbnailBits = manjaSlika };
    var photoFull = new PhotographFullImage { PhotoId = 2,
                            HighResolutionBits = vecaSlika };
    photo.PhotographFullImage = photoFull;
    ctx.Photograph.Add(photo);

    ctx.SaveChanges();
}
```

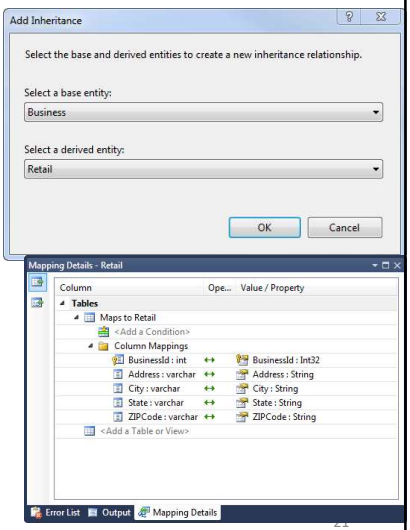
19

Više tabela sa dodatnim informacijama,
a želimo modelovanje koristeći TPT nasleđivanje



20

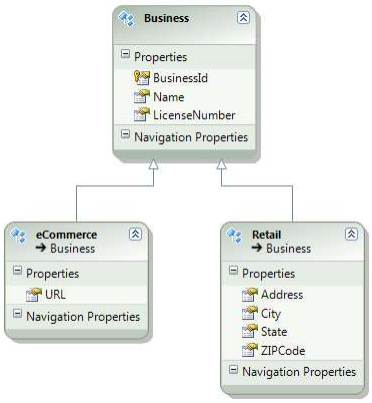
- Nakon formiranja modela, izbrišu se asocijacije između ovih entiteta.
- Desnim klikom na entitet dodaje se *Inheritance* i postavlja se Business kao roditeljska klasa.
- Izbriše se id u izvedenim klasama jer se nasleđuju.
- Dodati se mapiranje.
- Napomena: Obratiti pažnju na ograničenja. Na primer State polje može biti maksimalno dužine 2 karaktera.



21

Primer 5

```
using (var ctx = new EFVezbanjeEntities())
// primer sa relacijom vise na vise
{
    var r = new Retail {
        Name = "Posao1",
        LicenseNumber = "1x001",
        Address="Ustanicka 1",
        City="Beograd",
        State="Sr",
        ZIPCode="1"
    };

    ctx.Business.Add(r);
    ctx.SaveChanges();
}
```

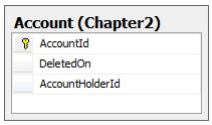


22

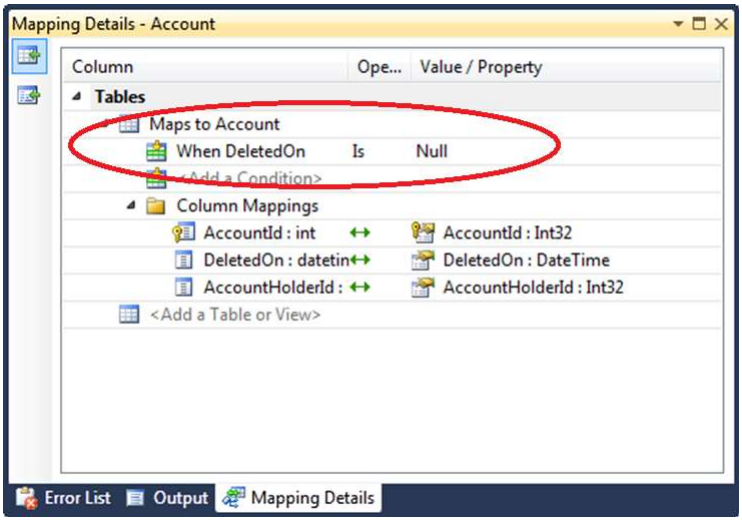
Formiranje posebnog entiteta primenom stalnog filtera

Na primer:

- Tabela Account ima polje *DeletedOn* koje ukazuje kada je ugašen račun.
- U aplikaciji treba da se koriste samo aktivni računi.



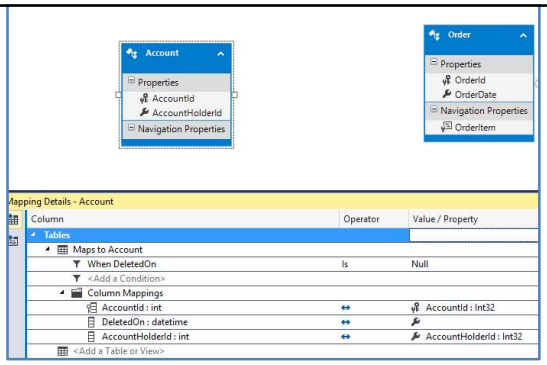
23



24

- U mapiranju u modelu potrebno je kliknuti na “Add a Condition”, odabrati kolonu DeletedOn. Kao operator odabrati “Is” a za vrednost “Null”. Na ovaj način se kreira uslov mapiranja kada je DeletedOn jednako Null.
- Pošto se kolona DeletedOn koristi za filtriranje, potrebno je izbrisati iz mapiranja (njena vrednost je uvek Null u ovom entitetu).

25



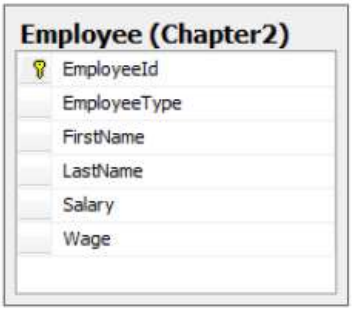

```
using (var ctx = new EFVezbanjeEntities()) // primer sa
relacijom vise na vise
{
    var a = new Account{AccountHolderId=2374};
    ctx.Account.Add(a);

    ctx.SaveChanges();
}
```

26

Modelovanje tabele sa diskriminativnim poljem u vise entiteta

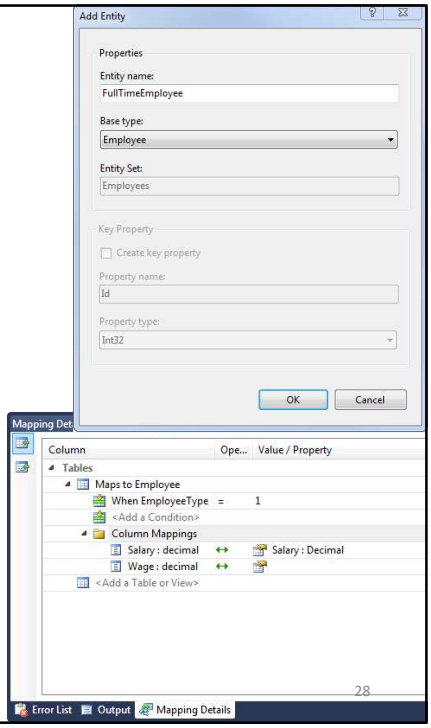
Na primer:

- Polje EmployeeType može imati samo jednu od mogućih vrednosti koje određuje vrstu zaposlenog. Vrednost ovog polja se koristi za filtriranje i formiranje novih entiteta.



27

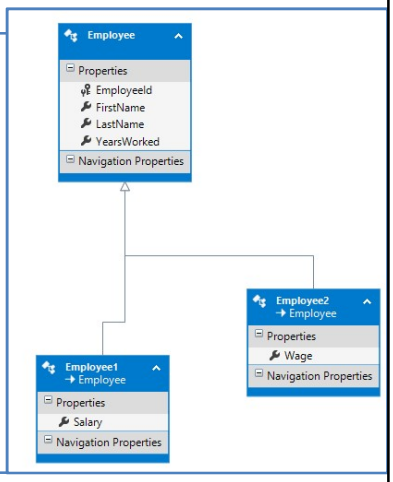
- Možemo da dodamo 2 nova entiteta u model. Na primer "FullTimeEmployee" odnosno "HourlyEmployee", oba izvedena iz entiteta "Employee".
- Koristeći cut/paste prebaciti Salary iz Employee u FullTimeEmployee, a Wage iz Employee u HourlyEmployee.
- Mapirati prebačene kolone u novim entitetima u odgovarajuće u tabelama, ali koristiti uslov na osnovu polja EmployeeType.
- Izmeniti svojstvo Abstract entiteta Employee u true.
- Izbrisati EmployeeType iz Employee



28

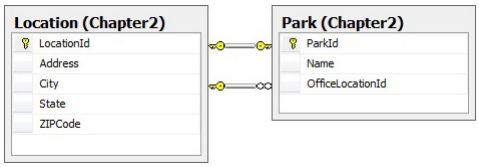
Primer 7

```
using (var ctx = new EFVezbanjeEntities())
// primer sa relacijom vise na vise
{
    var e1 = new Employee1 {
       FirstName="pera", LastName="peric",
       Salary=200000, YearsWorked=2
    };
    ctx.Employee.Add(e1);

    var e2 = new Employee2 {
       FirstName="aca", LastName="ilic",
       Wage=50000, YearsWorked=0
    };

    ctx.Employee.Add(e2);


    ctx.SaveChanges();
}
```



29

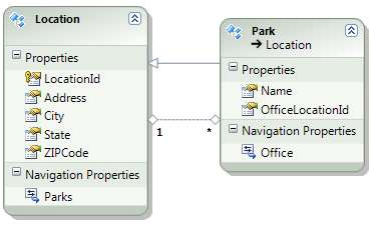
Modelovanje Is i Has relacija



- Dve tabele imaju relacije Is i Has i potrebno je ostvariti takvo modelovanje izmedju entiteta.
- Na primer, u bazi postoje tabele Park i Location. Svaki Park je istovremeno i Location i ima relaciju 1-1. Osim toga neki park može imati glavnu kancelariju sa adresom koja se takođe predstavlja tabelom Location. Dakle, neki entitet Park je izveden iz Location a može i imati Location

30

- Nakon generisanja modela, uvesti relaciju nasleđivanja između Park i Location, obrisati ParkId iz entiteta Park i srediti mapiranje tj. usmeriti ParkId na LocationId.

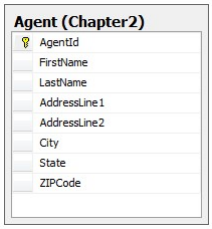


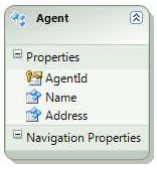

```
using (var context = new EFRecipesEntities())
{
    var park = new Park { Name = "11th Street Park",
                          Address = "801 11th Street", City = "Aledo",
                          State = "TX", ZIPCode = "76106" };
    var loc = new Location { Address = "501 Main", City = "Weatherford",
                             State = "TX", ZIPCode = "76201" };
    park.Office = loc;
    context.Locations.AddObject(park);
    park = new Park { Name = "Overland Park", Address = "101 High Drive",
                      City = "Springtown", State = "TX", ZIPCode = "76081" };
    loc = new Location { Address = "8705 Range Lane", City = "Springtown",
                         State = "TX", ZIPCode = "76081" };
    park.Office = loc;
    context.Locations.AddObject(park);
    context.SaveChanges();
}
```

31

Rad sa kompleksnim tipovima podataka

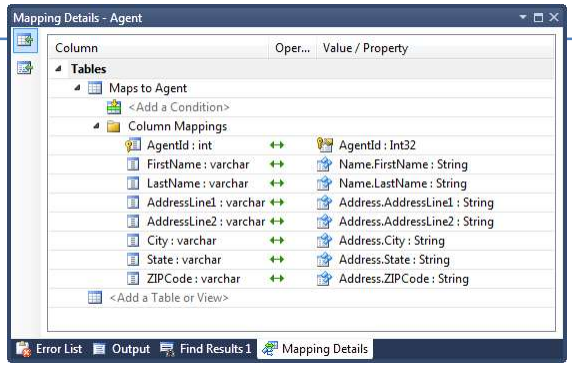
- Ako imamo entitet Agent kao na slici a želimo da sredimo podatke u dve grupe, kao na drugoj slici





32

- U modelu se selektuje FirstName i LastName, zatim na desni klik se odabere *Refactor Into Complex Type*.
- U Model Browser-u promeni se ime novog tipa iz ComplexType1 u Name. Ovo menja ime tipa u Agent entitetu.
- Kompleksni tip se može dodati i desnim klikom na površinu i odabirom Add->ComplexType
- U Model Browser-u se promeni ime u Address. Selektuju se polja od interesa (AddressLine1,…) odabere Cut, a zatim na novom tipu Paste.
- Zatim uraditi mapiranje



33

Primer 8

```
using (var ctx = new EFVezbanjeEntities()) // primer sa
relacijom vise na vise
{
    AddressT a1 = new AddressT{AddressLine1="linija1",
       AddressLine2="", City="Bgd", State="Sr", ZIPCode="1"};
    NameT n1 = new NameT{FirstName="joca", LastName="jocic"};
    var a = new Agent { Address = a1, Name=n1};
    ctx.Agent.Add(a);

    ctx.SaveChanges();
}
```

34

Izvršavanje sql upita

```
using (var context = new EFRecipesEntities())
{
    var sql = @"insert into Chapter3.Payment(Amount, Vendor) values (@Amount, @Vendor)";
    var parameters = new DbParameter[]
    {
        new SqlParameter {ParameterName = "Amount", Value = 99.97M},
        new SqlParameter {ParameterName = "Vendor", Value = "Ace Plumbing"}
    };

    var rowCount = context.Database.ExecuteSqlCommand(sql, parameters);

    parameters = new DbParameter[]
    {
        new SqlParameter {ParameterName = "Amount", Value = 43.83M},
        new SqlParameter
            {
                ParameterName = "Vendor",
                Value = "Joe's Trash Service"
            }
    };

    rowCount += context.Database.ExecuteSqlCommand(sql, parameters);
    Console.WriteLine("{0} rows inserted", rowCount.ToString());
}
```

35

Vraćanje objekta iz sql naredbe


```
using (var context = new
EFRecipesEntities())
{
    // insert student data
    context.Students.Add(new Student
        {
            FirstName = "Robert",
            LastName = "Smith",
            Degree = "Masters"
        });
    context.Students.Add(new Student
        {
            FirstName = "Julia",
            LastName = "Kerns",
            Degree = "Masters"
        });
    context.Students.Add(new Student
        {
            FirstName = "Nancy",
            LastName = "Stiles",
            Degree = "Doctorate"
        });
    context.SaveChanges();
}
```

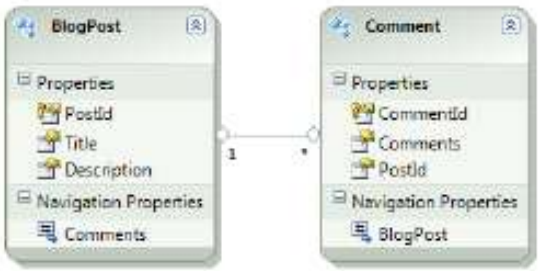
```
var sql = "select * from Chapter3.Student where Degree = @Major";
var parameters = new DbParameter[]
    {
        new SqlParameter {ParameterName = "Major", Value = "Masters"}
    };
var students = context.Database.SqlQuery<Student>(sql, parameters);
Console.WriteLine("Students...");
foreach (var student in students)
{
    Console.WriteLine("{0} {1} is working on a {2} degree",
                      student.FirstName, student.LastName,
student.Degree);
}
```

36

- Ovo se može primeniti i na kompleksne tipove, ali ne i na entitete koji sadrže kompleksne tipove ili nad kojima je realizovano nasleđivanje.

37

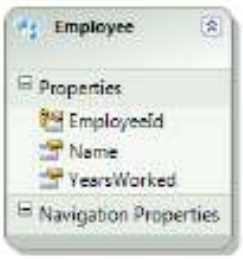
Pronalaženje entiteta, primer



- Između dva entiteta postoji asocijacija jedan na vise. Treba naci glavne entitete koji imaju bar jedan entitet detalja
- var posts = from post in context.BlogPosts
  where post.Comments.Any(()
  select post;

38

Dodeljivanje neke vrednosti ako polje ima vrednost null u upitu



- Polje YearsWorked može imati null.
- Može se koristiti operator ??

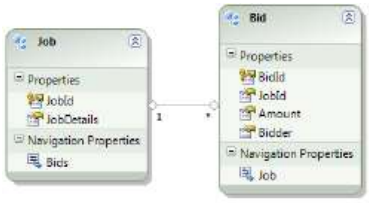
```
using(var context = new EFRecipesEntities()
{
   Console.WriteLine("Employees (using LINQ)");
   var employees = from e in context.Employees
                   select new {
                       Name=e.Name,
                       YearsWorked=e.YearsWorked ?? 0
                   }
```

39

Vraćanje više skupova iz uskladištenih procedura

Uskladištena procedura

```
begin
 select * from Chapter3.Job
 select * from Chapter3.Bid
end
```




```
using (var context = new EFRecipesEntities())
{
    var cs = @"Data Source=localhost\sqlexpress;Initial Catalog=EFRecipes;Integrated Security=True";
    var conn = new SqlConnection(cs);
    var cmd = conn.CreateCommand();
    cmd.CommandType = System.Data.CommandType.StoredProcedure;
    cmd.CommandText = "Chapter3.GetBidDetails";
    conn.Open();
    var reader = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    var jobs = ((IObjectContextAdapter) context).ObjectContext.Translate<Job>(reader, "Jobs",
        MergeOption.AppendOnly).ToList();
    reader.NextResult();
    ((IObjectContextAdapter) context).ObjectContext.Translate<Bid>(reader, "Bids", MergeOption.AppendOnly)
        .ToList();
    foreach (var job in jobs)
    {
        Console.WriteLine("\nJob: {0}", job.JobDetails);
        foreach (var bid in job.Bids)
        {
            Console.WriteLine("\tBid: {0} from {1}",
                bid.Amount.ToString(), bid.Bidder);
        }
    }

    Console.WriteLine("\nPress <enter> to continue...");
    Console.ReadLine();
}
```

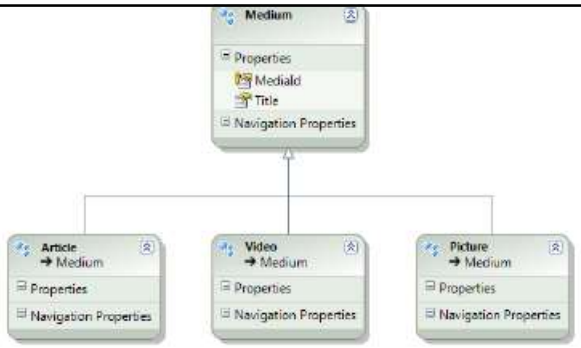
Vraćanje entiteta čija vrednost nekog svojstva odgovara vrednostima iz neke liste

Resenje

```
using (var context = new EFRecipesEntities())
{
    Console.WriteLine("Books (using LINQ)");
    List<string> cats = new List<string> { "Programming", "Databases" };
    var books = from b in context.Books
                where cats.Contains(b.Category.Name)
                select b;
```

41

Sortiranje po tipovima u slučaju nasleđivanja entiteta



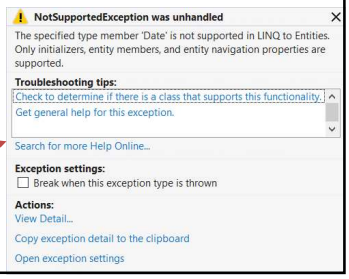

```
var allMedium = from m in context.Media
    let mediumtype = m is Article
        ? 1
        : m is Video ? 2 : 3
    orderby mediumtype
    select m;
Console.WriteLine("All Medium sorted by type...\n");
foreach (var medium in allMedium)
{
    Console.WriteLine("Title: {0} [{1}]", medium.Title, medium.GetType().Name);
}
```

42

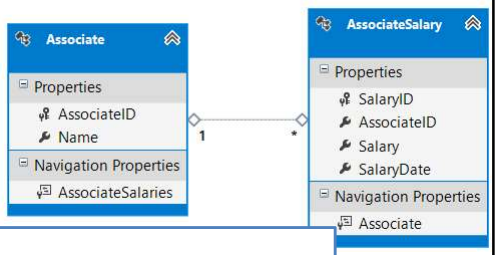
Grupisanje po datumu


```
var groups = from r in context.Registrations
             // leverage built-in TruncateTime function tp extract date portion
             group r by DbFunctions.TruncateTime(r.RegistrationDate)
             into g
             select g;
```

```
var groups = from r in context.Registrations
             group r by r.RegistrationDate.Value.Date // ne moze
             into g
             select g;
    foreach (var element in groups)
```



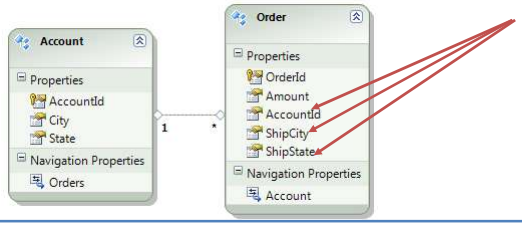
Dobijanje svih zapisa dva entiteta u asocijaciji jedan na više




```
Console.WriteLine("Using LINQ...");
var allHistory = from a in context.Associates
                 from ah in a.AssociateSalaries.DefaultIfEmpty()
                 orderby a.Name
                 select new
                 {
                     Name = a.Name,
                     Salary = (decimal?)ah.Salary,
                     Date = (DateTime?)ah.SalaryDate
                 };

Console.WriteLine("Associate Salary History");
foreach (var history in allHistory)
{
    if (history.Salary.HasValue)
        Console.WriteLine("{0} Salary on {1} was {2}", history.Name,
                          history.Date.Value.ToShortDateString(),
                          history.Salary.Value.ToString("C"));
    else
        Console.WriteLine("{0} --", history.Name);
}
```

44

Složeni “join” (po više kolona)

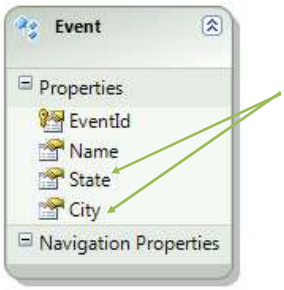



```
var orders = from o in context.Orders
             join a in context.Accounts on
             new {Id = o.AccountId, City = o.ShipCity, State = o.ShipState}
             equals
             new {Id = a.AccountId, City = a.City, State = a.State}
             select o;

Console.WriteLine("Orders shipped to the account's city, state...");
foreach (var order in orders)
{
    Console.WriteLine("\tOrder {0} for {1}", order.AccountId.ToString(),
        order.Amount.ToString());
}
```

45

Grupisanje po više svojstava

```
using (var context = new EFRecipesEntities())
{
    Console.WriteLine("Using LINQ");
    var results = from e in context.Events
        group e by new { e.State, e.City } into g
        select new
           {
              State = g.Key.State,
              City = g.Key.City,
              Events = g
           };
    Console.WriteLine("Events by State and City...");
    foreach (var item in results)
    {
        Console.WriteLine("{0}, {1}", item.City, item.State);
        foreach (var ev in item.Events)
        {
            Console.WriteLine("\t{0}", ev.Name);
        }
    }
}
```



46