**Rukovanje pogledima**

**00:10:52**

Jedinica: 11 od 19

**Rezime**

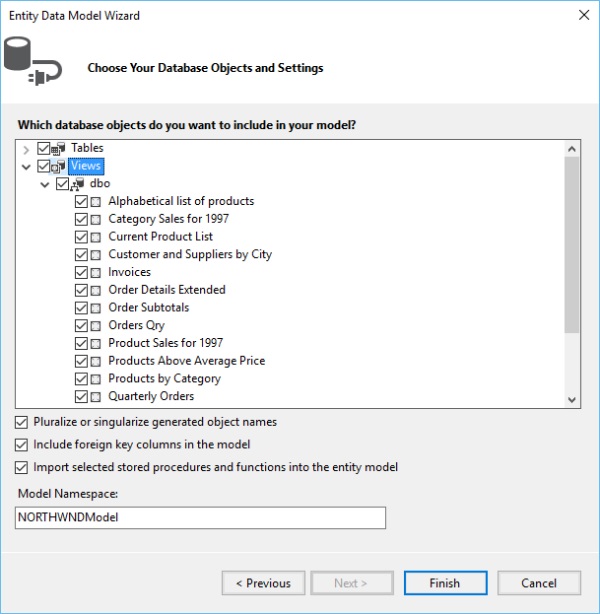
* Pogled se može doživeti kao neka vrsta virtualne tabele, sa mogućnošću agregiranja podataka iz više različitih tabela.
* EF poseduje ugrađenu podršku za rad sa pogledima.
* EF tretira rezultate, vraćene od strane pogleda, kao entitete.
* Pogled nema mogućnost upisivanja podataka, već je njegova osnova funkcija pregled podataka.
* Postoji tehnika zahvaljujući kojoj se pogled može oplemeniti mogućnošću upisivanja, izmene i brisanja podataka, i to korišćenjem uskladištenih procedura.

Baš kao što i samo ime kaže, pogled u bazi podatka omogućava pregled podataka. Sada se može postaviti pitanje u čemu je razlika između tabela i pogleda? Pa, pogled se može doživeti kao neka vrsta virtualne tabele, sa mogućnošću agregiranja podataka iz više različitih tabela. Tako je moguće značajno uticati na poboljšanje čitljivosti, ali i bezbednosti podataka. U ovoj lekciji biće demonstrirane tehnike za rad sa pogledima.

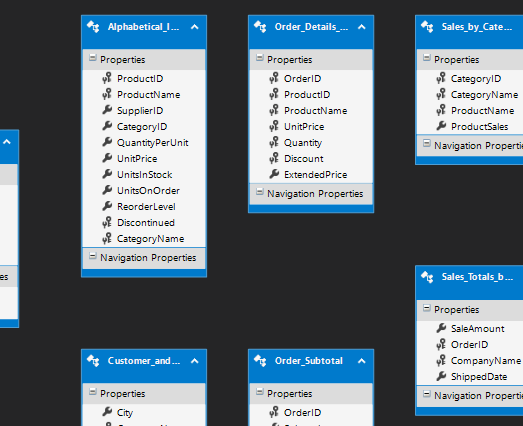
|  |
| --- |
| ***Napomena:****o pogledima više reči je bilo u kursu SQL programming and administration.* |

**Pogledi i EF**

EF poseduje ugrađenu podršku za rad sa pogledima. Najprostije rečeno, EF rezultate vraćene od strane pogleda tretira kao entitete. Prilikom kreiranja modela na osnovu baze podataka u okviru Entity Data Model čarobnjaka, postoji opcija za uključivanje pogleda u model, kao na slici 11.1.

   
*Slika 11.1 – Uključivanje pogleda u konceptualni model*

Kao referentna baza podataka, u ovoj lekciji biće korišćena baza *Northwind*. Tako je u model potrebno uključiti sve tabele i sve poglede ove baze podataka, kao na slici 11.1. Nakon uključivanja pogleda u model može se primetiti da EF poglede tretira kao entitete, i da se zapravo oni pojavljuju na radnoj površini dizajnera kao entiteti, što se može videti na slici 11.2.



*Slika 11.2 – Pogledi kao entiteti unutar konceptualnog modela*

|  |
| --- |
| ***Napomena:****još jedna stvar koja se odmah nakon kreiranja modela uočava jesu upozorenja koja se dobijaju unutar Error List prozora, kao na slici 11.3.*    https://www.link-elearning.com/linkdl/coursefiles/1142/MDA_11_03.jpg    *Slika 11.3 – Greške koje se dobijaju prilikom uključivanja pogleda u konceptualni model*  *S obzirom na to da EF poglede tretira kao entitete, Visual Studio – razvojno okruženje, obaveštava nas da kreirani entiteti nemaju primarni ključ, te da su oni kreirani tako da budu samo čitljivi. Stoga, nije zgoreg proći ručno kroz sve kreirane entitete na dijagramu i odabrati samo jednu kolonu kao kolonu primarnog ključa, što je prikazano na slici 11.4.*  https://www.link-elearning.com/linkdl/coursefiles/1142/MDA_11_04.jpg    *Slika 11.4 – Opcija za definisanje kolone, kao kolone identiteta* |

**Kako EF tretira podatke vraćene od strane pogleda?**

 Kao proste tipove

 Kao entitete

 Kao entitete

 Kao funckije

**Korišćenje pogleda**

Nakon uključivanja pogleda u EF model moguće je koristiti entitete kreirane na osnovu pogleda u kodu. Sledeći primer to i ilustruje.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | using (NORTHWNDEntities ctx = new NORTHWNDEntities())              {                  var result = from t in ctx.Order\_Details\_Extendeds select t;                  foreach (Order\_Details\_Extended t in result)                  {                      Console.WriteLine("{0} {1} {2} {3} {4}",                          t.ExtendedPrice, t.ProductName, t.ProductName,                          t.ExtendedPrice, t.UnitPrice);                  }              } |

U prikazanom primeru iskorišćen je dobro poznati obrazac za korišćenje entiteta. Unutar using bloka instanciran je objekat kontekstne klase i definisan kod za korišnjenje entiteta kreiranog na osnovu pogleda. Vrlo je jasno da se pogledima rukuje na isti način kao i tabelama.

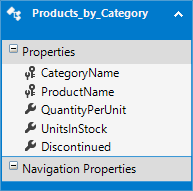
**Pogledi sa funkcijom pisanja**

Sam po sebi, pogled nema mogućnost upisivanja podataka, već je njegova osnova funkcija pregled podataka. Ipak, postoji tehnika zahvaljujući kojoj se pogled može oplemeniti mogućnošću upisivanja podataka. Takva tehnika podrazumeva kreiranje uskladištene procedure, koja bi obavljala operacije koje inicijalno nisu dostupne pogledu.

Pogledajmo kao primer pogled *Products by Category*, Northwind baze podataka:

|  |  |
| --- | --- |
| 1  2  3  4 | create view "Products by Category" AS  SELECT Categories.CategoryName, Products.ProductName, Products.QuantityPerUnit, Products.UnitsInStock, Products.Discontinued  FROM Categories INNER JOIN Products ON Categories.CategoryID = Products.CategoryID  WHERE Products.Discontinued <> 1 |

Ovo je pogled koji omogućava pregled proizvoda po kategorijama. Proizvodi (Products) i kategorije (Category) su dve zasebne tabele u bazi podataka, a unutar pogleda one se agregiraju korišćenjem jedne [INNER JOIN](https://www.link-elearning.com/linkdl/opisPojma.php?id=142490) naredbe. Umesto da se svaki put piše komplikovan upit za selektovanje podataka iz dve tabele, takva logika je enkapsulirana unutar prikazanog pogleda. Kada se ovakav pogled uključi u EF model, situacija je kao na slici 11.5.



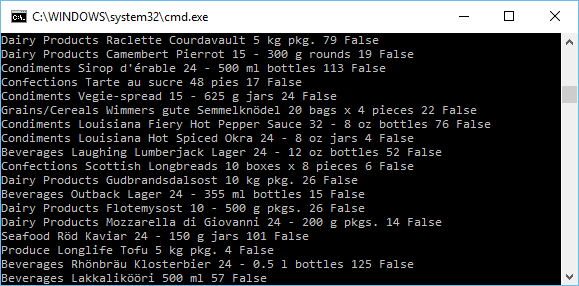
*Slika 11.5 – Products\_by\_Category entitet*

|  |
| --- |
| ***Napomena:****EF je inicijalno obeležio više kolona kao Entity Key, ali je to na ranije prikazan način svedeno samo na dve prve kolone.* |

Da bi se pregledali podaci koje Products\_by\_Category izlaže, može se napisati sledeći kod:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | using (NORTHWNDEntities ctx = new NORTHWNDEntities())  {      var result = from t in ctx.Products\_by\_Categories select t;      foreach (Products\_by\_Category cp in result)      {          Console.WriteLine("{0} {1} {2} {3} {4}",              cp.CategoryName, cp.ProductName, cp.QuantityPerUnit,              cp.UnitsInStock, cp.Discontinued);      }  } |

Rezultat prikazanog koda je kao na slici 11.6.



*Slika 11.6 – Podaci koji se čitaju korišćenjem pogleda Products\_by\_Category*

Sada se postavlja pitanje kako omogućiti kreiranje, ažuriranje i brisanje entiteta, koji su kreirani korišćenjem pogleda. Ranije je rečeno da se za tako nešto koriste uskladištene procedure.

U nastavku će biti kreirana uskladištena procedura za unos novog entiteta. Ona će imati sledeći izgled:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | CREATE PROCEDURE AddProductAndCategory  @CategoryName nvarchar(15),  @ProductName nvarchar(40),  @QuantityPerUnit nvarchar(20),  @UnitsInStock smallint,  @Discontinued bit  AS  /\* Determine whether the category already exists. \*/  DECLARE @CategoryCount Int  SET @CategoryCount = (SELECT COUNT(CategoryName) FROM Categories WHERE CategoryName=@CategoryName)  /\* When @CategoryCount equals 0, the category doesn't exist.  The call to INSERT INTO will add the category. \*/  IF @CategoryCount = 0  BEGIN  INSERT INTO Categories(CategoryName)  VALUES (@CategoryName)  END  /\* Obtain the category's Id value for use in adding a new record to the  Product table. \*/  DECLARE @CatId Int  SET @CatId = (SELECT CategoryID FROM Categories WHERE CategoryName=@CategoryName)  /\* Perform the insertion. \*/  INSERT INTO Products(ProductName, QuantityPerUnit, UnitsInStock, Discontinued, CategoryID)  VALUES (@ProductName, @QuantityPerUnit, @UnitsInStock, @Discontinued, @CatId) |

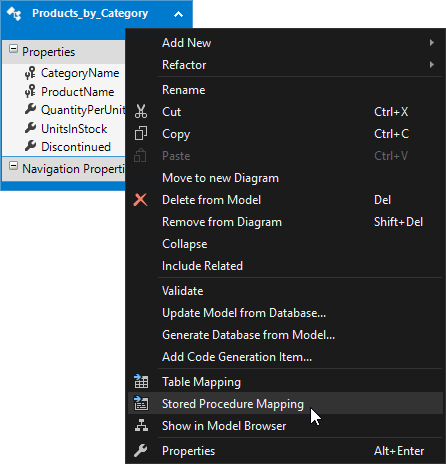
Unutar procedure prvo se obavlja definisanje ulaznih parametara. To su upravo oni parametri koji predstavljaju svojstva unutar pogleda. Na početku uskladištene procedure vrši se provera postojanja kategorija, koja se pokušava kreirati. Ukoliko kategorija postoji, nju nije potrebno kreirati, i to se upravo i događa unutar uskladištene procedure. Unutar promenljive CategoryCount smešta se broj kategorija sa imenom kategorije koja se pokušava kreirati. Ukoliko je taj broj 0, znači da kategorija ne postoji, te se stoga ona kreira.

Nakon eventualnog kreiranja kategorije potrebno je doći do id vrednosti kategorije za proizvod koji se ubacuje u bazu, zato što je ta vrednost potrebna prilikom unosa novog proizvoda. Bez obzira da li je kategorija upravo kreirana ili je od ranije postojala u bazi, tehnika dolaska do id vrednosti kategorije je identična, i uvek se izvršava.

Kada je dobavljena id vrednost kategorije, preostaje unos proizvoda. Proizvod se unosi u tabelu Products, a vrednosti koje se unose su ulazni parametri uskladištene procedure, i id vrednost kategorije koja je upravo kreirana.

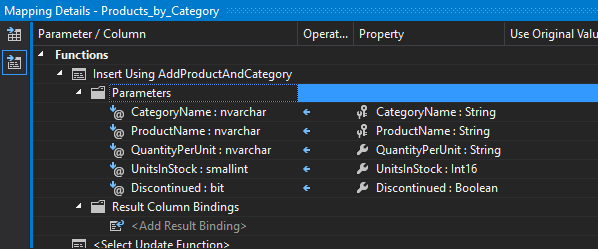
Nakon kreiranja uskladištene procedure, potrebno ju je uključiti u postojeći EF model, kao što je to objašnjeno u prethodnoj lekciji.

Da bi se ovako kreirana uskladištena procedura iskoristila za omogućavanje unosa podataka za entitet koji je kreiran na osnovi pogleda, potrebno je kliknuti desnim tasterom miša na Products\_by\_Category entitet i odabrati opciju *Stored Procedure Mapping*, kao na slici 11.7.



*Slika 11.7 – Opcija za definisanje mapiranja uskladištenih procedura*

Aktiviranjem opcije *Stored Procedure Mapping*, otvara se prozor za definisanje mapiranja. Okruženje omogućava odabir funkcija koje će obavljati Insert, Update i Delete operacije. Tako je za funkciju koja obavlja Insert operaciju, potrebno odabrati upravo kreiranu uskladištenu proceduru. Efekat će biti kao na slici 11.8.

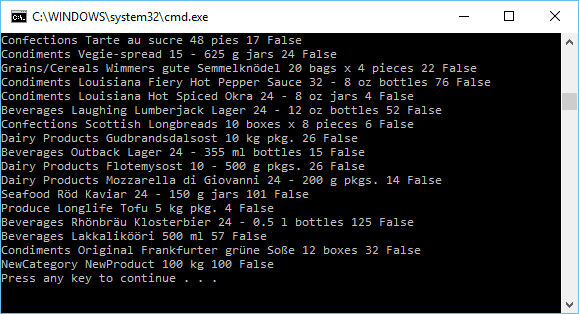


*Slika 11.8 – Podešavanje funkcije koja će obaviti Insert operaciju*

Nakon svih preduzetih akcija moguće je napisati kod za kreiranje novog entiteta, kao u sledećem primeru:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | using (NORTHWNDEntities ctx = new NORTHWNDEntities())              {                  //line for inserting entity based on view                  ctx.AddProductAndCategory("NewCategory", "NewProduct", "100 kg", 100, false);                  ctx.SaveChanges();                  var result = from t in ctx.Products\_by\_Categories select t;                  foreach (Products\_by\_Category cp in result)                  {                      Console.WriteLine("{0} {1} {2} {3} {4}",                          cp.CategoryName, cp.ProductName, cp.QuantityPerUnit,                          cp.UnitsInStock, cp.Discontinued);                  }              } |

Efekat prikazanog koda će biti kao na slici 11.9.



*Slika 11.9 – Pročitani podaci korišćenjem pogleda, gde sa jasno vidi da je unos novih podataka obavljen uspešno*

Evidentno je da je unos prošao bez ikakvih problema, a uvidom u stanje u bazi podataka može se videti da je kreirana nova kategorija i nov proizvod.