PRIMENA EF

1

19.54

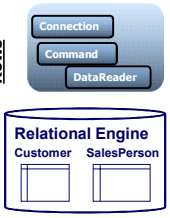
STARO: DATA ACCESS (ROWS)

Eksplicitna konekcija

```
void EmpsByDate(DateTime date) {
using( SqlConnection con = new SqlConnection(
        Settings.Default.AdventureWorksSQL)) {
  con.Open();

  SqlCommand cmd = con.CreateCommand();
      cmd.CommandText = @"
      SELECT SalesPersonID, FirstName, HireDate
      FROM SalesPerson sp
      INNER JOIN Employee e ON
              sp.SalesPersonID = e.EmployeeID
      INNER JOIN Contact c ON
              e.EmployeeID = c.ContactID
      WHERE e.HireDate < @date" ;
      cmd.Parameters.AddWithValue( "@date" , date);

  DbDataReader r = cmd.ExecuteReader();
  while(r.Read()) {
      Console.WriteLine(
      "{0:d}:\t{1}", r["HireDate"], r["FirstName"]);
  }
}
```

Nerazumljiva komanda

Rows



Entities ≠ Rows

Netipiziran rezultat-skup

NOVO: DATA ACCESS (OBJECTS)

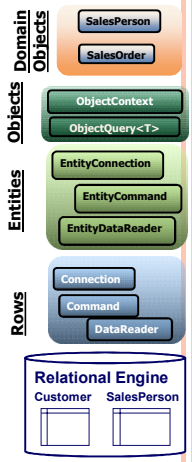
```
public partial class AdventureWorksEntities:System.Data.Objects.ObjectContext
{
   public System.Data.Objects.
         ObjectQuery<SalesOrder> SalesOrders
   { … }
   public System.Data.Objects.
         ObjectQuery<SalesPerson> SalesPeople
   { … }
}

void EmpsByDate(DateTime date) {

 using (AdventureWorksEntities aw =
                 new AdventureWorksEntities()) {

            var people =  from p in aw.SalesPeople
                          where p. HireDate < date
                          select p;

            foreach (SalesPerson p in people) {
                Console.WriteLine("{0:d}\t{1}",
                 p.HireDate, p.FirstName );
            }
}
```

Auto-Gen klase

Bez eksplicitne konekcije

LINQ izrazi

Jako tipizirani rezultat-skup

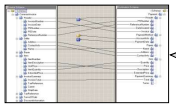


USING

```
{
    Font font1 = new Font("Arial", 10.0f);
    try
    {
        byte charset = font1.GdiCharSet;
    }
    finally
    {
        if (font1 != null)
            ((IDisposable)font1).Dispose();
    }
}
```

```
using (Font font1 = new Font("Arial", 10.0f))
{
    byte charset = font1.GdiCharSet;
}
```

ENTITY-RELATIONAL MAPPING



Correspondences
(specified by users)

```
Select ord#, prod#,
cust#
From  Shipped
⊆
Select ord#, prod#,
cust#
From Order Join Item
on ord#
```

Compiled Mappings
(View generation)

Upiti i Ažuriranja (drive the runtime)

- Deklarativno mapiranje
  - Omogućava da i kompleksna mapiranja obave neekspertski korisnici
  - Omogućava analize uticaja i razvoj relacione šeme
- Bidirekcioni pogledi
  - Automatsko prevođenje iz mapiranja
  - Uniformno i efikasno tokom izvršavanja
  - Pojednostavljen razvoj & test (vs. case-by-case)
- Moguće je ažuriranje preko uskladištenih procedura

19.54

6

TIPOVI NASLEĐIVANJA

- Entity Framework podržava tri različita tipa nasleđivanja.

1. THP - *engl. Table Per Hierarchy*
2. TPT - *engl. Table Per Type*
3. TPC - *engl. Table Per Concrete Class*

19.54

7

TABLE PER HIERARCHY - TPH

- Table-per-hierarchy nasleđivanje koristi jednu tabelu iz baze za održavanje podataka u više od jednog entity tipa u jednoj hijerarhiji nasleđivanja u EF. Drugim rečima, postoji više „entity sets“ u EF za jednu tabelu u bazi. U TPH, bazna tabela ili entitet može biti kreiran i ima nekoliko izvedenih entiteta (mora biti više od jednog) koji su izvedeni iz baznog.

19.54

8

- **Nasleđivanje koristeći TPH**

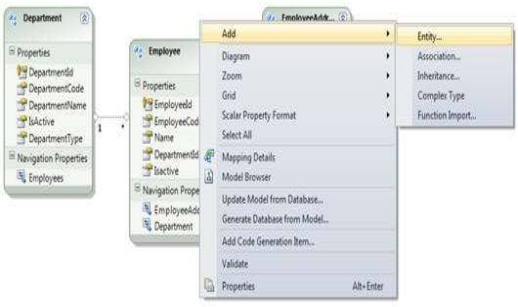
  Definisati entitet u konceptualnom modelu koji predstavlja bazni entitet i druge koji su izvedeni tipovi iz baznog.
- Izvedeni tip entiteta mora se naslediti od baznog tipa postavljanjem „Base Type“ atribut.
- U konceptualnom modelu se definišu (nenasledna) svojstva za izvedene tipove.
- *Diskriminator kolona* (koji deluje kao separator za izvedene entitete) koristi se tokom definisanja Mapiranja modela. Ako *Diskriminator kolona* sadrži više vrednosti kao broj, onda kolona mora biti *nullable* ili *diskriminator* ima neku vrednost koja osigurava da kada se kreira novi tip kolona ima obavezno neke vrednosti. Uslovi moraju da se koriste za mapiranje za svaki izvedeni tip i bazni tip u hijerarhiji. Nema mapiranje ukoliko je bazni tip apstraktna klasa. Bazni tip entiteta i izvedeni entitet su mapirani u "*EntitiSetMapping*".
- *IsTipeOf* se koristi za podešavanje vrednosti *Type Name*. Koristite stanje mapiranje za razlike između tipovai.

19.54

9

- **PRIMER: Kreiranje TPH nasleđivanja**

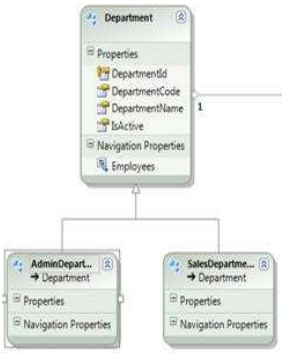
1 : Kreiranje Entity Model-a iz baze.
2 : Dodati novi Entity (desni klik na Entity designer Add - zatim Entity).



19.54

10

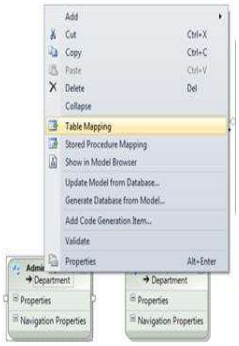
- **3 :** Napraviti izvedeni entitet, npr. "*AdminDepartment*" i "*SalesDepartment*".

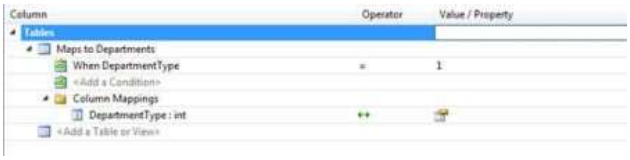


19.54

11

- **4 :** Izbacite *diskriminantnu kolonu* iz baznog tipa tabele "Department".
  5 : Definisati *Table mapping* – desni klik na izvedeni tip a zatim odaberite "*Table Mapping*".





19.54

12

- **6 :** Mapirajte tabelu i uslov.
- **7 :** Napravite bazni entitet apstraktnim.

19.54

13

- **Table per Type**

  Table-per-type nasleđivanje koristi jednu posebnu tabelu u bazi za održavanje podataka i koristi jedan entity tip u EF. Drugim rečima postoji jedan „entity set“ u EF za više tabela u bazi.

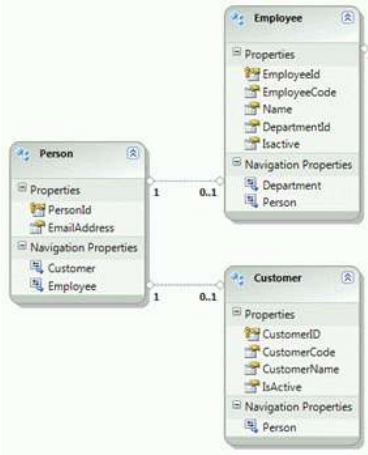
  Osnovna prednost *Table-per-type* je što se dobija normalizovana SQL šema po želji programera.

19.54

14

NASLEĐIVANJE KORISTEĆI TPT

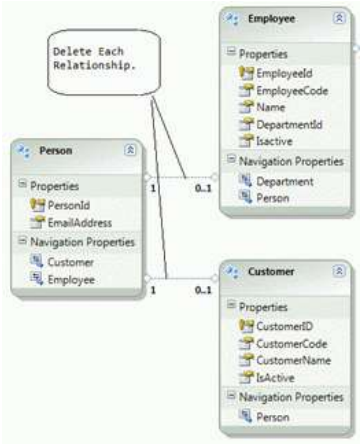
1. Kreirajte Entity Model na osnovu baze. Na primer:



19.54

15

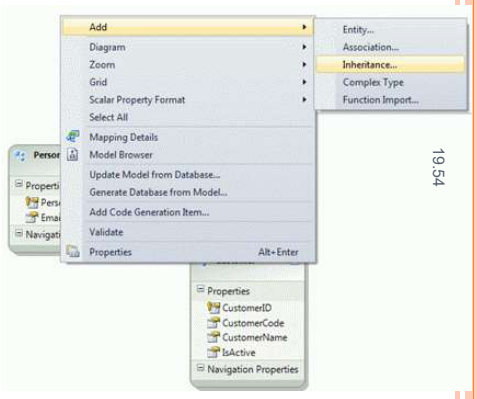
2: Obrišite Entity relacije.



19.54

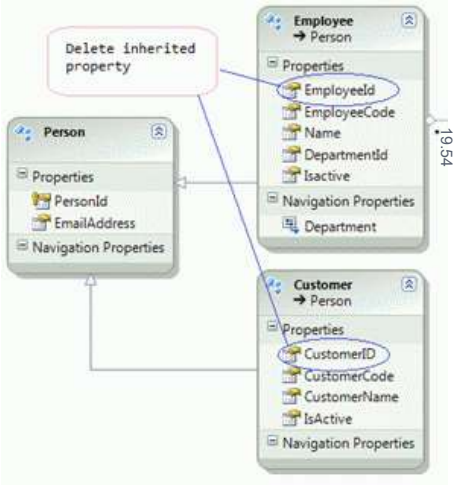
16

3: Dodavanje nasleđivanja



17

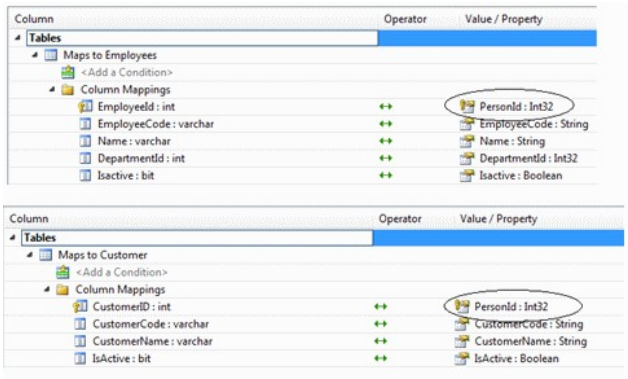
4: Izbrišite izvedena svojstva koja su ključevi



Obrišite ključeve izvedenih entiteta (Customer, Employee) (u ovom slučaju CustomerID, EmployeeId respektivno). Ovi ključevi će biti mapirani u svojstvo PersonId u narednom koraku.

18

5: Mapirati Key svojstva izvedenih tipova na key svojstvo baznog tipa.

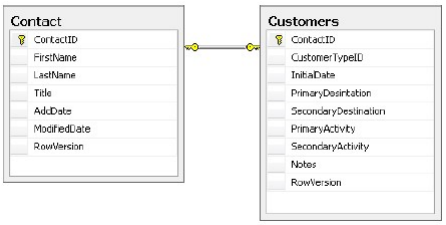


19.54

19

PRIMER 2.

- TPT nasleđivanje definiše nalеđivanje definisano u bazi sa odvojenim tabelama, pri čemu jedna tabela definiše sve podatke potrebne za definisanje tipa druge tabele.



19.54

Na slici se vidi 1:0..1 (Jedan na Nula ili Jedan) relacija između *Contact* i *Customer*. Ovo znači da jedan *Contact* može imati jedan *Customer* entitet u relaciji, ali to nije neophodno. Takođe znači da *Contact* ne može imati više od jednog *Customer* entiteta u relaciji. *Customer* tabela daje dodatne informacije u vezi podskupa kontakata.

20

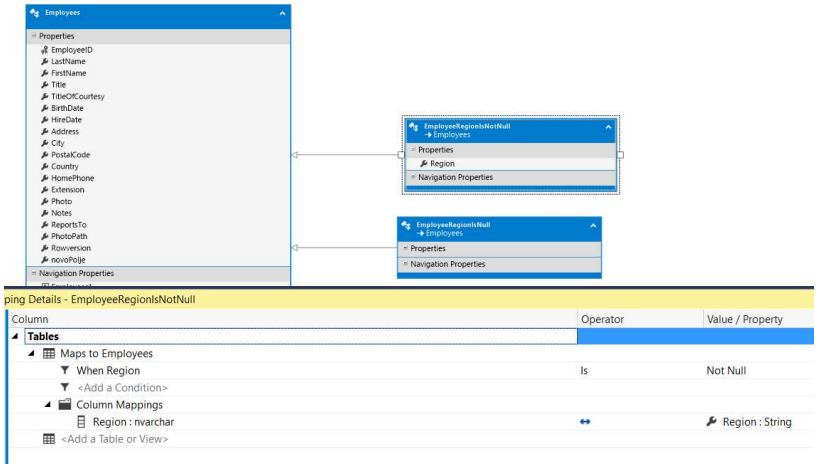
- Da bi zamenili navigaciju koju je *Entity Data Model Wizard* kreirao između *Contact* odnosno *Customer* sa nasleđivanjem koje mapira ka tabelama baze uraditi:
- 1. Izbrišite asocijaciju između *Contact* i *Customer* koju je *EDM Wizard* kreirao kada ste kreirali model.

  Dizajner omogućava 2 načina za dodavanje nasleđivanja. Možete selektovati jedan objekat za nasleđivanje iz *Toolbox*, kliknete na entitet koji treba da služi kao bazni, a zatim kliknete na entitet koji treba da bude izveden iz baznog. Alternativno, možete dodati iz konteksnog menija entiteta.
- 2. Desni klik na *Contact* entitet. Odaberite *Add* a zatim *Inheritance* iz konteksnog menija.
- 3. U prozoru *Add Inheritance*, odaberite *Contact* kao bazni a *Customer* kao izvedeni entitet. *Customer* će naslediti svojstva od *Contact*.
- 4. Obrišite *EntityKey* (*ContactID*) iz izvedenog (*Customer*). *Customer* će sada nasleđivati svojstvo *EntityKey* iz *Contact*.
- 5. Promenite svojstvo *ConcurrencyMode* polja *RowVersion'* kod Customer entiteta na *None*.
- 7. Otvorite *Mapping Details* prozor za *Customer*.
- 8. Mapirajte novi *Customer ContactID* svojstvo (koje sada dolazi od *Contact* entiteta) ka *ContactID* kolone u *Customers* tabeli.
- Kada je nasleđivanje postavljeno, entitet Customer će dobiti strelicu koja označava nasleđivanje. Strelica pokazuje na bazni entitet.

19.54

21

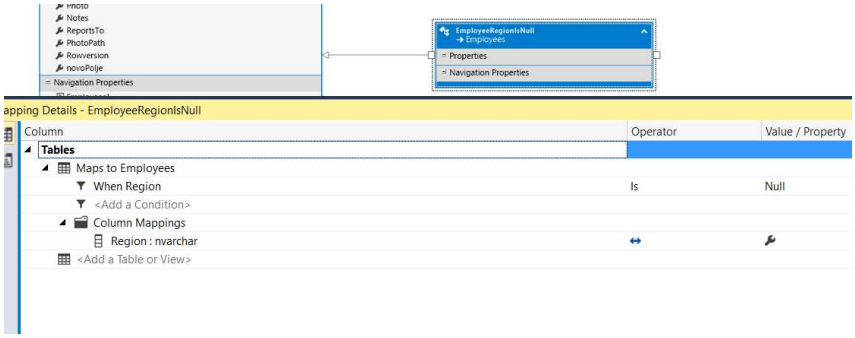
NORTHWIND PRIMER



19.54

22

19.54



23

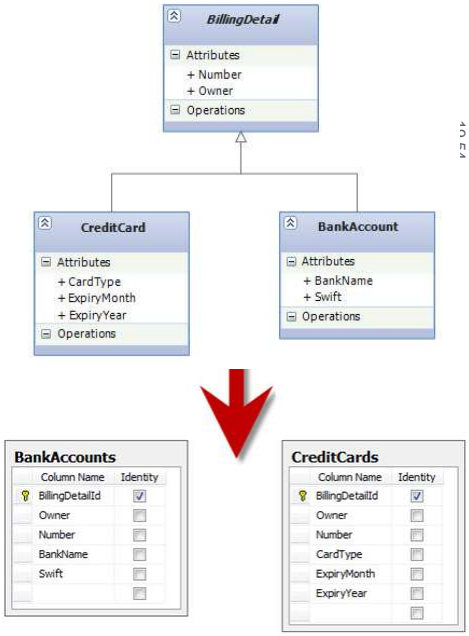
19.54

```
using (NorthwindEntities2 ctx = new NorthwindEntities2())
{
    var q = from e in ctx.Employees.OfType<EmployeeRegionIsNotNull>()
                // where
            select e;


    EmployeeRegionIsNotNull el = new EmployeeRegionIsNotNull();
    el.FirstName = "Dzek";
    el.LastName = "London";
    //el.Region = "Beogradski";
    ctx.Employees.Add(el);

    ctx.SaveChanges();
}
```

24

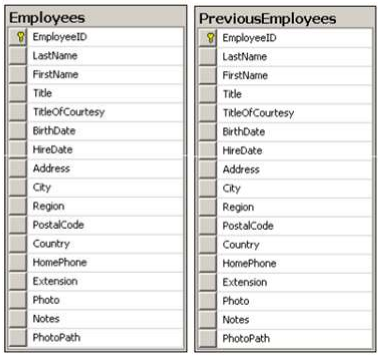
Table per Concrete type

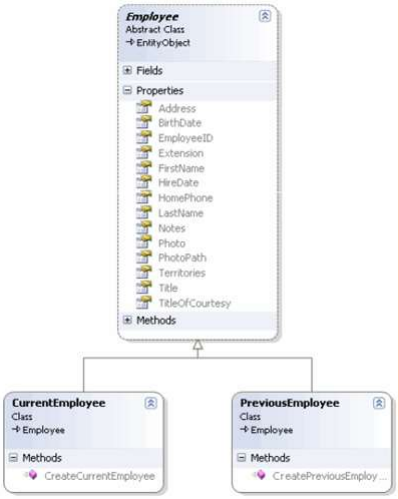
- Za ovaj tip nasleđivanja koristi se tačno jedna tabela za svaki neapstraktni tip tj klasu. Sva svojstva klase mogu biti mapirana u kolone tabele.



- Obično se izvodi kada u bazi postoje duplirana polja u više tabela. Na primer, ako postoje dve tabele *Employees* odnosno *PreviousEmployees* koje sadrže ista polja za dve grupe zaposlenih. Tada se izvodi nasleđivanje jednog tipa u drugi i to ručno.
- Da bi se kreiralo nasleđivanje, potrebno je da izbacite sva preklopljena svojstva iz izvedenog entiteta.

- Primer 2.





Employee je apstraktni zajednički tip za izvedene tipove.

27

19.54

28

EntityConnection

KONEKCIJSKI STRING U KODU

Konekcijski string se može kreirati koristeći *ConnectionStringBuilder*.
Naredni kod koristi dva objekta *ConnectionStringBuilders*: prvi za kreiranje SQL Server konekcijskog stringa a drugi za kreiranje EF konekcijskog stringa.

19.54

```
SqlConnectionStringBuilder sqlCnStrBuilder = new SqlConnectionStringBuilder();
sqlCnStrBuilder.DataSource = ".";
sqlCnStrBuilder.InitialCatalog = "Northwind";
sqlCnStrBuilder.IntegratedSecurity = true;
sqlCnStrBuilder.MultipleActiveResultSets = true;

EntityConnectionStringBuilder entityCnStrBuilder = new EntityConnectionStringBuilder();
entityCnStrBuilder.Provider = "System.Data.SqlClient";
entityCnStrBuilder.ProviderConnectionString = sqlCnStrBuilder.ConnectionString;
entityCnStrBuilder.Metadata =
@"res://*/Northwind.csdl|res://*/Northwind.ssdl|res://*/Northwind.msl";

Console.WriteLine(entityCnStrBuilder.ConnectionString);
```

KONEKCIJSKI STRING U *CONFIG* FAJLU

19.54

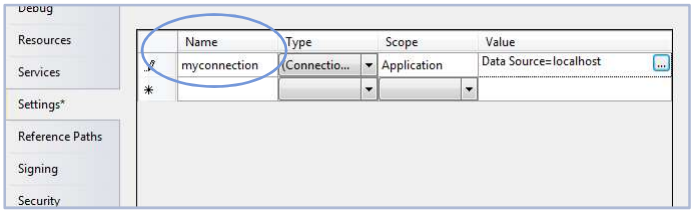
```
  ="utf-8"?>
<configuration>
  <connectionStrings>
    <add
      name="NorthwindEntities"
      connectionString="
                metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;
                provider=System.Data.SqlClient;
                provider connection string=&amp;quot;
                Data Source=localhost\sqlexpress;
                Initial Catalog=Northwind;
                Integrated Security=True;
                MultipleActiveResultSets=True&amp;quot;"
       providerName="System.Data.EntityClient" />
  </connectionStrings>
</configuration>
```

30

DODAVANJE U CONFIG FAJL

- Koristeći svojstva projekta u sekciji *Settings*

19.54




```
<connectionStrings>
  <add name="NorthwindEntities"
       connectionString="metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;provider=System.Data.SqlClient;
provid   er connection string=&quot;Data Source=localhost\sqlexpress;Initial Catalog=Northwind;Integrated
       Security=True;MultipleActiveResultSets=True&quot;"
           providerName="System.Data.EntityClient" />
  <add name="efPrimer1.Properties.Settings.myconnection" connectionString="Data Source=localhost\sqlexpress;Initial
       Catalog=Northwind;Integrated Security=True"
           providerName="System.Data.SqlClient" />
 </connectionStrings>
```

AUTOMATSKO OTVARANJE KONEKCIJA

- Konekcija se OTVARA odnosno ZATVARA automatski pri svakom izvršavanju upita

19.54

```
using (var ctx = new NorthwindEntities())
{
    foreach(var o in ctx.Orders)
    {
        foreach(var od in ctx.Order_Details){ }
    }
    var o1 = ctx.Orders.First();
}
```

otvaranje konekcije

Zatvaranje konekcije na poslednjem detalju

otvaranje konekcije

32

"RUČNO" OTVARANJE KONEKCIJE

- Kada se konekcija "ručno" otvori ona ostaje otvorena.
- Zatvaranje se obavlja
  - eksplicitnim zatvaranjem ili
  - pozivom *Dispose* metode ili
  - na kraju *using* bloka.

19.54

otvaranje konekcije

```
private void button1_Click(object sender, EventArgs e)
{
    using(var ctx = new NorthwindEntities())
    {
        ctx.Database.Connection.Open(); // eksplicitno otvaranje konekcije

        foreach(var o in ctx.Orders)
        {
            foreach(var od in ctx.Order_Details)
            {

            }
        }
        var o1 = ctx.Orders.First();
    } // zatvaranje konekcije
}
```

33

PODRAZUMEVANA TRANSAKCIJA

- Postoji ugrađena transakcija koja se kreira za izvršavanje metode **SaveChanges**. To znači, da će sve promene biti izvedene ili neće ni jedna!

19.54

```
Product[] products = GetProducts(ctx, 1, 2, 3);
products[0].CategoryID = 2;
products[1].CategoryID = 0;

try
{
    ctx.SaveChanges();
}
catch (UpdateException ex)
{
    ctx.Refresh(RefreshMode.StoreWins, products);
}
```

if exception takes place all changes are rolled back

34

```
using (var context = new BAEntities())
{
    var contact = context.Contacts.Where(c => c.ContactID == 5)
    .FirstOrDefault();
    context.DeleteObject(contact);
    var reservation = context.Reservations.FirstOrDefault;
    var payment = new Payment();
    payment.Amount = "500";
    payment.PaymentDate = System.DateTime.Now;
    payment.Reservation = reservation;
    context.SaveChanges();
}
```

Pokušaj brisanja jednog contact objekta iz baze neće uspeti zbog referecijalnog ograničenja. Zato će biti automatski pokrenut **roolback** za sve komande.

ObjectContext.AcceptAllChanges menja stanje svih entiteta koji su menjani. Postavljaju se *OriginalValues* šta god da je tekuća vrednost i menja *EntityState* na *Unchanged*.
U toku izvršenja *SaveChanges*, nakon kraja podrazumevane transakcije, metoda *AcceptAllChanges* se automatizovano poziva, na taj način uzrokujući da *ObjectContext* bude ažuran, a njegovi entiteti odgovaraju podacima u bazi.

19.54

35

SYSTEM.TRANSACTIONS

- EF koristi transakcije:
  - ako postoji, koristiće okolnu transakciju
  - kontroliše *timeout* trajanje, *isolation* nivo ...

```
TransactionOptions txOpts = new TransactionOptions();
txOpts.IsolationLevel = IsolationLevel.ReadUncommitted;
txOpts.Timeout = TimeSpan.FromMinutes(5);

using (var txScope = new TransactionScope
    (TransactionScopeOption.Required, txOpts))
{
   using (var ctx = new NorthwindEntities())
   {
      // Retrieve entities, update, save changes
```

19.54

36

OBJECTSTATEMANAGER

- *ObjectContext* sadrži svojstvo ***ObjectStateManager...***
  - ...koji sadrži ***ObjectStateEntry*** za praćenje stanja objekata.
  - ***ObjectStateEntry*** čuva originalnu i tekuću vrednost

19.54

```
void DumpChangeState(ObjectContext ctx, object entity)
{
   ObjectStateEntry entry = ctx.ObjectStateManager
      .GetObjectStateEntry(entity);
   Console.WriteLine(entry.State);
   for (int i=0; i<entry.OriginalValues.FieldCount; i++)
   {
      string propertyName =
         entry.OriginalValues.GetName(i);
      Console.WriteLine(propertyName);
      Console.WriteLine(entry.OriginalValues[i]);
      Console.WriteLine(entry.CurrentValues[i]); }
}
```

37

SAVECHANGES

- *ObjectContext* definiše *SaveChanges* kao *virtual*
  - Da bi dobili sopstvenu logiku, validaciju, logovanje i sl. potrebno je prekopiti ovu metodu
  - *SavingChanges* događaj ima sličnu namenu

19.54

```
public override int SaveChanges(SaveOptions options)
{
   foreach(var entry in ObjectStateManager
      .GetObjectStateEntries(EntryState.Modified)
   {
      Customer c = entry.Entity as Customer;
      if (c != null)
          c.Modified = DateTime.Now;
   }
   return base.SaveChanges(options);
}
```

mark entity as modified

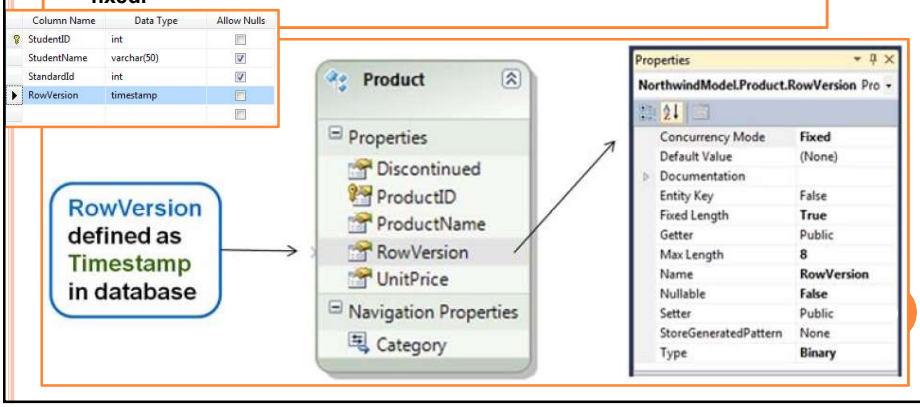
38

SVOJSTVO KONKURENTNOSTI

- Da bi se upravljalo sa eventualnim izuzecima u vezi konkurentsnosti, potrebno je da dodate kolonu RowVersion u tabelu. Potrebno je da odgovarajući tip podataka u SQL Serveru bude automatski generisan jedinstveni binarni broj kad god se promeni (Update) ili doda (Insert) zapis u tabeli. Ovaj broj je jednostavno broj koji se uvek uvećava, a može se koristiti tip *timestamp*.

  Zatim se formira EF model ili se postojeći ažurira.

  Nakon toga može da spostavi *concurrency mode* u EF modelu na vrednost **fixed.**



19.54

UPRAVLJANJE KONFLIKTIMA KONKURENTNOSTI

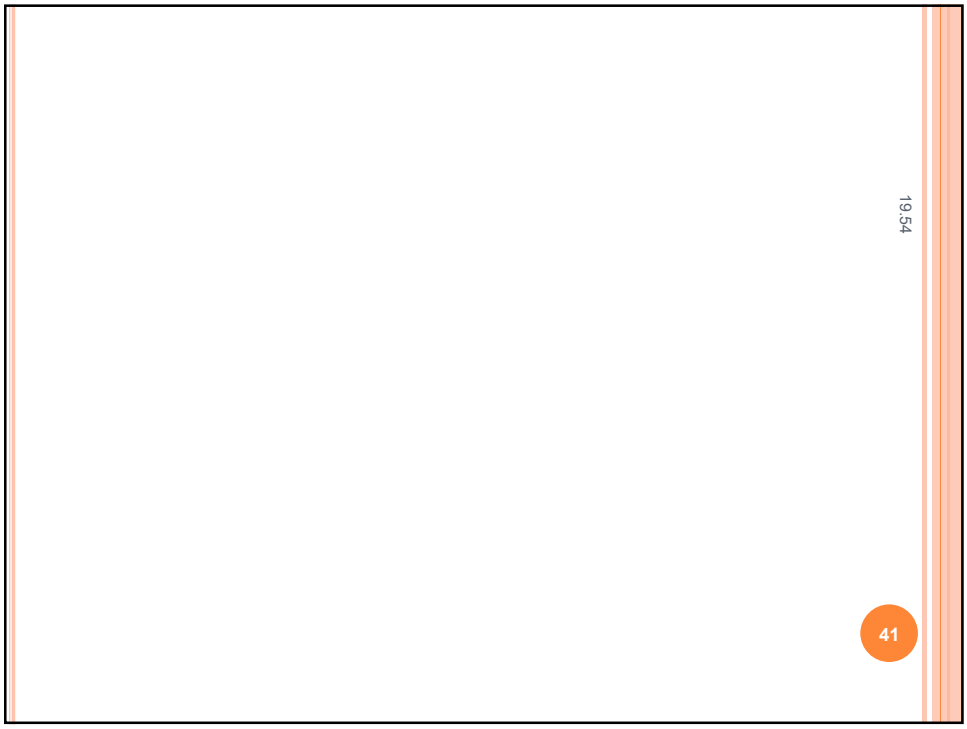
- EF koristi *OptimisticConcurrencyException* podrazumevano. Ukoliko EF ustanovi promenu podatka izbacuje se izuzetak.
  - konflikti sadrže svojstvo **StateEntries**
  - poziva se **Refresh** sa *StoreWins* ili *ClientWins*

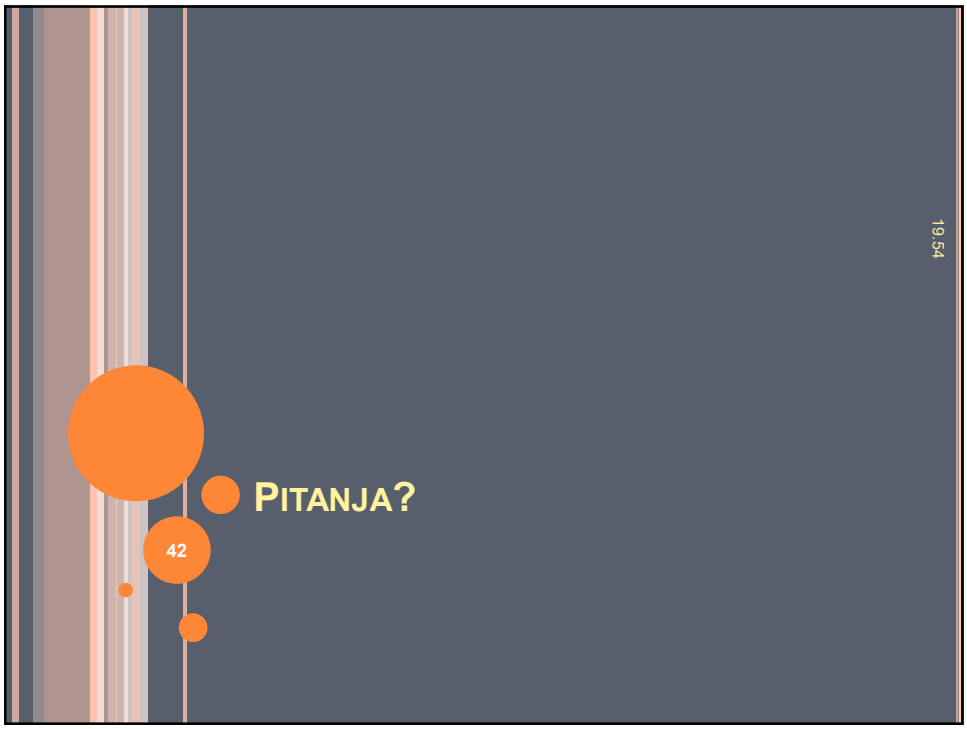
ClientWins uses PreserveChanges MergeOption

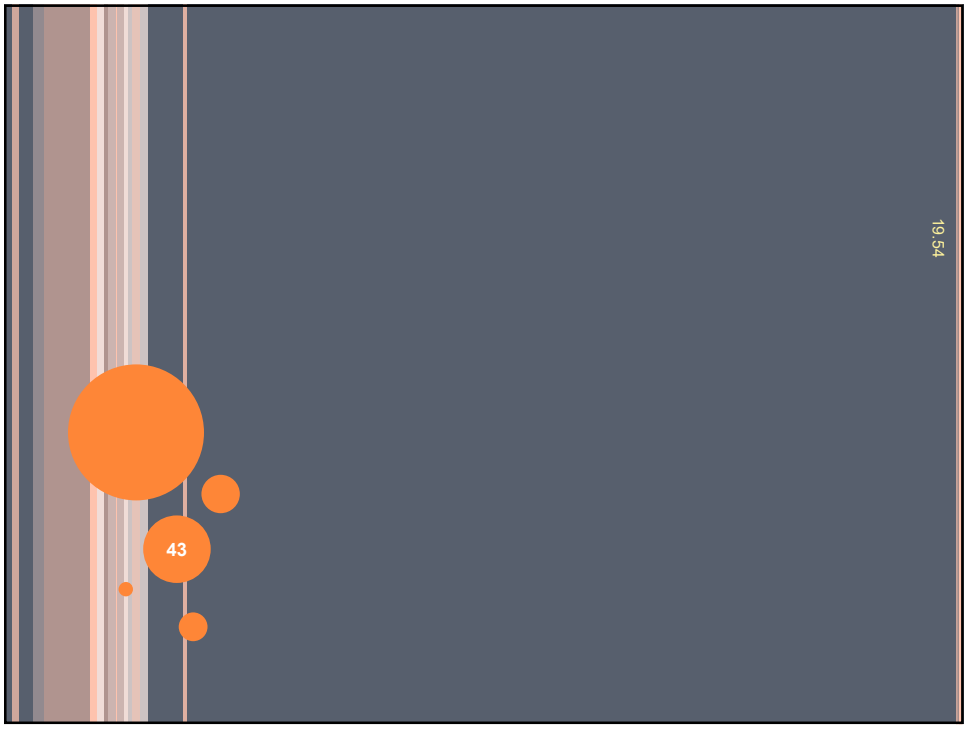
```
try
{  bool acceptDefeat = false;
   ctx.SaveChanges();
}
catch (OptimisticConcurrencyException conflictEx)
{
   foreach(var entry in conflictEx.StateEntries) {
     if (acceptDefeat)
       ctx.Refresh(RefreshMode.StoreWins, entry.Entity);
     ctx.Refresh(RefreshMode.ClientWins, entry.Entity); }
     ctx.SaveChanges();
}
```

19.54

40

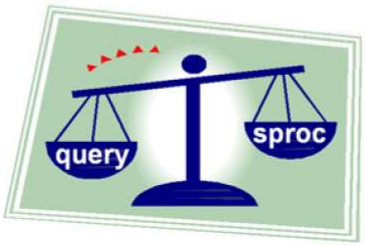

19.54
41


19.54
PITANJA?
42



Stored procedures [justification]

- **When to use stored procedures ?**
  - security (table access is locked down)
  - optimization (avoid query translation step)
- **When to NOT use stored procedures?**
  - to encapsulate business logic
  - query execution speed
    - not much faster than dynamic SQL on most DBs

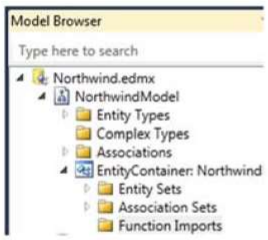


19.54

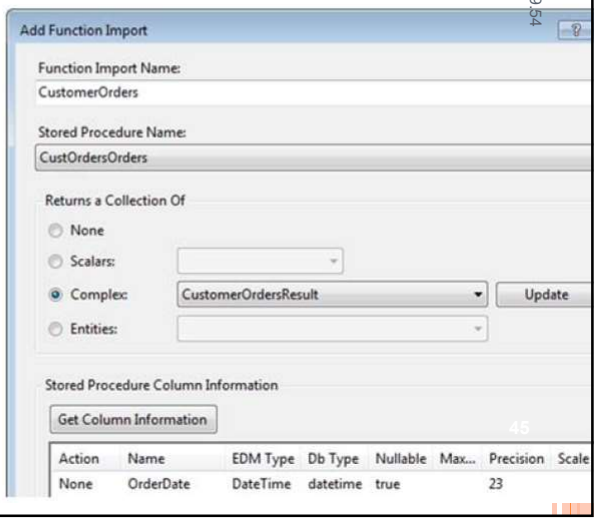
Stored procedures [queries]

- **better designer support in EF4 for stored procedures**
  - map results to scalars, entities or complex types

19.54



1. add sproc to model
2. add function import
3. invoke method on object context



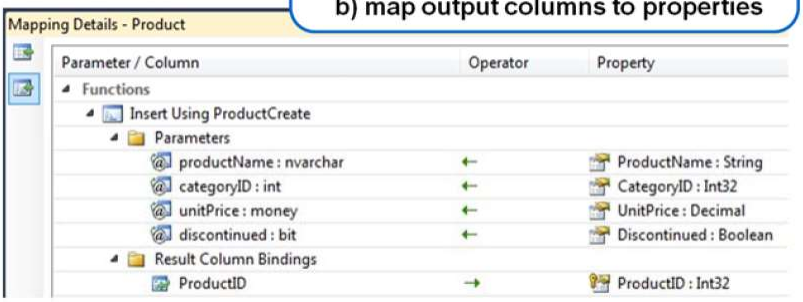
45

Stored procedures [updates]

- **Story for CUD sprocs vastly improved in EF4**
  - can supply sprocs for some, but not all, operations*
  - no need for dummy foreign key parameter on delete sproc

19.55

1. add CUD sprocs to model
2. add entity function mappings
   a) map properties to input parameters
   b) map output columns to properties



19.53

* Although the model will validate if not all CUD (Create, Update, Delete) operation
map stored procedures for those operations you wish to invoke. For example, you c
procedure if all you are doing is updating an entity. But you will need to map create
well if you wish to save inserts and deletes on the entity. EF will not perform update
or more operations are mapped to stored procedures.

Note that CUD sprocs will automatically be wrapped in a transaction when you call

47

Stored procedures [concurrency token]

- Sprocs need to include concurrency token in where clause
  - set output parameter to rows affected
  - return concurrency token so next update will succeed

19.54

```
CREATE PROCEDURE [dbo].[ProductUpdate]
   (@productId int, @productName nvarchar(40),
   @rowversion timestamp, @updateCount int OUTPUT)
AS
   UPDATE Products
   SET ProductName = @productName
   WHERE ProductID = @productId
   AND RowVersion = @rowversion

   SELECT RowVersion FROM Products
   WHERE ProductID = @productId AND @@ROWCOUNT > 0

   SELECT @updateCount = @@ROWCOUNT
```

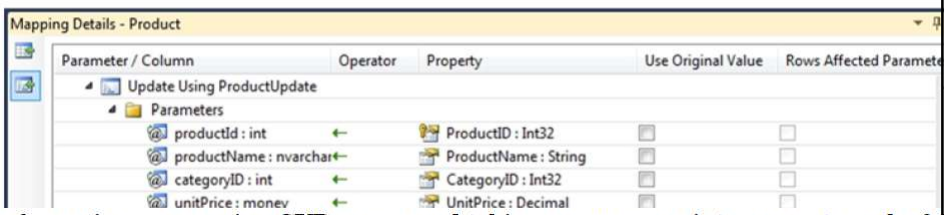
rows affected
zero if updated
by another user

48

Stored procedures [concurrency mapping]

- **Select output parameter as "Rows Affected Parameter"**
  - OptimisticConcurrencyException will occur if no rows affected
  - also check "Use Original Value" for concurrency token

19.54



information on mapping CUD sprocs and taking concurrency into account see the fo

lification Functions to Stored Procedures: http://msdn.microsoft.com/en-/cc716711%28VS.100%29.aspx

ectedParameter : http://msdn.microsoft.com/en-us/library/dd560889%28VS.100%29.

EF throws OptimisticConcurrencyException if Rows Affected Parameter returns zero

Summary

- **Use System.Transactions to control timeout, isolation level**
  - but be careful about promotion to a distributed tx
  - manually open connection if using SQL Server 2005 or earlier
- **Override SaveChanges to replace or modify persistence**
  - insert custom logic, validation, logging, etc.
- **Use timestamp column for concurrency management**
  - set ConcurrencyMode to Fixed
  - use Refresh (ClientWins, StoreWins) to resolve conflicts
- **Use stored procedures for security or political reasons**
  - use designer to map results to complex types
  - more work to manage concurrency with spocs

19.54