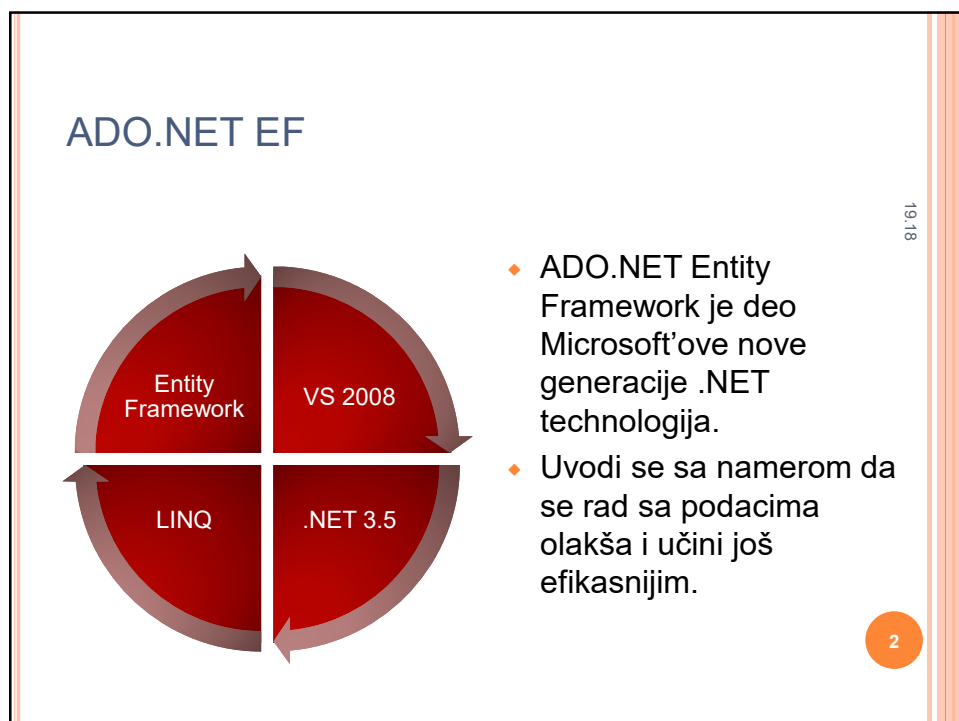
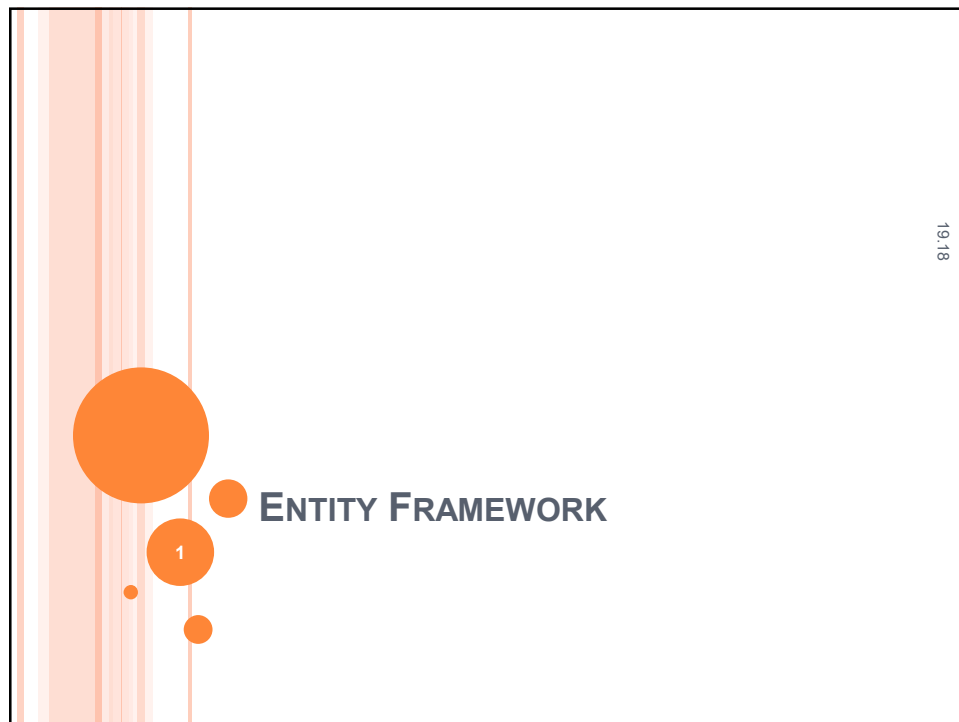




ŠTA DONOSI EF?

Data modeling

Konceptualni modeli *Conceptual Models* – Odvajanje apl. od stvarnih šema podataka

Better Integration

Integracija sa podacima kao objektima ili sa kolonama i redovima

Programming languages

LINQ podrška u celom ADO.NET^{3u.}

EF | ADO.NET

4

19.18

DALJI RAZVOJ ADO.NET - A

- Gotovo sve aplikacije su orijentisane ka podacima - *data-centric app*.
- Razvoj ADO.NET je neophodan, ali je neophodno i čuvanje mogućnosti prethodnih verzija. To se postiže:
 - Proširenjem funkcionalnosti na postojećem steku biblioteka, kada je moguće
 - Izgradnjom novih slojeva na vrhu steka



KLIJENTSKI POGLEDI KOD ADO.NET-A

- Svaka aplikacija ima odgovarajuće poglede na podatke.
- Pogledi na podatke se u potpunosti implementiraju na strani klijenta.
 - Treba izbegavati nepotrebno dodavanje pogleda na stani baze zbog pogleda potrebnih za aplikaciju.
 - Razvoj aplikacija ne bi trebalo vezivati za razvoj baze.
- Velike mogućnosti
 - Veoma širok set pogleda koji se mogu koristiti pri ažuriranju
 - Ažuriranje se radi preko tabela, relacija...

19:18

MODEL PODATAKA - TRADICIONALNI

- Svaka aplikacija ima, eksplicitno ili implicitno definisan neki konceptualni model podataka.
- Da li je *Db Schema* idealna?

SalesOrder	
PK	<u>SalesOrderID</u>
FK1	SalesPersonID OrderDate Status OnlineOrderFlag AccountNumber TaxAmt TotalDue

Contact	
PK	<u>ContactID</u>
	FirstName MiddleName LastName EmailAddress Phone

Employee	
PK,FK1	<u>EmployeeID</u>
	LoginID ManagerID Title BirthDate HireDate SalariedFlag VacationHours SickLeaveHours CurrentFlag

SalesPerson	
PK,FK1	<u>SalesPersonID</u>
	Bonus SalesLastYear SalesQuota SalesYTD

7

TRADICIONALNI UPIT - 1

- Primer:
- Napisati upit za zaposlene koji su zaposleni tokom 2006 i prikazati imena i titule.

Contact	
PK	<u>ContactID</u>
	FirstName MiddleName LastName EmailAddress Phone

Employee	
PK,FK1	<u>EmployeeID</u>
	LoginID ManagerID Title BirthDate HireDate SalariedFlag VacationHours SickLeaveHours CurrentFlag

8

TRADICIONALNI UPIT - 2

```
SELECT c.FirstName, e.Title
FROM Employee e
INNER JOIN Contact c ON e.EmployeeID = c.ContactID
WHERE e.HireDate >= '2006-01-01'
AND e.SalariedFlag = 1
```

Kreiranje
"odgovarajućeg
pogleda" koristeći
JOIN

Pogled na samo
jedan eksplicitan
podskup podataka

9

TRADICIONALNI UPIT - 3

- Primer:
- Dobijanje svih prodavaca koji imaju narudžbine vrednije od \$200,000:

Contact		Employee		SalesPerson	
PK	<u>ContactID</u>	PK,FK1	<u>EmployeeID</u>	PK,FK1	<u>SalesPersonID</u>
	FirstName		LoginID		Bonus
	MiddleName		ManagerID		SalesLastYear
	LastName		Title		SalesQuota
	EmailAddress		BirthDate		SalesYTD
	Phone		HireDate		
			SalariedFlag		
			VacationHours		
			SickLeaveHours		
			CurrentFlag		

10

TRADICIONALNI UPIT - 4

```

SELECT SalesPersonID, FirstName, LastName,
FROM SalesPerson sp
INNER JOIN Employee e
    ON sp.SalesPersonID = e.EmployeeID
INNER JOIN Contact c
    ON e.EmployeeID = c.ContactID
INNER JOIN SalesOrder o
    ON sp.SalesPersonID = o.SalesPersonID
WHERE o.TotalDue > 20000
AND e.SalariedFlag =

```

Šta je sp?

Komplikovano!

Šema je izdeljena

Teško pratiti relacije

19:18

11

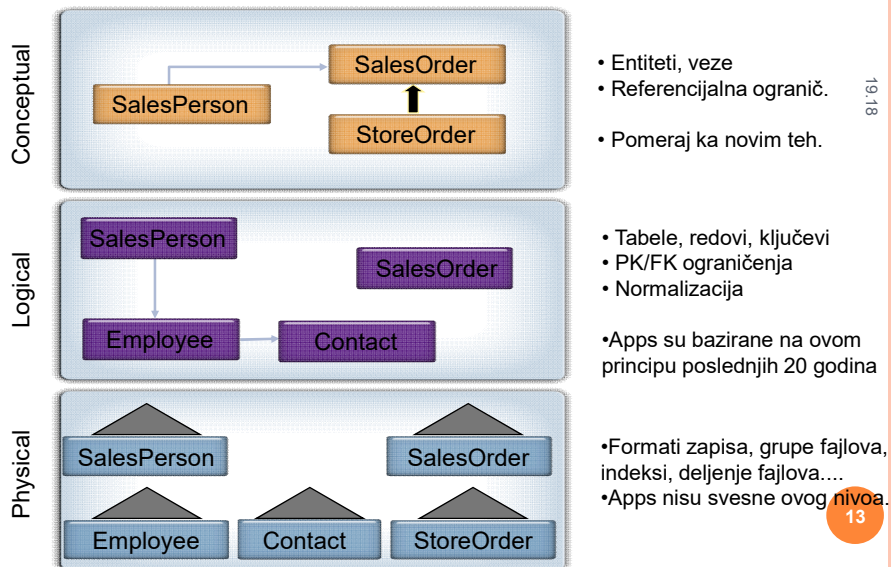
ZAKLJUČAK, U REALNIM SLUČAJEVIMA...

- ...db šeme nisu uvek idealne. Nedostaci
 - Treba ispuniti operacione zahteve ili zahteve za boljim performansama aplikacija.
 - Nove aplikacije i novi zahtevi nakon već kreiranih šema
- ...šeme se provlače i kroz kod
 - Apl kreiraju poglede na podatke koji su od značaja
 - U kodu postoji implicitno “znanje” o podacima

19:18

12

PREMEŠTANJE NA KONCEPTUALNI MODEL



KAKO SE MODEL ZAPISUJE

- Kao metapodaci u XML formatu
- 3 fajla
 - .CSDL (Conceptual Schema Definition Language)
 - Mapira tipove entiteta koji se koriste u konceptualnom modelu
 - .SSDL (Store Schema Definition Language)
 - Šema podataka za bazu
 - .MSL (Mapping Schema Language)
 - Mapira konceptulnu šemu u šemu podataka baze
- Kreira se koristeći alatke pri dizajnu ili
 - EMDGen
- Mora biti na raspolaganju u toku izvršavanja programa
 - Kopira se u bin folder, ili se koristi ugrađivanje resursta tokom build opcije

19:18

14

LOGIČKI MODEL VS. OBJEKTNO ORIJENTISANI MODEL

Tradicionalni, zasnovan na modelu podataka u bazi

Tables

Views

Stored Procedures

Foreign Key Relationships

- Tipičan, masovan, dobro poznat i primenljiv.
- Ovo znači i brigu oko stanog ključa, pogleda i procedure.

19.18

15

OBJEKTNO-ORIJENTISANI MODEL

Objektni model koji je prisutan u aplikacijama.

Objects

Behavior

Properties

Inheritance

Complex Types

- Aplikacije zapisuju i predstavljaju podatke na drugačiji način.
- Isti podaci koji su u relacionim bazama su najčešće potpuno drugačije predstavljeni u aplikacijama.

19.18

16

REZULTAT

Aplikacije koje rade sa modelima podataka u relacionoj bazi...

Logical Data Model

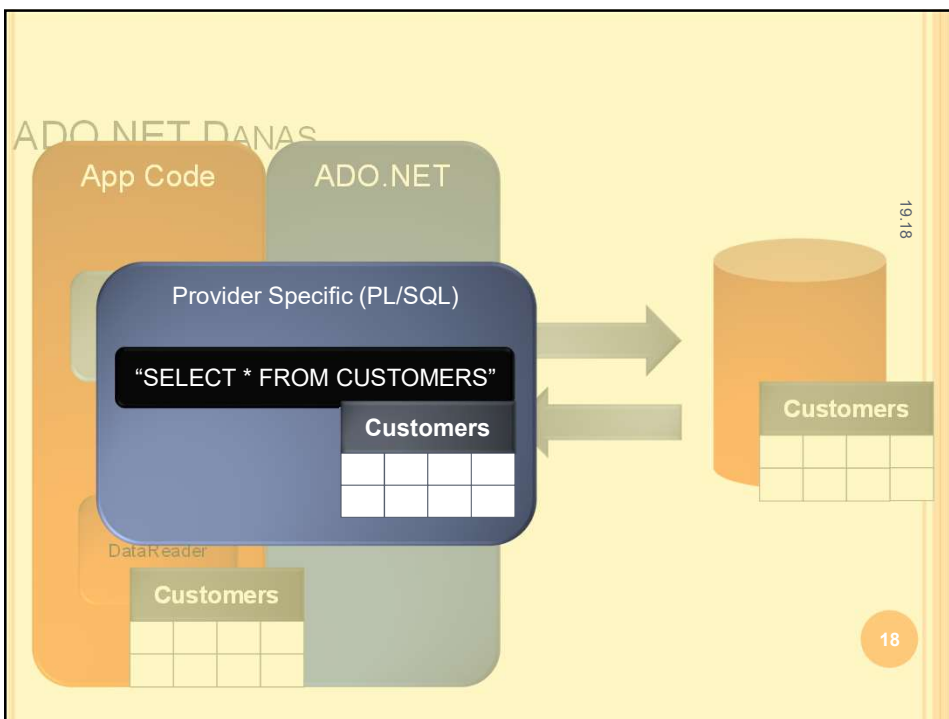
Lots of Custom Code

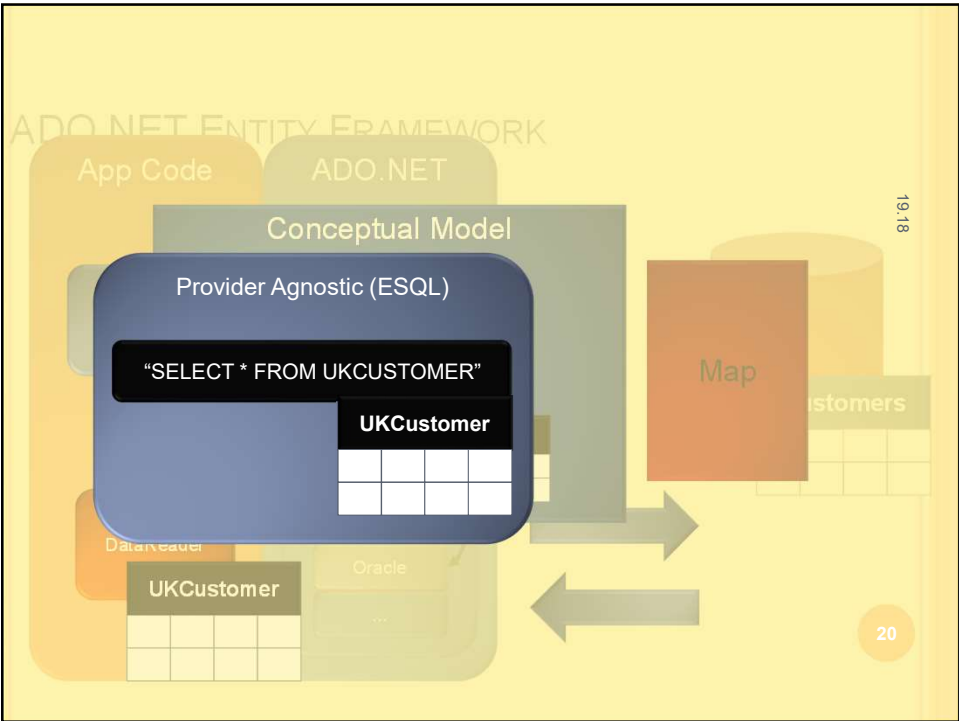
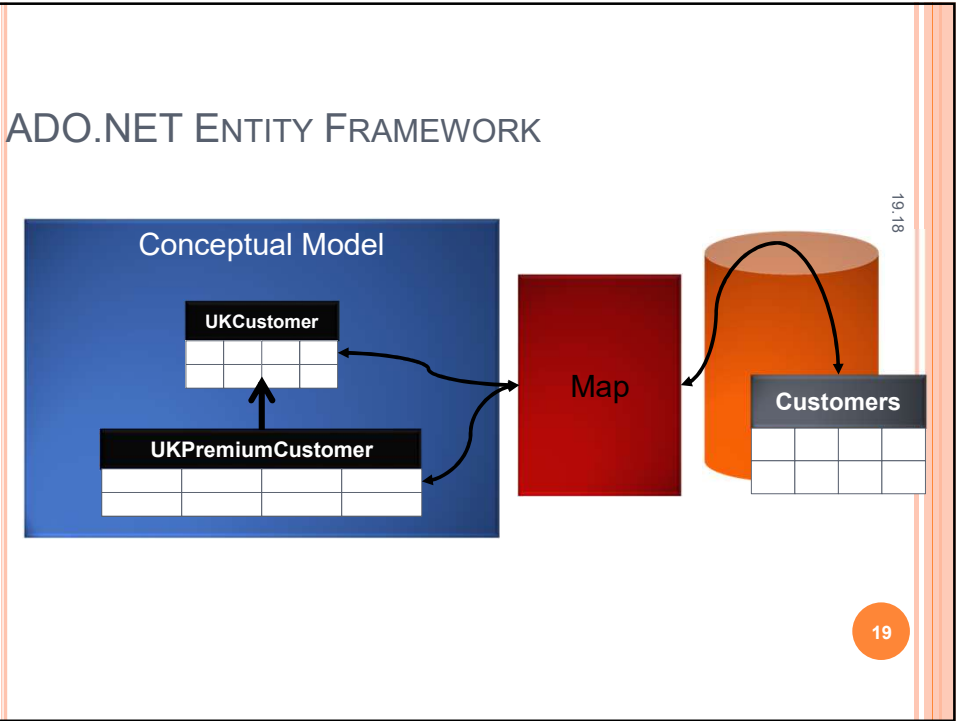
Application Domain

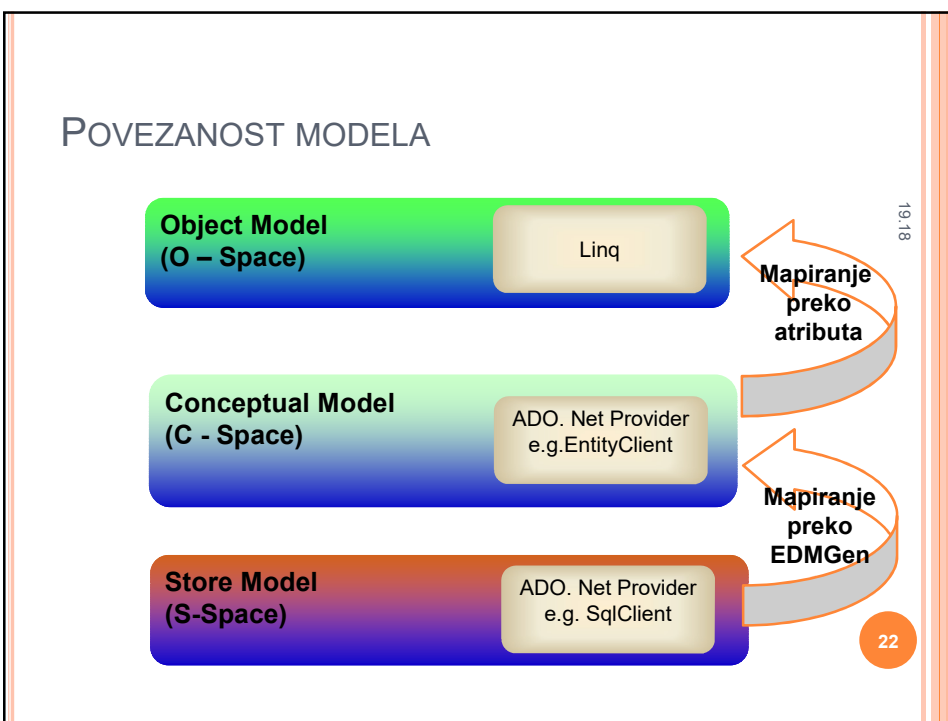
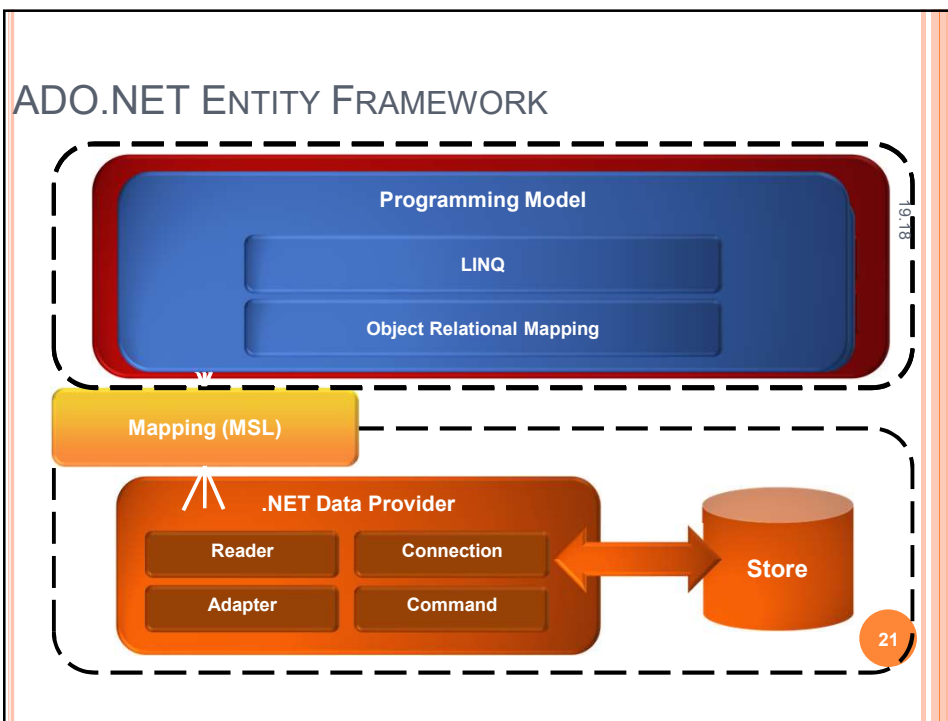
- ♦ Rezultat “nezavisnog prilagođavanja” je da programeri troše puno vremena i energije za pisanje koda koji prevodi podatke koji odgovaraju aplikaciji u one koji odgovaraju bazi i obrnuto.

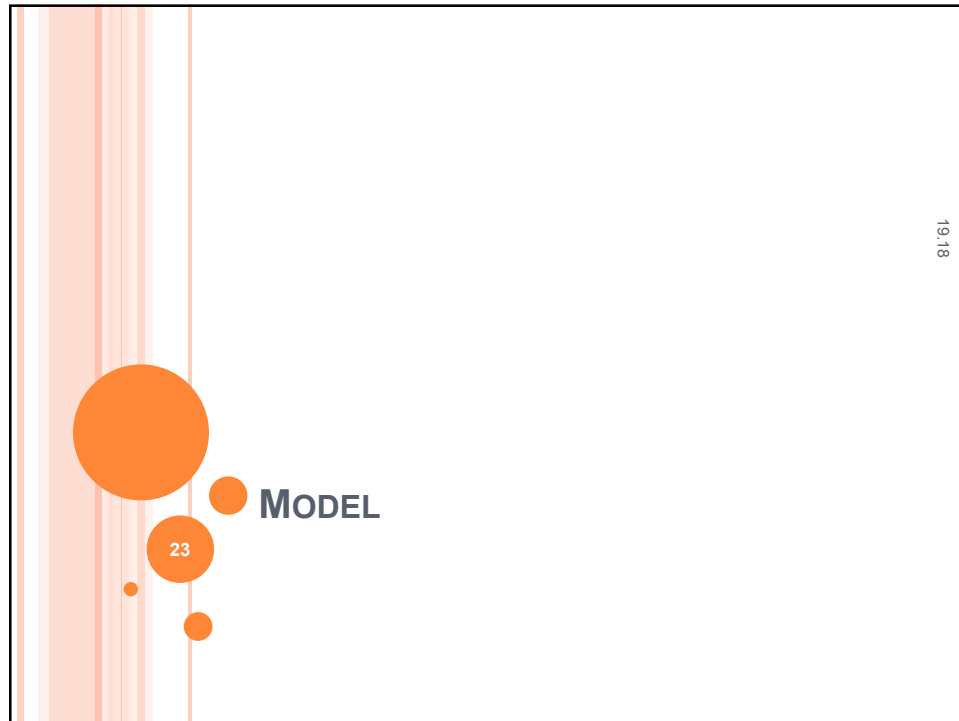
19:18

17







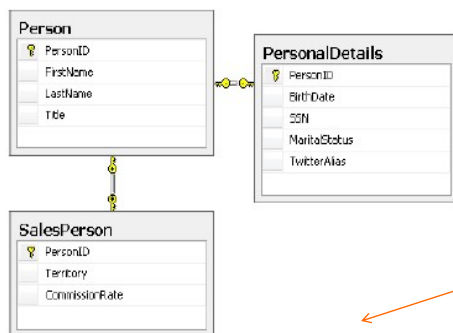


ENTITY DATA MODEL (EDM)

- Predstavlja model sastavljen od elemenata *entitet-relacije*
- Glavni koncepti:
 - ❖ *Tip Entiteta* je strukturirani zapis sa ključem
 - ❖ *Entitet* je jedna instanca tipa entiteta
 - ❖ Entiteti su grupisani u skupove entiteta: *Entity-Sets*
 - ❖ Tip jednog entiteta može naslediti drugi tip

EDM

- EDM je model podataka i predstavlja jezgro EF-a. NIJE ISTI KAO MODEL BAZE.
- EDM je model koji je prilagođen strukturi poslovnih objekata.

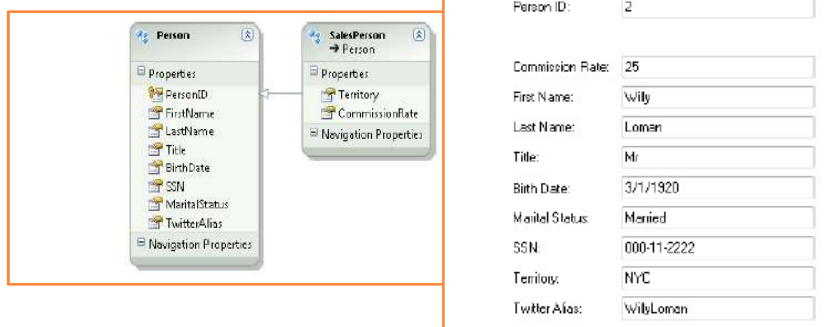


Na primer:

```
SELECT SalesPerson.*, PersonalDetails.*, Person.*
FROM Person
INNER JOIN PersonalDetails
ON Person.PersonID = PersonalDetails.PersonID
INNER JOIN SalesPerson ON Person.PersonID = SalesPerson.PersonID
```

EDM (2)

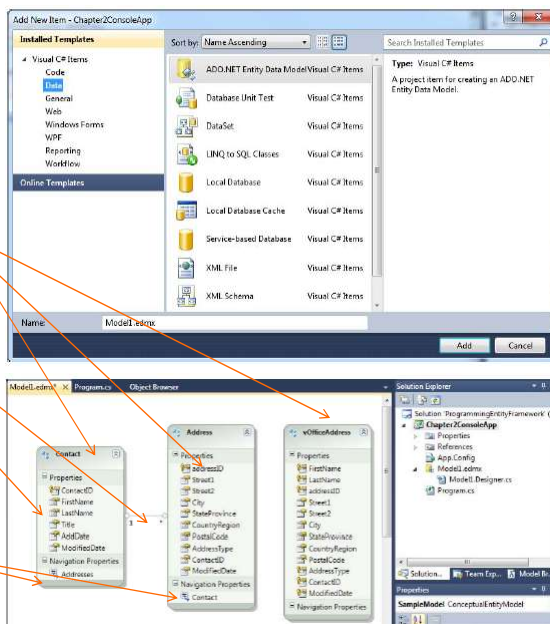
- Ako aplikacija treba da ostvari vezu i da je šema pri tome:



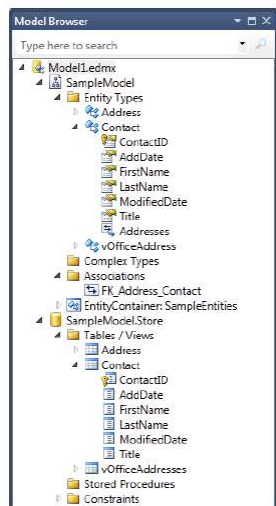
from p in People.OfType<SalesPerson> select p

KREIRANJE MODELA

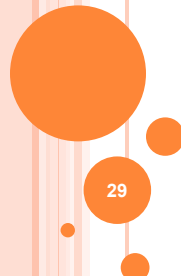
- *Contact* i *Address* potiču od tabela, a treći entitet od pogleda.
- Postoji i linija koja označava vezu između *Contact* i *Address* i to jedan na više. Svaki entitet ima izvestan broj skalarnih svojstava, a takodje i entitete sa kojima je u relaciji preko dodatnih navigacionih svojstava.



MODEL BROWSER



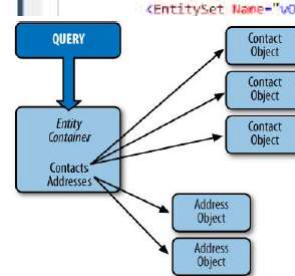
19.18



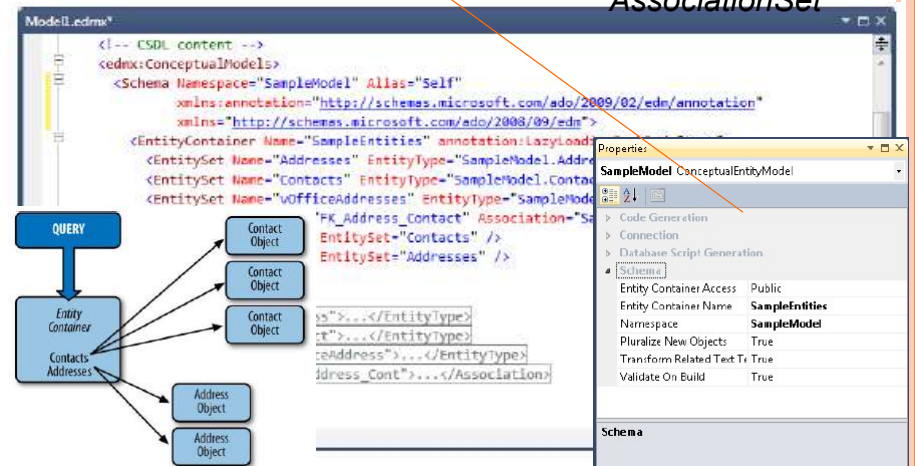
DETALJI VEZANI ZA CELINE EDM MODELA

29 **Konceptualni model**
Model skladišta podataka
Model mapiranja

KONCEPTUALNA ŠEMA



○ **EntityContainer** – predstavlja omotač za **EntitySet** i **AssociationSet**



The screenshot shows the Visual Studio EDMX editor with the following CSDL content:

```

<!-- CSDL content -->
<edmx:ConceptualModels>
  <Schema Namespace="SampleModel" Alias="Self"
    xmlns:annotation="http://schemas.microsoft.com/ado/2009/02/edm/annotation"
    xmlns="http://schemas.microsoft.com/ado/2008/09/edm">
    <EntityContainer Name="SampleEntities" annotation:lazyload="true">
      <EntitySet Name="Addresses" EntityType="SampleModel.Address" />
      <EntitySet Name="Contacts" EntityType="SampleModel.Contact" />
      <EntitySet Name="wOfficeAddresses" EntityType="SampleModel.Contact" />
      <AssociationSet Name="FK_Address_Contact" Association="SampleModel.Address_Contact"
        EntitySet="Contacts" EntitySet="Addresses" />
    </EntityContainer>
  </Schema>
</edmx:ConceptualModels>
  
```

The Properties window on the right shows the following details for the SampleModel:

Property	Value
Code Generation	True
Connection	True
Database Script Generation	True
Entity Container Access	Public
Entity Container Name	SampleEntities
Namespace	SampleModel
Pluralize New Objects	True
Transform Related Text To	True
Validate On Build	True

ENTITYSET

- Predstavlja kontejner za tipove jednog entiteta. Sadrži dva polja: *Name* i *EntityType*.
- Definiše koji entitet je sadržan u skupu koristeći strogo tipizirana imena. Na primer:


```
<EntitySet Name="Addresses"
  EntityType="SampleModel.Address" />
<EntitySet Name="Contacts"
  EntityType="SampleModel.Contact" />
```
- Koristeći ovaj skup, pristupa se pojedinačnim objektima putem raznih upita.

ENTITYTYPE

- *EntityType* je tip podataka u modelu. U prethodnom primeru postoji *Contact* i *Address*.
- Kada se u XML šemi proširi tip *Address* dobija se XML koji je prikazan. Prepoznaju se:
 - ključ "Key" i
 - svojstva tj. *Property* elemenata.

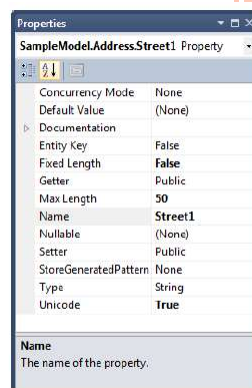
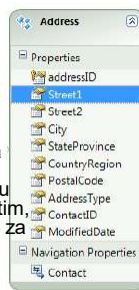
```
<EntityType Name="Address">
  <Key>
    <PropertyRef Name="addressID" />
  </Key>
  <Property Name="addressID" Type="Int32" Nullable="false"
    annotation:StoreGeneratedPattern="Identity" />
  <Property Name="Street1" Type="String" MaxLength="50"
    Unicode="true" FixedLength="true" />
  <Property Name="Street2" Type="String" MaxLength="50"
    Unicode="true" FixedLength="true" />
  <Property Name="City" Type="String" MaxLength="50"
    Unicode="true" FixedLength="true" />
  <Property Name="StateProvince" Type="String" MaxLength="50"
    Unicode="true" FixedLength="true" />
  <Property Name="CountryRegion" Type="String" MaxLength="50"
    Unicode="true" FixedLength="true" />
  <Property Name="PostalCode" Type="String" MaxLength="20"
    Unicode="true" FixedLength="true" />
  <Property Name="AddressType" Type="String" Nullable="false"
    MaxLength="10" Unicode="true" FixedLength="true" />
  <Property Name="ContactID" Type="Int32" Nullable="false" />
  <Property Name="ModifiedDate" Type="DateTime" Nullable="false" />
  <NavigationProperty Name="Contact"
    Relationship="SampleModel.FK_Address_Contact"
    FromRole="Address" ToRole="Contact" />
</EntityType>
```

Key element

- Definiše koja svojstva formiraju identitet tj. jedinstvenost entiteta.
- Može biti sačinjen od više svojstava.
- U Dizajneru i kodu predstavlja se sa *EntityKey*.

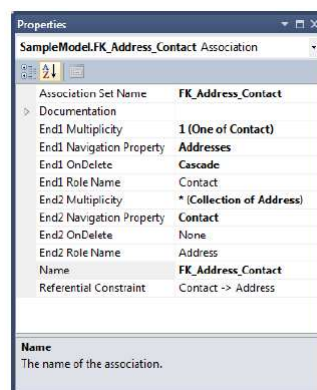
Property elementi

- Oni su definisani imenom ali i dodatnim tipom podataka. Tipovi podataka koji definišu ova svojstva nazivaju je prostim tipovima - *simple types*. Ovo su osnovni tipovi u Entity Framework objektnom modelu koji su najbliži tipovima u .NET Framework-u. Međutim, osnovni tipovi u Entity Framework's se koriste samo za definisanje svojstava entiteta.
- Osnovna svojstva ovih elemenata se vide i iz *Properties* prozora kao na slici
- Navigation properties**
- Navigaciona svojstva su čvrsto povezana sa asocijacijama koje predstavljaju linije između entiteta.

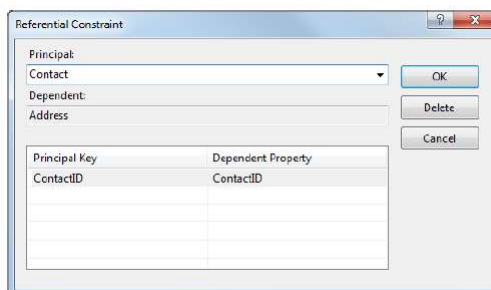


Asocijacije - Associations

- Definišu veze između entitetskih tipova. Asocijacija ne definiše relaciju u potpunosti. Međutim, definišu se krajnje tačke (entiteti koji su uključeni u relaciju) kao i njihova multiplikacija.
- U primeru modela, postoji samo jedna asocijacija, koja je između entitetskih tipova *Contact* i *Address*, ukazujući da postoji relacija između njih.
- Ime ove asocijacije je uzeto iz predefinisano imena relacije u bazi u fazi kada je "čarobnjak" formirao model. Kao za bilo koji drugi element u modelu, može se editovati ime.
- Properties window* za asocijaciju iz primera je prikazan na slici.

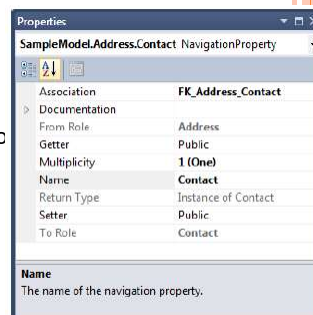


- Zapazimo svojstvo *Referential Constraint*.
- U modelu koji sadrži strane ključeve u entitetima, kao što je svojstvo *ContactID* iz *Address*, definišu se zavisnost između entiteta koji su u relaciji.
- Svaki objekat *Address* mora pokazivati na objekat *Contact*.



○ Navigation Property

- Navigaciona svojstva su deo entitetskih tipova *Address* i *Contact* (prethodno samo spomenuta).
- Kada se radi sa *Address* objektom u kodu, svojstvo *Contact navigation* će se pojavljivati baš kao i druga svojstva. Mada su druga svojstva označena kao skalarna, tj. ona su vrednosti, navigaciono svojstvo opisuje kako se dolazi do entiteta sa kojim postoji relacija.
- Važno svojstvo navigacije je *Association*. Ovo svojstvo ukazuje na asocijaciju u modelu koja sadrži informacije u vezi navigacije do entiteta u relaciji (*Contact.Addresses*).
- Atributi *FromRole* odnosno *ToRole* opisuju da Entity Framework, kada koristi asocijaciju, potrebna je navigacija od tačke *Address* do krajnje tačke *Contact*.



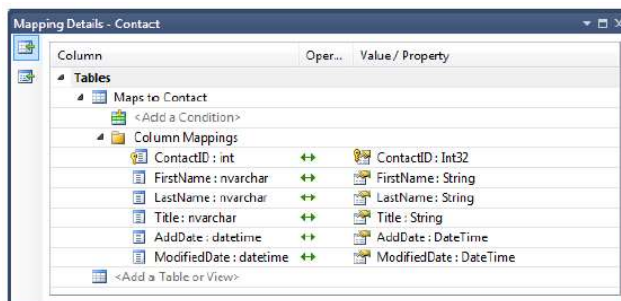
ŠEMA ZA SKLADIŠTENJE PODATAKA SSDL: THE STORE SCHEMA

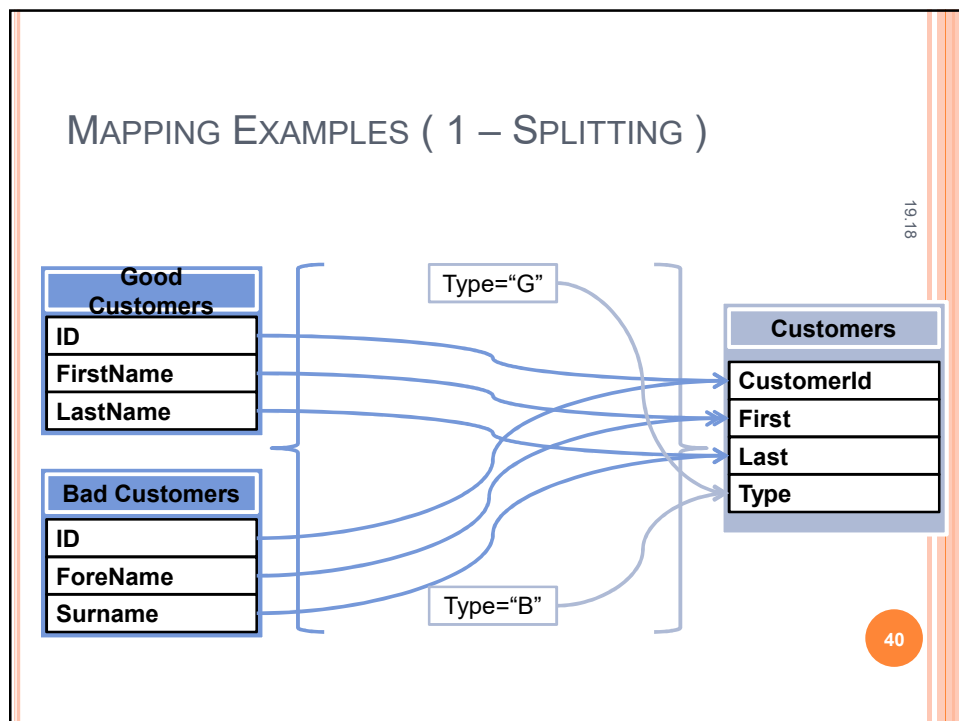
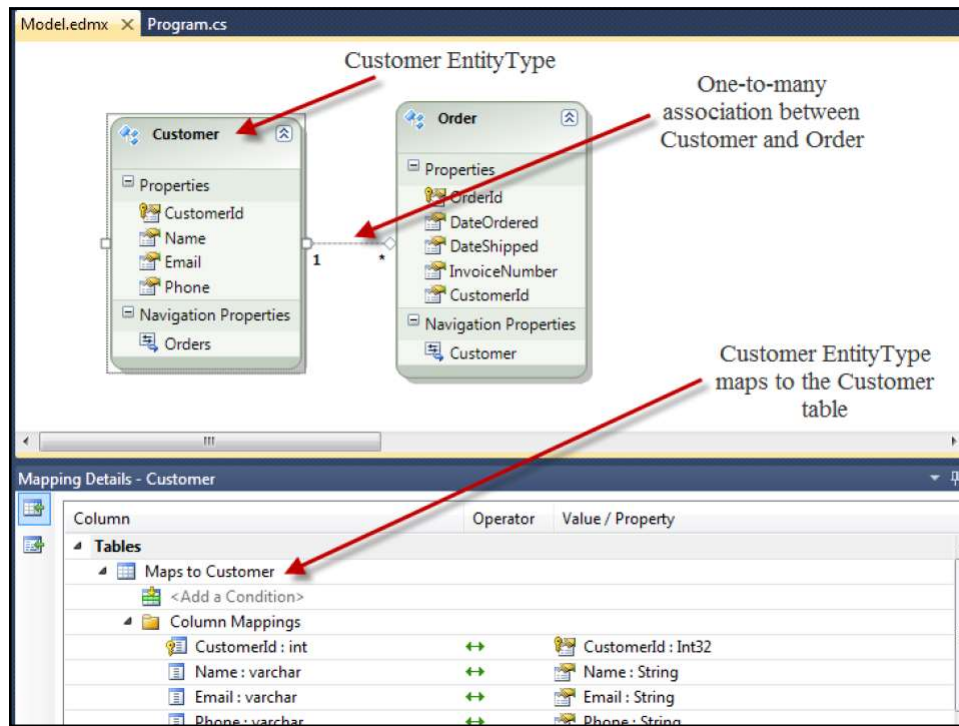
- Ovaj deo modela podseća na prethodni, kao što se može videti iz XML koda. Predstavlja šematsku reprezentaciju pridruženog skladišta za podatke.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- EF Runtime content -->
<edmx:Runtime>
  <!-- SQL content -->
  <edmx:storageModels>
    <Schema Namespace="SampleModel.Store" Alias="Self" Provider="System.Data.SqlClient" ProviderManifestToken="2008"
      xmlns:store="http://schemas.microsoft.com/ado/2007/11/data/entitytypeschemasamplesstore"
      xmlns="http://schemas.microsoft.com/ado/2008/02/edmx/11">
      <EntityContainer Name="SampleModelStoreContainer">
        <EntityType Name="Address" EntityType="SampleModel.Store.Address" store:type="Table" Schema="dbo" />
        <EntityType Name="Contact" EntityType="SampleModel.Store.Contact" store:type="Table" Schema="dbo" />
        <EntityType Name="VOfficedAddress" EntityType="SampleModel.Store.VOfficedAddress" store:type="View" store:schema="dbo" store:isView="true" />
        <AssociationSet Name="FK_Address_Contact" Association="SampleModel.Store.FK_Address_Contact">
          <End Role="Contact" EntityType="Contact" />
          <End Role="Address" EntityType="Address" />
        </AssociationSet>
      </EntityContainer>
      <EntityType Name="Address">
        <key>
          <PropertyRef Name="AddressID" />
        </key>
        <Property Name="AddressID" Type="int" Nullable="false" store:generatedPattern="Identity" />
        <Property Name="Street" Type="nvarchar" MaxLength="50" />
        <Property Name="Street2" Type="nvarchar" MaxLength="50" />
        <Property Name="City" Type="nvarchar" MaxLength="40" />
        <Property Name="StateProvince" Type="nvarchar" MaxLength="50" />
        <Property Name="CountryRegion" Type="nvarchar" MaxLength="50" />
        <Property Name="PostalCode" Type="nvarchar" MaxLength="50" />
        <Property Name="AddressType" Type="nvarchar" Nullable="false" MaxLength="50" />
        <Property Name="ContactID" Type="int" Nullable="false" />
        <Property Name="ModifiedDate" Type="datetime" Nullable="false" />
      </EntityType>
      <EntityType Name="Contact">
        <key>
          <PropertyRef Name="ContactID" />
        </key>
        <Property Name="ContactID" Type="int" Nullable="false" store:generatedPattern="Identity" />
        <Property Name="FirstName" Type="nvarchar" MaxLength="50" />
        <Property Name="LastName" Type="nvarchar" MaxLength="50" />
        <Property Name="Title" Type="nvarchar" MaxLength="50" />
        <Property Name="AddDate" Type="datetime" />
        <Property Name="ModifiedDate" Type="datetime" />
      </EntityType>
      <Association Name="FK_Address_Contact" />
    </Schema>
  </edmx:storageModels>
</edmx:Runtime>
```

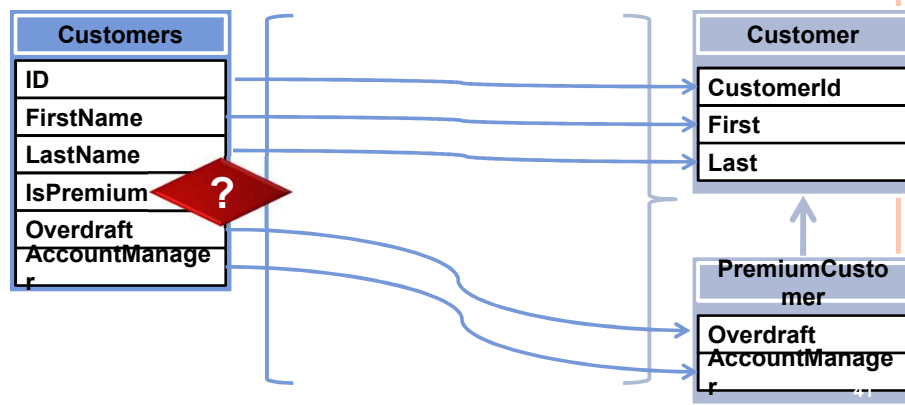
MAPIRANJE MSL: THE MAPPINGS

- Poslednja sekcija u EDMX fajlu, a ujedno i deo EDM modela je, mapiranje.
- Mapiranje je smešteno između konceptualnog sloja i sloja za skladištenje.
- Mapiranje je sekcija u fajlu, ali se mapiranje podešava po pravilu koristeći Dizajner, dvostrukim klikom na EDMX model fajl u *Solution Explorer*-u.
- Da bi videli *Mapping Details window*, desni klik na *Contact* a zatim selektuj *Table Mapping* iz menija..

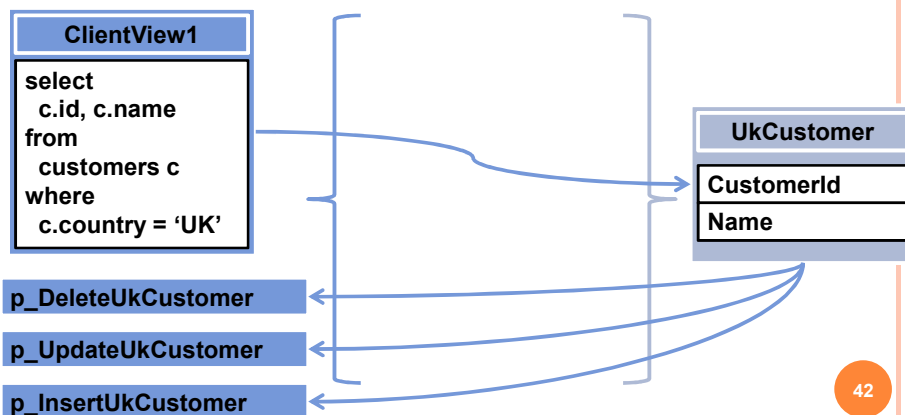




MAPPING EXAMPLES (2 – TPH)



MAPPING EXAMPLES (3 – VIEW + SPs)



EDM UPITI

```

private static void QueryContacts()
{
    using (var context = new SampleEntities())
    {
        var contacts = context.Contacts;
        foreach (var contact in contacts)
        {
            Console.WriteLine("{0} {1}",
                contact.FirstName.Trim(),
                contact.LastName);
        }
    }
    Console.WriteLine("Press Enter...");
    Console.ReadLine();
}

```

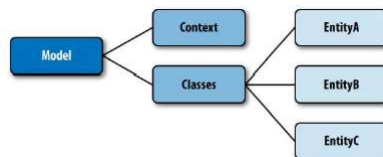
19.18

43

- Odakle klasa za objekat context?
- Jedna od karakteristika EDM alata za dizajn je da Dizajner automatski generiše kod na osnovu modela. Fajl *Designer.cs* je dodat modelu u *Solution Explorer*.



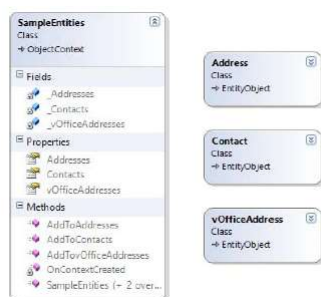
- Automatsko generisanje koda koji se dodaje modelu
- Pošto je fajl generisan automatski, ne može da se edituje direktno.
- Generator čita konceptualni sloj modela i na osnovu njega kreira *ObjectContext* klasu zasnovanu na *EntityContainer-u*, a zatim jednu klasu entiteta za svaki entitet u modelu.



19.18

44

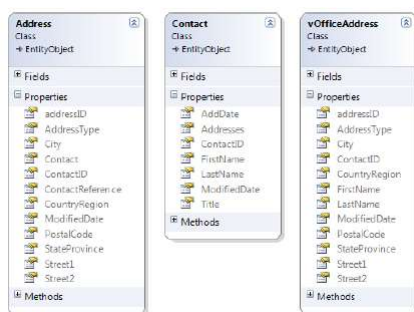
○ Generisane klase



Klase vidljive iz VisualStudio ClassView DesignerView-a

45

○ Klase entiteta vidljive iz Dizajnera



46

ENTITYCLIENT

- Upotreba klasa iz navedenog prostora imena daje mogućnost dobavljanja podataka. Na primer.

```
static void EntityClientQueryContacts()
{
    using (EntityConnection conn = new
        EntityConnection("name=SampleEntities"))
    {
        conn.Open();
        var queryString = "SELECT VALUE c " +
            "FROM SampleEntities.Contacts AS c " +
            "WHERE c.FirstName='Robert'";
        EntityCommand cmd = conn.CreateCommand();
        cmd.CommandText = queryString;
        using (EntityDataReader rdr =
            cmd.ExecuteReader(CommandBehavior.SequentialAccess |
                CommandBehavior.CloseConnection))
        {
            while (rdr.Read())
            {
                var firstname = rdr.GetString(1);
                var lastname = rdr.GetString(2);
                var title = rdr.GetString(3);
                Console.WriteLine("{0} {1} {2}",
                    title.Trim(), firstname.Trim(), lastname);
            }
        }
        conn.Close();
        Console.WriteLine("Press Enter...");
        Console.ReadLine();
    }
}
```

19:18

47

IZMENE NA ENTITETIMA I SNIMANJE PROMENA

- Čuvanje traga o entitetima
- KlaseObjectContext i SampleEntities class koji se nasleđuju iz klaseObjectContext koriste se za izvršavanje upita.
- ObjectContext ima metodu *SaveChanges*, koja može da sačuva sve promene na entitetima u bazi. Pozivom metode *SaveChanges* biće provereni svi objekti, tačnije njihova stanja, *ObjectStateEntry* kojima se upravlja od strane konteksta čije stanje nije *Unchanged*, a zatim će biti korišćeni detalji za izgradnju posebnih upita Insert, Update, and Delete koji se šalju bazi.

```
using (PEF context = new PEF())
{
    var contact = context.Contacts.First();
    contact.FirstName = "Julia";
    contact.ModifiedDate = DateTime.Now;
    context.SaveChanges();
}
```

```
exec sp_executesql N'update [dbo].[Contact]
set [FirstName] = @0, [ModifiedDate] = @1
where ([ContactID] = @2)
',N'@0 nvarchar(50),@1 datetime2(7),@2 int',@0=N'Julia',
@1='2009-11-30 09:27:20.3335098',@2=1
```

19:18

DODAVANJE NOVOG OBJEKTA

```
var contact =
    context.Contacts.Where(c => c.FirstName == "Robert").First();

var address = new Address();
address.Street1 = "One Main Street";
address.City = "Burlington";
address.StateProvince = "VT";
address.AddressType = "Business";
address.ModifiedDate = DateTime.Now;

address.Contact = contact;

context.SaveChanges();
```

19.18

49

DODAVANJE NOVIH RODITELJSKIH ODNOSNO DEČJIH OBJEKATA

```
var contact = Contact.CreateContact
(0, "Camey", "Combs", DateTime.Now, DateTime.Now);

var address = new Address();
address.Street1 = "One Main Street";
address.City = "Olympia";
address.StateProvince = "WA";
address.AddressType = "Business";
address.ModifiedDate = DateTime.Now;
address.Contact = contact;

context.Contacts.AddObject(contact);
context.SaveChanges();
```

19.18

```
exec sp_executesql N'insert [dbo].[Contact]([FirstName], [LastName], [Title],
[AddDate], [ModifiedDate])
values (@0, @1, null, @2, @3)
select [ContactID]
from [dbo].[Contact]
where @@ROWCOUNT > 0 and [ContactID] = scope_identity()',
N'@0 nvarchar(50),@1 nvarchar(50),@2 datetime2(7),@3 datetime2(7)',
@0=N'Camey',@1=N'Combs',@2='2009-08-30 09:27:31.7449098',
@3='2009-11-30 09:27:31.7449098'
exec sp_executesql N'insert [dbo].[Address]([Street1], [Street2], [City],
[StateProvince], [CountryRegion], [PostalCode], [AddressType],
[ContactID], [ModifiedDate])
values (@0, null, @1, @2, null, @3, @4, @5)
select [addressID]
from [dbo].[Address]
where @@ROWCOUNT > 0 and [addressID] = scope_identity()',
N'@0 nvarchar(50),@1 nvarchar(50),@2 nvarchar(50),@3 nvarchar(50),
@4 int,@5 datetime2(7)',
@0=N'One Main Street',@1=N'Olympia',@2=N'WA',@3=N'Business',
@4=714,@5='2009-11-30 09:27:31.7449098'
```

BRISANJE

```
System.Data.EntityKey contactKey =
new System.Data.EntityKey("PEF.Contacts", "ContactID", 438);
var contact = context.GetObjectByKey(contactKey);
context.DeleteObject(contact);
context.SaveChanges();
```

```
Employees emp =
ctx.Employees.Where(x=>x.EmployeeID==14).First();
ctx.Employees.Remove(emp);
ctx.SaveChanges();
```

```
exec sp_executesql N'SELECT
[Extent1].[ContactID] AS [ContactID],
[Extent1].[FirstName] AS [FirstName],
[Extent1].[LastName] AS [LastName],
[Extent1].[Title] AS [Title],
[Extent1].[AddDate] AS [AddDate],
[Extent1].[ModifiedDate] AS [ModifiedDate]
FROM [dbo].[Contact] AS [Extent1]
WHERE [Extent1].[ContactID] = @p0',N'@p0 int',@p0=438
exec sp_executesql N'delete [dbo].[Contact]
where ([ContactID] = @0)',N'@0 int',@0=438
```

19:18