**PROJEKAT iz predmeta “BAZE PODATAKA 3”**

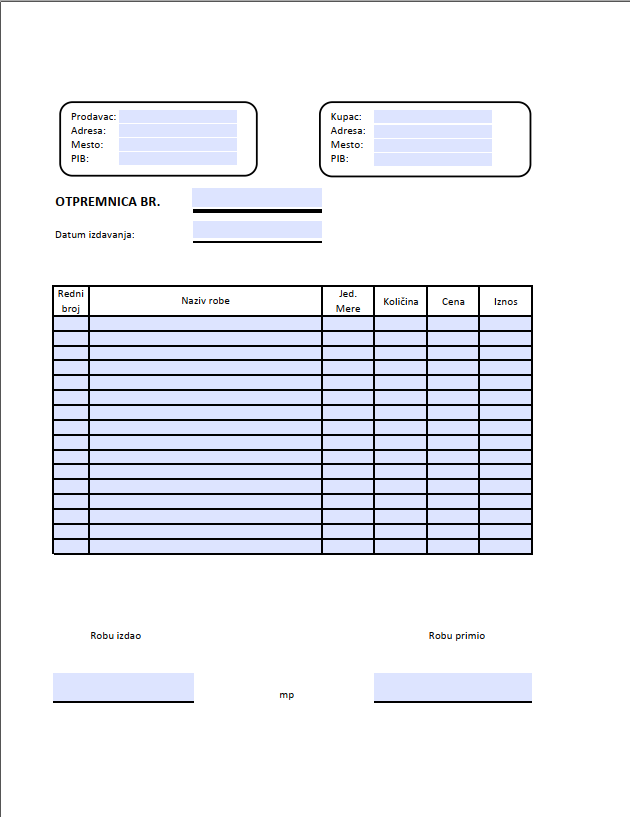
|  |  |  |  |
| --- | --- | --- | --- |
| **IME I PREZIME** | | **Milica Stojanović** | |
| Broj indeksa | | **3019/2023** | |
| Studijski modul | | Informacioni sistemi | |
| Stečena diploma osnovnih akademskih studija | | Diplomirani inženjer organizacionih nauka | |
| **NAZIV PROJEKTA** | | Sistem za praćenje rada rada preduzeća za pružanje usluga servisiranja električnih uređaja | |
| Napomena | |  | |
| *Verzija* | *Datum promene* | | *Opis promene* | |
| 1.1. | 9.11.2023. | | **Određena odekvatna dokumentacija** | |
| 2.1. | 15.11.2023. | | **Korigovani podmodeli PMOV-a i relacioni modeli** | |
|  |  | |  | |
|  |  | |  | |
|  |  | |  | |
|  |  | |  | |

# OPIS PROJEKTA

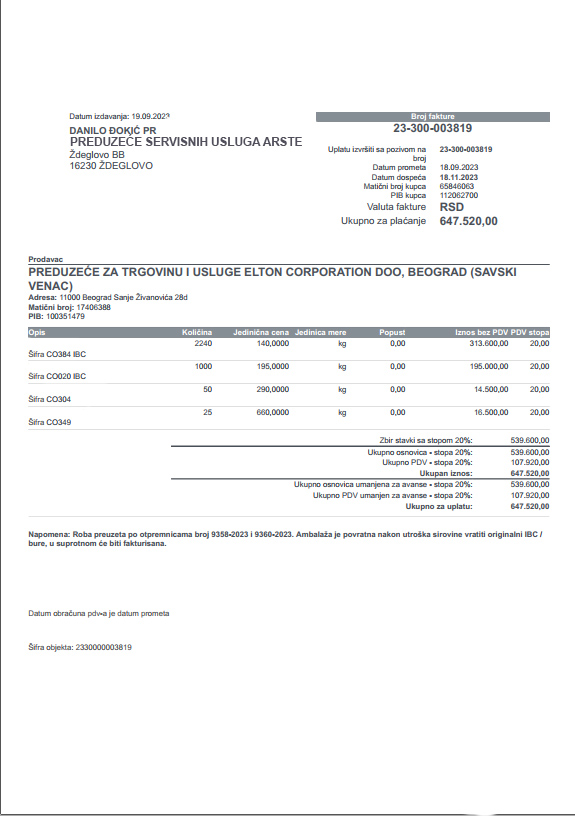
## Opis podsistema za koji se projektuje baza podataka

|  |
| --- |
| Projektuje se baza podataka za sistem za praćenje rada preduzeća za pružanje usluga servisiranja električnih uređaja, od nabavke rezervnih delova, do procesa servisiranja električnih uređaja.  Na samom početku nabavljaju se rezervni delovi koji mogu biti potrebni prilikom servisiranja, poput rezervnih kablova, baterija, itd. Dokumenti koji prate sve to su otpremnica i faktura, na kojima se nalaze podaci dobavljača, kupca, kao i podaci o delovima. Nakon što roba stigne kupcu izdaje se prijemnica i tada se povećava količina sirovina na skladištu gde su delovi bili poručeni.  Nakon što klijent dođe u radnju za servisiranje izdaje se servisna prijemnica, koja potvrđuje da je uređaj primljen da je neophodno izvršiti njegovu popravku. On sadrži podatke o uređaju, kvaru, radnji, serviser, kao i klijentu. Pošto je utvrđeno šta je kvar, izdaje se nalog za trebovaje, kojim se iz skladišta sa rezervnim delovima poručuju delovi koji su neophodni za tu popravku I ostale popravke koje je neophodno izvršiti.  U toku procesa servisiranja izdaje se radni nalog, koji izdaje serviser, gde se između ostalog nalaze i podaci o utrošenim delovima, kao i samo vreme za koje je popravka završena, kako bi posle toga na osnovu njega bilo moguće izdati račun kupcu. |

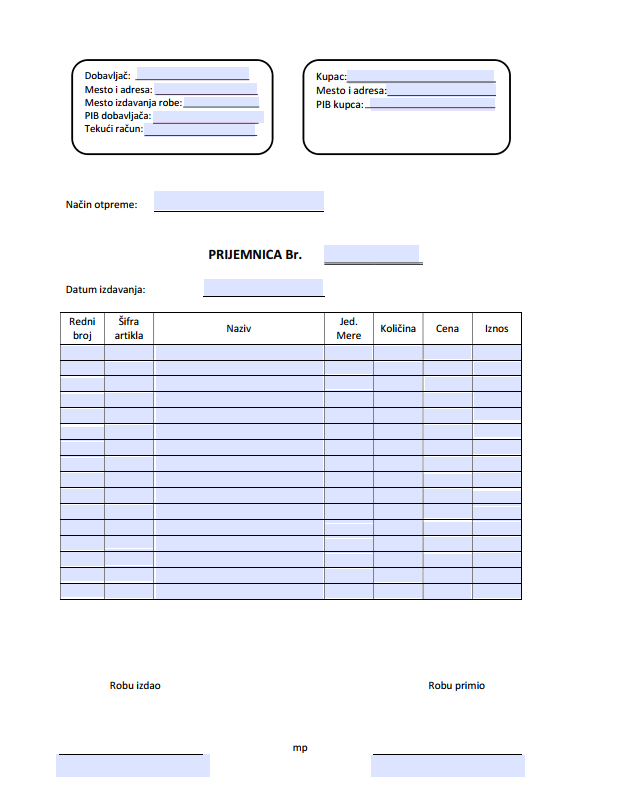
## Poslovna dokumenta koja se koriste u procesu

**

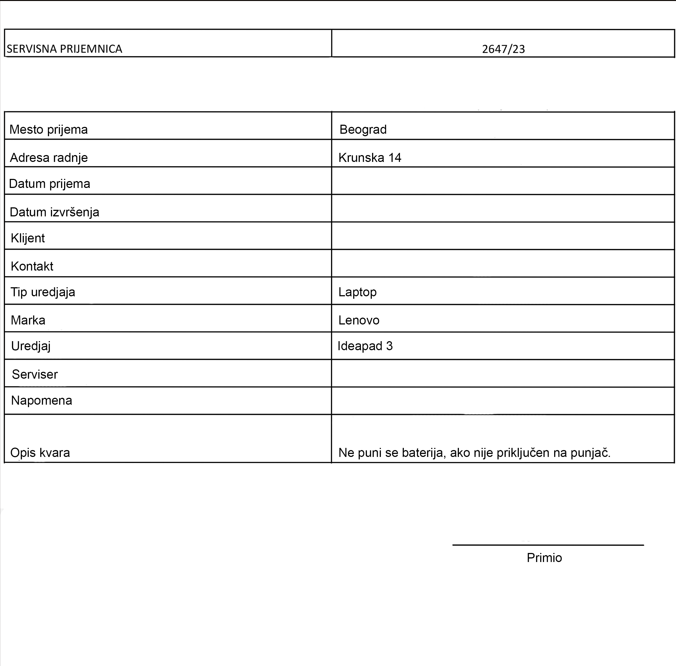
Slika 1. Otpremnica za nabavku rezervnih delova

**

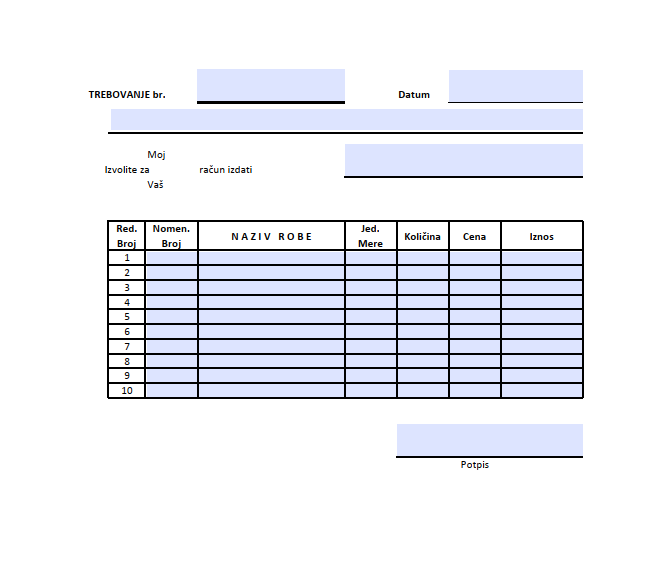
Slika 2. Faktura za nabavku rezervnih delova



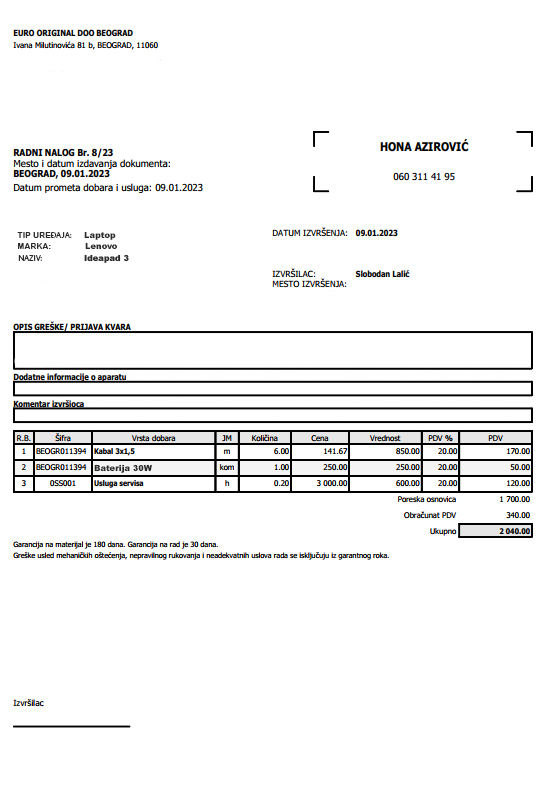
Slika 3. Prijemnica



Slika 4. Servisna prijemnica



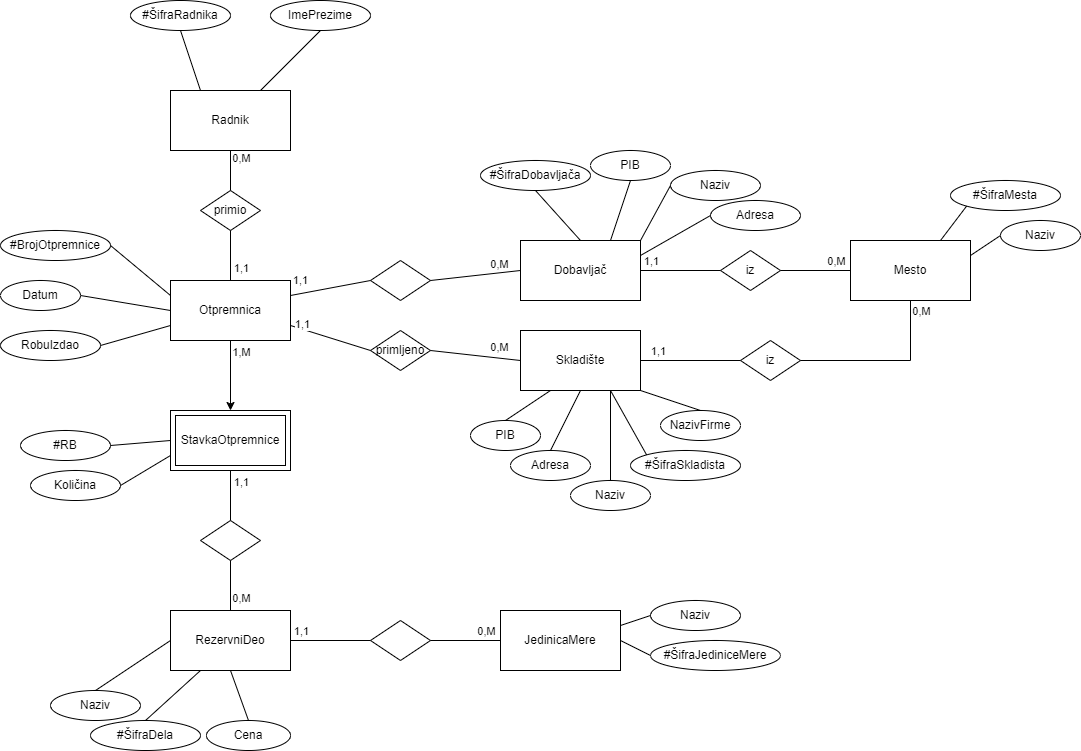
Slika 5. Trebovanje sirovina iz skladišta



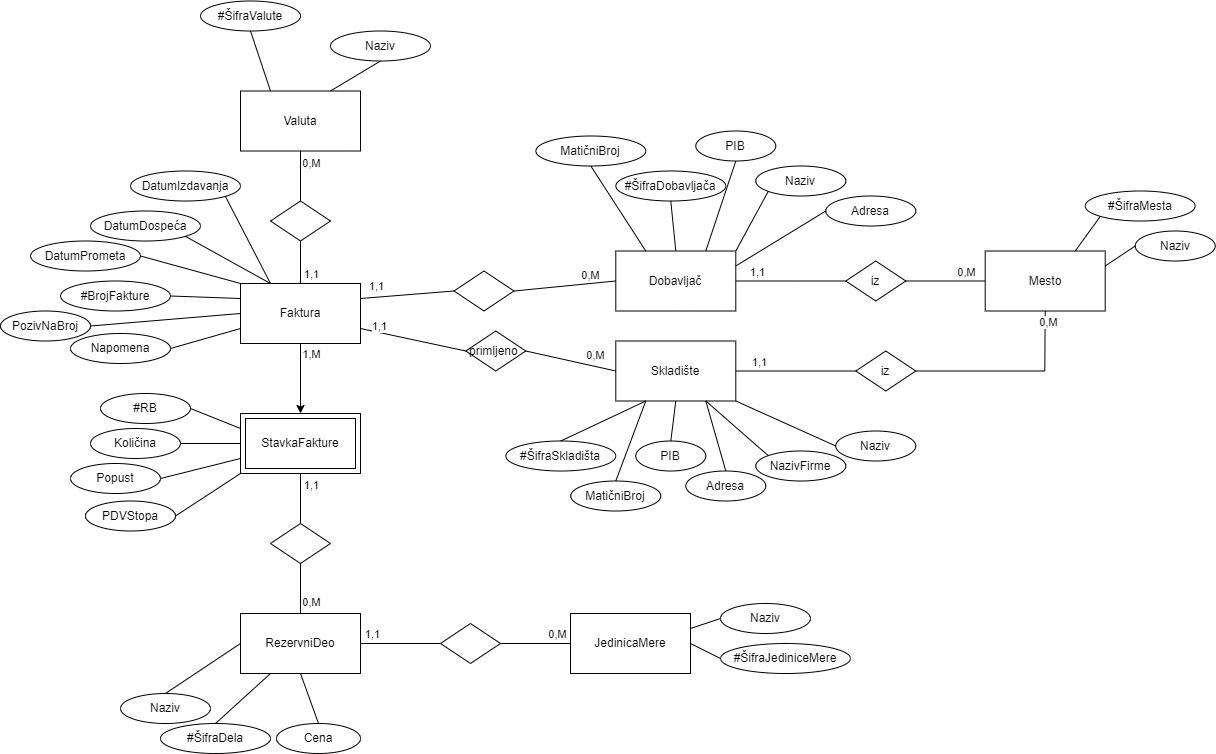
Slika 6. Radni nalog

# PROJEKTOVANJE BAZE PODATAKA

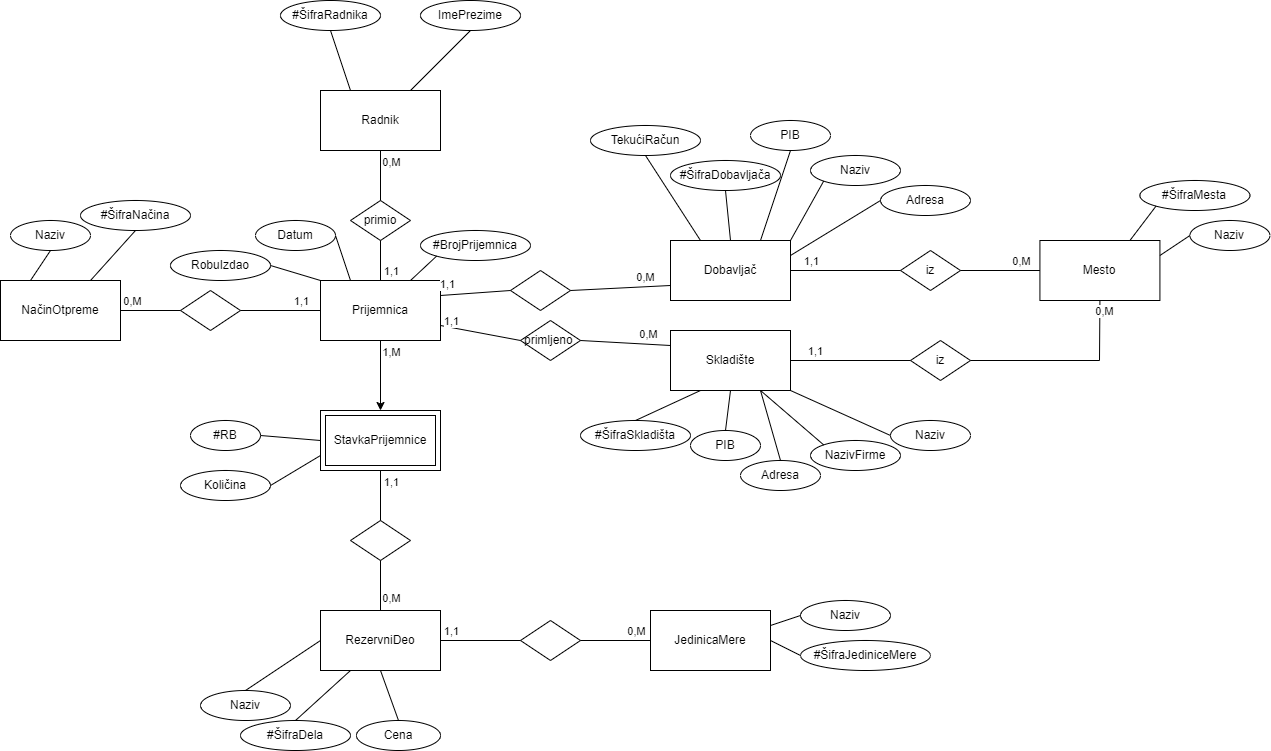
## Model podatka



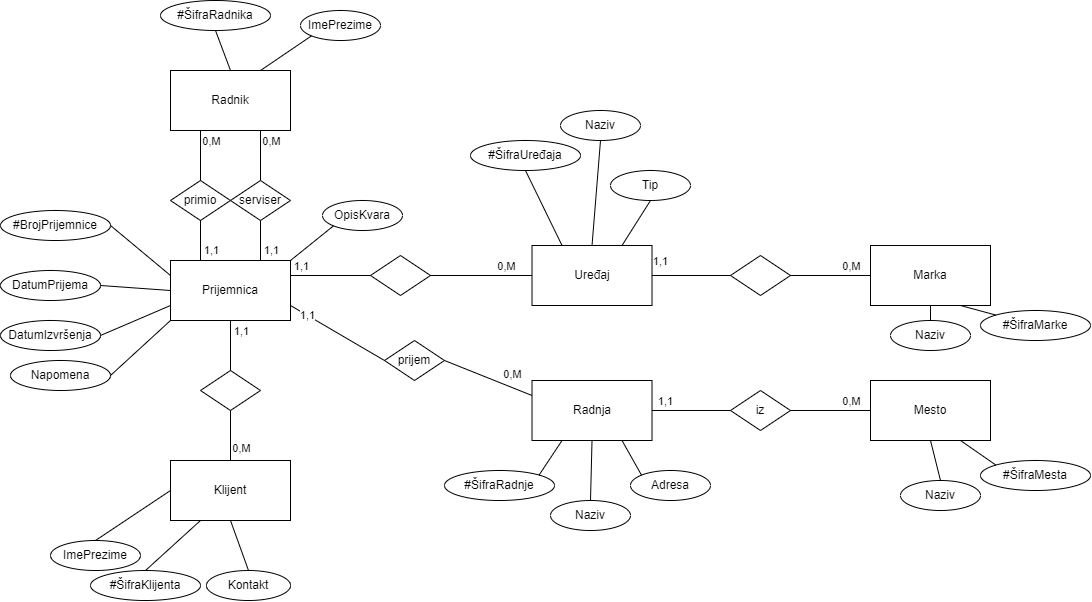
Slika 7. PMOV za dokument Otpremnica za nabavku rezervnih delova



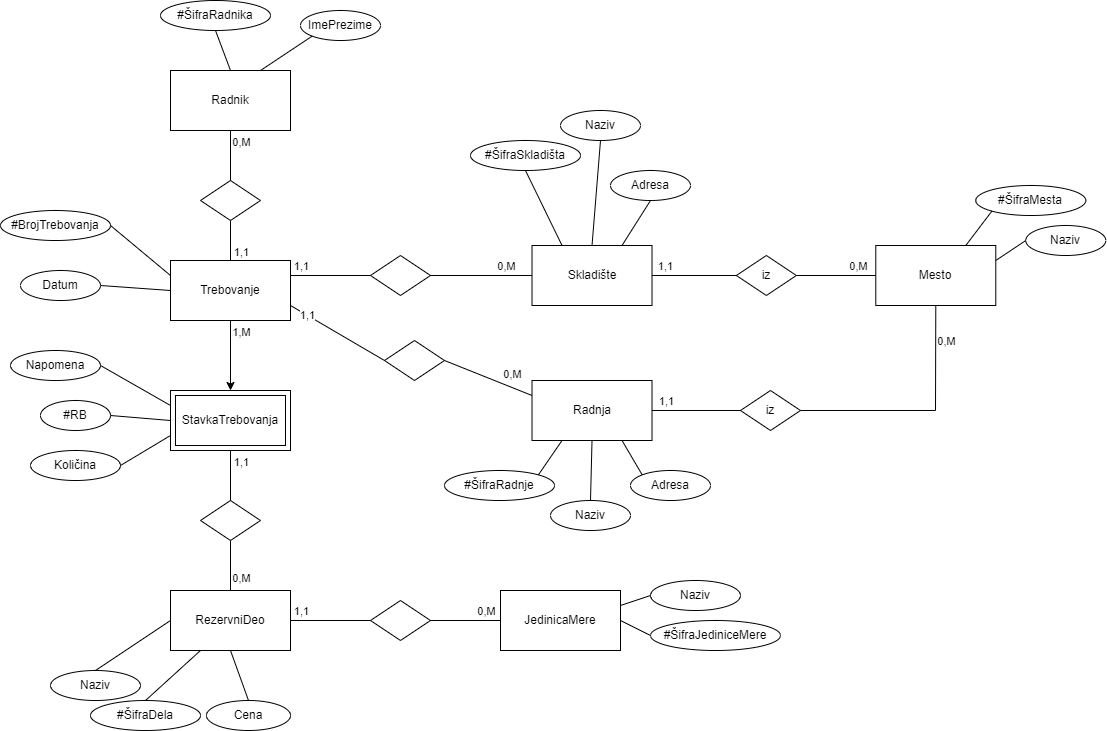
Slika 8. PMOV za dokument Faktura za nabavku rezervnih delova



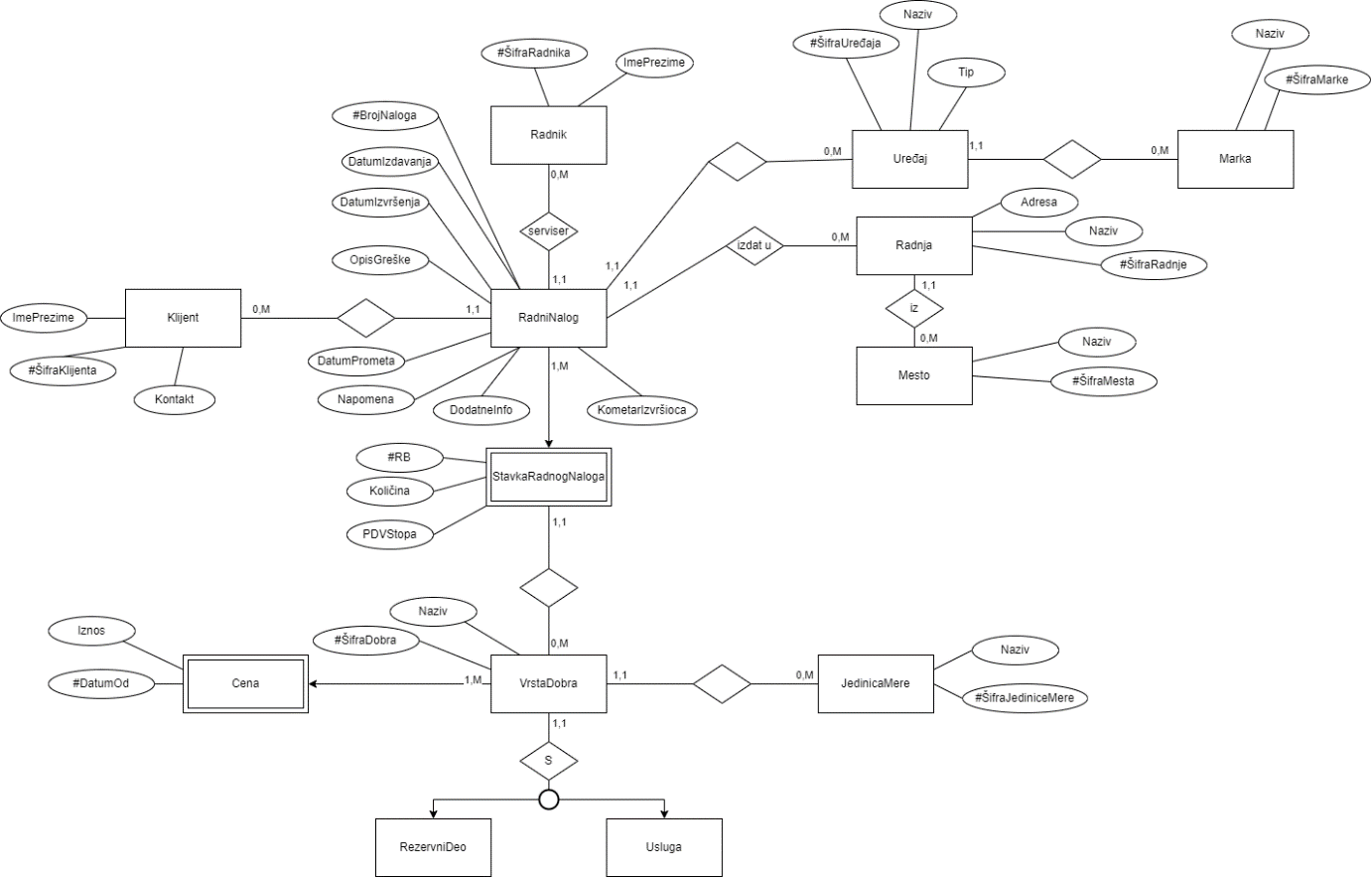
Slika 9. PMOV za dokument Prijemnica



Slika 10. PMOV za dokument Servisna prijemnica



Slika 11. PMOV za dokument Trebovanje rezervnih delova iz skladišta



Slika 12. PMOV za dokument Radni nalog

## Relacioni model

### Relacioni model za dokument Otpremnica za nabavku rezervnih delova

Otpremnica(BrojOtpremnice, Datum, RobuIzdao, *ŠifraDobavljača, ŠifraSkladišta, ŠifraRadnika*)  
StavkaOtpremnice(BrojOtpremnice, Rb, Količina, *ŠifraRezervnogDela*)  
RezervniDeo(ŠifraRezervnogDela, Naziv, Cena, *ŠifraJedniceMere*)  
JedinicaMere(ŠifraJedniceMere, Naziv)  
Radnik(ŠifraRadnika, ImePrezime)  
Dobavljač(ŠifraDobavljača, PIB, Naziv, Adresa, *ŠifraMesta*)  
Skladište(ŠifraSkladišta, PIB, NazivFirme, Naziv, Adresa, *ŠifraMesta*)  
Mesto(ŠifraMesta, Naziv)

### Relacioni model za dokument Faktura za nabavku rezervnih delova

Faktura(BrojFakture, PozivNaBroj, Napomena, DatumPrometa, DatumDospeća, DatumIzdavanja, *ŠifraDobavljača, ŠifraSkladišta, ŠifraValute*)  
Valuta(ŠifraValute, Naziv)  
StavkaFakture (BrojFakture, Rb, Količina, Popust, PDVStopa, *ŠifraRezervnogDela*)  
RezervniDeo(ŠifraRezervnogDela, Naziv, Cena, *ŠifraJedniceMere*)  
JedinicaMere(ŠifraJedniceMere, Naziv)   
Dobavljač(ŠifraDobavljača, PIB, Naziv, Adresa, MatičniBroj, *ŠifraMesta*)  
Skladište (ŠifraSkladišta, PIB, NazivFirme, Naziv, Adresa, MatičniBroj, *ŠifraMesta*)  
Mesto(ŠifraMesta, Naziv)

### Relacioni model za dokument Prijemnica

Prijemnica(BrojPrijemnice, Datum, RobuIzdao, *ŠifraDobavljača, ŠifraSkladišta, ŠifraRadnika, ŠifraNačinaOtpreme*)  
NačinOtpreme(ŠifraNačinaOtpreme, Naziv)  
StavkaPrijemnice(BrojPrijemnice, Rb, Količina, *ŠifraRezervnogDela*)  
RezervniDeo(ŠifraRezervnogDela, Naziv, Cena, *ŠifraJedniceMere*)  
JedinicaMere(ŠifraJedniceMere, Naziv)  
Radnik(ŠifraRadnika, ImePrezime)  
Dobavljač(ŠifraDobavljača, PIB, Naziv, Adresa, TekućiRačun, *ŠifraMesta*)  
Skladište (ŠifraSkladišta, PIB, NazivFirme, Naziv, Adresa, *ŠifraMesta*)  
Mesto(ŠifraMesta, Naziv)

### Relacioni model za dokument Servisna prijemnica

ServisnaPrijemnica(BrojPrijemnice, DatumPrijema, DatumIzvršenja, Napomena, OpisKvara, *ŠifraUređaja, ŠifraRadnje, ŠifraRadnika, ŠifraRadnika2, ŠifraKlijenta*)  
Radnik(ŠifraRadnika, ImePrezime)  
Klijent(ŠifraKlijenta, ImePrezime, Kontakt)  
Uređaj(ŠifraUređaja, Naziv, Tip, *ŠifraMarke*)  
Marka(ŠifraMarke, Naziv)  
Radnja(ŠifraRadnje, Naziv, Adresa, *ŠifraMesta* )  
Mesto(ŠifraMesta, Naziv)

### Relacioni model za dokument Trebovanje rezervnih delova iz skladišta

Trebovanje(BrojTrebovanja, Datum, *ŠifraRadnje, ŠifraSkladišta, ŠifraRadnika*)  
StavkaTrebovanja(BrojTrebovanja, Rb, Napomena, Količina, *ŠifraRezervnogDela*)  
RezervniDeo(ŠifraRezervnogDela, Naziv, Cena, *ŠifraJedniceMere*)  
JedinicaMere(ŠifraJedniceMere, Naziv)  
Radnik(ŠifraRadnika, ImePrezime)  
Radnja(ŠifraRadnje, Naziv, Adresa, *ŠifraMesta* )  
Skladište (ŠifraSkladišta, Naziv, Adresa, *ŠifraMesta*)  
Mesto(ŠifraMesta, Naziv)

### Relacioni model za dokument Radni nalog

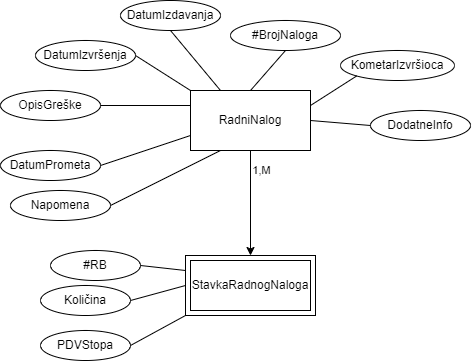
RadniNalog(BrojNaloga, DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje*)  
StavkaRadnogNaloga(BrojNaloga, Rb, Količina, PDVStopa, *ŠifraDobra*)  
VrstaDobra(ŠifraDobra, Naziv, *ŠifraJedniceMere*)  
Cena(ŠifraDobra, DatumOd, Iznos)  
Usluga(ŠifraDobra)  
RezervniDeo(ŠifraDobra)  
JedinicaMere(ŠifraJedniceMere, Naziv)  
Klijent(ŠifraKlijenta, ImePrezime, Kontakt)  
Uređaj(ŠifraUređaja, Naziv, Tip, *ŠifraMarke*)  
Marka(ŠifraMarke, Naziv)  
Radnja(ŠifraRadnje, Naziv, Adresa, *ŠifraMesta* )  
Mesto(ŠifraMesta, Naziv)  
Radnik(ŠifraRadnika, ImePrezime)

# DENORMALIZACIJA RELACIJA

## Denormalizacija relacija uz narušavanje 2NF

### Dokument Radni nalog

#### Normalizovani model podataka



RadniNalog(BrojNaloga, DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje*)

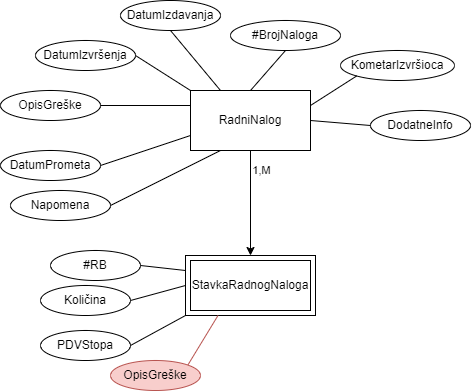
StavkaRadnogNaloga(BrojNaloga, Rb, Količina, PDVStopa, *ŠifraDobra*)

#### Funkcionalne zavisnosti pre denormalizacije

BrojNaloga 🡪 DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje,*

BrojNaloga, Rb 🡪 Količina, PDVStopa, *ŠifraDobra*

#### Denormalizovani model podataka



RadniNalog(BrojNaloga, DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje*)

StavkaRadnogNaloga(BrojNaloga, Rb, Količina, PDVStopa, OpisGreške, *ŠifraDobra*)

#### Funkcionalne zavisnosti posle denormalizacije

BrojNaloga 🡪 DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje*

BrojNaloga, Rb 🡪 Količina, PDVStopa, OpisGreške, *ŠifraDobra*

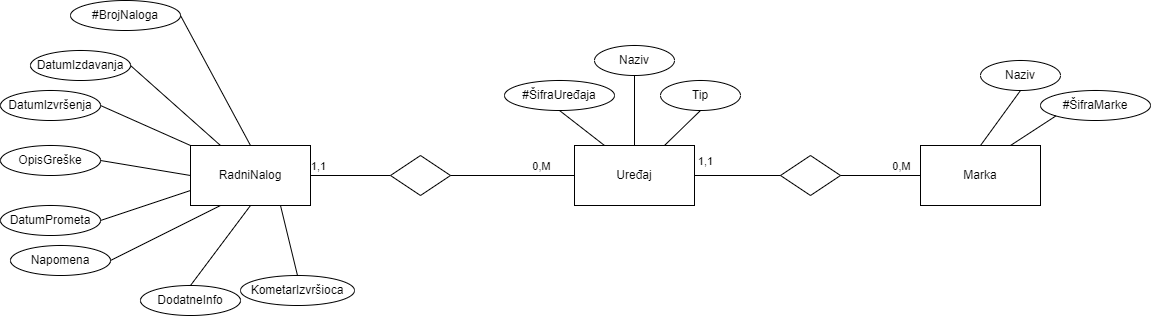
#### Identifikacija trigera

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba uraditi?** |
| **RadniNalog** | insert |  | NE |  |
| update | OpisGreške | DA | Prilikom izmene vrednosti kolone OpisGreške u tabeli RadniNalog, pokreće se triger koji izmenjenu vrednost ažurira u tabeli StavkaRadnogNaloga |
| delete |  | NE |  |
| **StavkaRadnogNaloga** | insert |  | DA | U kolonu OpisGreške tabele StavkaRadnogNaloga upisuje odgovarajuću vrednost za unetu vrednost kolone BrojNaloga |
| update | OpisGreške | DA | Sprečava direktnu izmenu |
| BrojNaloga | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |

## Denormalizacija relacija uz narušavanje 3NF

### Dokument Radni nalog

#### Normalizovani model podataka



RadniNalog(BrojNaloga, DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje*)

Uređaj(ŠifraUređaja, Naziv, Tip, *ŠifraMarke*)

Marka(ŠifraMarke, Naziv)

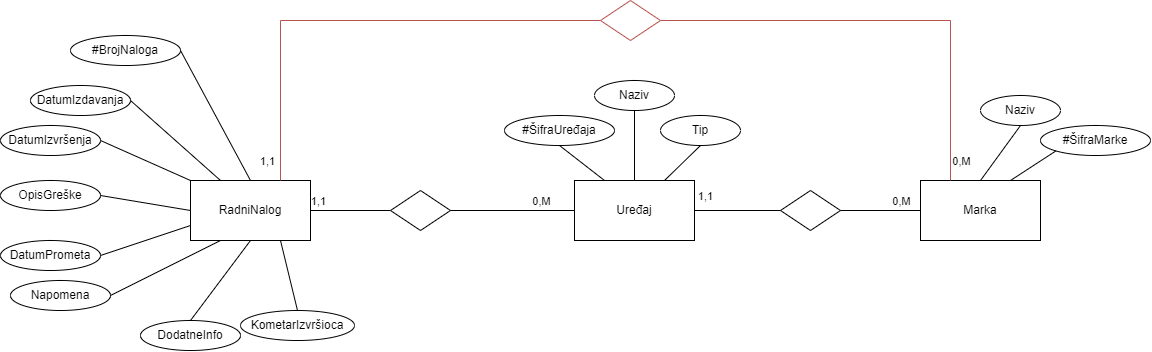
#### Funkcionalne zavisnosti pre denormalizacije

BrojNaloga 🡪 DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje*

ŠifraUređaja 🡪 Naziv, Tip, *ŠifraMarke*

ŠifraMarke 🡪 Naziv

#### Denormalizovani model podataka



RadniNalog(BrojNaloga, DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DodatneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje, ŠifraMarke*)

Uređaj(ŠifraUređaja, Naziv, Tip, *ŠifraMarke*)

Marka(ŠifraMarke, Naziv)

#### Funkcionalne zavisnosti posle denormalizacije

BrojNaloga 🡪 DatumIzdavanja, DatumIzvršenja, DatumPrometa, OpisGreške, Napomena, DotadneInformacije, KomentarIzvršioca, *ŠifraKlijenta, ŠifraUređaja, ŠifraRadnika, ŠifraRadnje, ŠifraMarke*

ŠifraUređaja 🡪 Naziv, Tip, *ŠifraMarke*

ŠifraMarke 🡪 Naziv

#### Identifikacija trigera

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba uraditi?** |
| **Uređaj** | insert |  | NE |  |
| update | ŠifraMarke | DA | Ukoliko postoji RadniNalog za Uređaj, ažurira kolonu ŠifraMarke u tabeli RadniNalog na osnovu nove vrednosti kolone ŠifraMarke u tabeli Uređaj |
| delete |  | NE |  |
| **RadniNalog** | insert |  | DA | U kolonu ŠifraMarke upisuje odgovarajuću vrednost na osnovu unete vrednosti kolone ŠifraUređaja |
| update | ŠifraUređaja | DA | U kolonu ŠifraMarke tabele RadniNalog upisuje odgovarajuću vrednost na osnovu nove vrednosti kolone ŠifraUređaja |
| ŠifraMarke | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |

# KORISNIČKI DEFINISANI TIPOVI I TRIGERI

## Definicija korisničkom tipa i njegovo korišćenje

### Distinct tip – PIB

CREATE OR REPLACE TYPE PIB AS OBJECT(  
 brojPIBa NUMBER(9),  
 MEMBER FUNCTION get\_brojPIBa RETURN NUMBER,  
 CONSTRUCTOR FUNCTION PIB(brojPIBa NUMBER) RETURN SELF AS RESULT  
)  
INSTANTIABLE NOT FINAL;

CREATE OR REPLACE TYPE BODY PIB AS

MEMBER FUNCTION get\_brojPIBa RETURN NUMBER IS

BEGIN

RETURN SELF.brojPIBa;

END;

CONSTRUCTOR FUNCTION PIB(brojPIBa NUMBER) RETURN SELF AS RESULT AS

BEGIN

IF brojPIBa >= 100000000 AND brojPIBa < 1000000000 THEN

SELF.brojPIBa := brojPIBa;

ELSE

RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'PIB mora imati 9 cifara!');

END IF;

RETURN;

END;  
END;

**Naredba za kreiranje tabele Skladiste**

CREATE TABLE Skladiste(

SifraSkladista NUMBER(10) PRIMARY KEY,

PIB PIB,

NazivFirme VARCHAR2(30),

Naziv VARCHAR2(30),

Adresa VARCHAR2(30),

SifraMesta NUMBER(10) NOT NULL,

CONSTRAINT mesto\_fk FOREIGN KEY (SifraMesta) REFERENCES Mesto(SifraMesta));

**Naredba za unos podataka u tabelu Skladiste**

INSERT INTO skladiste VALUES (1, PIB(123456789), ’Stark doo’, ‘Hala 1’, ‘Prote Stojana 92’, 1);

**Naredba za prikaz podataka iz tabele Skladiste**

SELECT s.SifraSkladista AS ID, s.Naziv, s.PIB.get\_brojPIBa() AS Skladiste  
FROM skladiste s;

**Naredba za izmenu podataka u tabeli Skladiste**

UPDATE skladiste s SET s.PIB.brojPIBa = 123456788 WHERE s.SifraSkladista = 1;

**Naredba za brisanje podataka iz tabele Skladiste**

DELETE skladiste s WHERE s.PIB.get\_brojPIBa() = 123456788;

### Struktuirani tip – Naziv

CREATE OR REPLACE TYPE Naziv AS OBJECT(  
 ime VARCHAR2(20),  
 tip VARCHAR2(10),  
 MEMBER FUNCTION get\_ime RETURN VARCHAR2,  
 MEMBER FUNCTION get\_tip RETURN VARCHAR2,  
 CONSTRUCTOR FUNCTION Naziv(ime VARCHAR2, tip VARCHAR2) RETURN SELF AS RESULT  
)  
INSTANTIABLE NOT FINAL;

CREATE OR REPLACE TYPE BODY Naziv AS

MEMBER FUNCTION get\_ime RETURN VARCHAR2 IS

BEGIN

RETURN SELF.ime;

END;

MEMBER FUNCTION get\_tip RETURN VARCHAR2 IS

BEGIN

RETURN SELF.tip;

END;

CONSTRUCTOR FUNCTION Naziv(ime VARCHAR2, tip VARCHAR2) RETURN SELF AS RESULT AS

BEGIN

SELF.ime := ime;

IF tip NOT IN ('doo', 'ad', 'od', 'kd') THEN

RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Tip preduzeca moze biti samo neka od ponudjenih vrednosti: doo, ad, od, kd!');

ELSE

SELF.tip := tip;

END IF;

RETURN;

END;  
END;

**Naredba za kreiranje tabele Poreski Obveznik**

CREATE TABLE dobavljac(

SifraDobavljaca NUMBER(10) PRIMARY KEY,

PIB PIB,

Naziv Naziv,

Adresa VARCHAR2(30),

SifraMesta NUMBER(10) NOT NULL,

CONSTRAINT mesto\_fk FOREIGN KEY (SifraMesta) REFERENCES Mesto(SifraMesta));

**Naredba za unos podataka u tabelu Poreski Obveznik**INSERT INTO dobavljac VALUES (1, PIB(123456888), Naziv(‘Lenovo’, ‘ad’), ‘Presevska 2’, 1);

**Naredba za prikaz podataka iz tabele Poreski Obveznik**SELECT d.naziv.get\_ime() AS Ime, d.naziv.get\_tip() AS TipPreduzeca   
FROM dobavljac d;

**Naredba za izmenu podataka u tabeli Poreski Obveznik**UPDATE dobavljac d SET d.naziv.ime = 'Lenovo1' WHERE d.SifraDibavljaca = 1;

**Naredba za brisanje podataka iz tabele Poreski Obveznik**DELETE dobavljac d WHERE d.naziv.get\_ime() = 'Lenovo1';

## Realizacija proceduralne logike uz pomoć trigera

### Trigeri za narušavanje 2NF

**INSERT Stavka, u koloni OpisGreske upisuje ono sto pise u Radnom nalogu**  
  
CREATE OR REPLACE TRIGGER INS\_STAVKA\_NALOGA  
BEFORE INSERT ON StavkaRadnogNaloga  
FOR EACH ROW  
DECLARE v\_opis VARCHAR2(50);  
BEGIN  
 SELECT OpisGreske INTO v\_opis FROM RadniNalog WHERE BrojNaloga =: NEW.BrojNaloga;  
 :NEW.OpisGreske := v\_opis;  
END;

**UPDATE OpisGreske u Stavci, zabrana**  
  
CREATE OR REPLACE TRIGGER UPD\_OPIS\_ZABRANA  
BEFORE UPDATE OF OpisGreske ON StavkaRadnogNaloga  
FOR EACH ROW   
BEGIN  
 RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Nije moguce direktno menjati Opis greke u stavci!');  
END;

**UPDATE BrojNaloga u stavci, zabrana**  
  
CREATE OR REPLACE TRIGGER UPD\_BROJNALOGA\_ZABRANA  
BEFORE UPDATE OF BrojNaloga ON StavkaRadnogNaloga  
FOR EACH ROW   
BEGIN  
 RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Nije moguce menjati BrojNaloga u stavci!');  
END;

**UPDATE OpisGreske u Nalogu, automatski se prenosi na stavke tog naloga**  
  
CREATE OR REPLACE TRIGGER UPD\_OPIS\_GRESKE  
AFTER UPDATE OF OpisGreske ON RadniNalog  
FOR EACH ROW  
DECLARE PRAGMA AUTONOMOUS\_TRANSACTION;  
BEGIN  
 EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_OPIS\_ZABRANA DISABLE';  
 UPDATE StavkaRadnogNaloga SET OpisGreske = :NEW.OpisGreske WHERE BrojNaloga = :NEW.BrojNaloga;  
BEGIN  
 EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_OPIS\_ZABRANA ENABLE';  
END;  
END;

### Trigeri za narušavanje 3NF

**INSERT RadniNalog, upisuje u SifraMarke automatdki, u odnosu na SifraUredjaja**  
  
CREATE OR REPLACE TRIGGER INS\_RADNINALOG  
BEFORE INSERT ON RadniNalog  
FOR EACH ROW  
DECLARE v\_marka NUMBER(10);  
BEGIN  
 SELECT SifraMarke INTO v\_marka FROM Uredjaj WHERE SifraUredjaja = :NEW.SifraUredjaja;  
 :NEW.SifraMarke := v\_marka;  
END;

**UPDATE SifraUredjaja u Radnom nalogu, unosi se nova SifraMarke u odnosu na SifruUredjaja**  
  
CREATE OR REPLACE TRIGGER UPD\_SIFRAUREDJAJA  
BEFORE UPDATE OF SifraUredjaja ON RadniNalog  
FOR EACH ROW  
DECLARE v\_marka1 NUMBER(10);  
BEGIN  
 SELECT SifraMarke INTO v\_marka1 FROM Uredjaj WHERE SifraUredjaja = :NEW.SifraUredjaja;  
 :NEW.SifraMarke := v\_marka1;  
END;

**UPDATE SifraMarke u Nalogu, ne moze direktno da se menja**  
  
CREATE OR REPLACE TRIGGER UPD\_SIFRAMARKE\_ZABRANA  
BEFORE UPDATE OF SifraMarke ON RadniNalog  
FOR EACH ROW   
BEGIN  
 RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Nije moguca direktno menjati SifruMarke!');  
END;

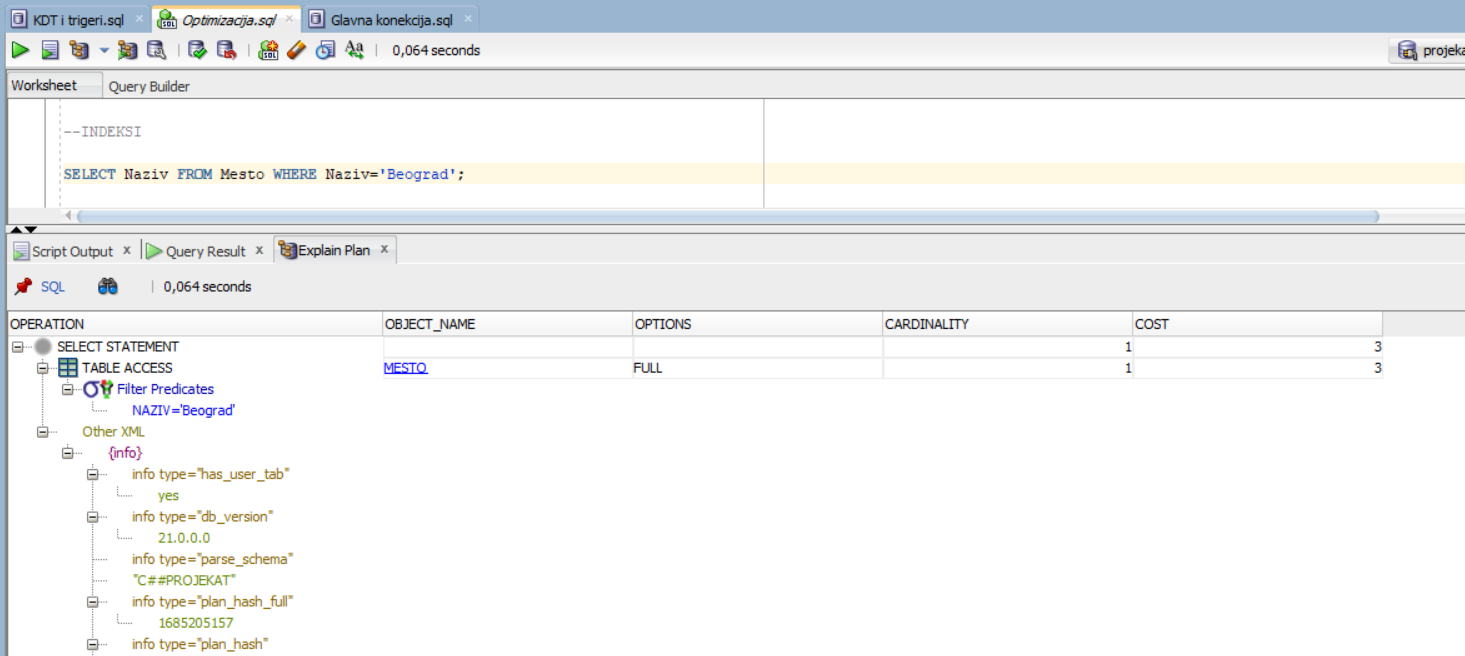
**UPDATE SifraMarke u Uredjaju, da ukoliko Uredjaj koji se menja ima radne naloge i kod njih se menja SifraMarke**  
  
CREATE OR REPLACE TRIGGER UPD\_MARKAUREDJAJ  
AFTER UPDATE OF SifraMarke ON Uredjaj  
FOR EACH ROW  
DECLARE PRAGMA AUTONOMOUS\_TRANSACTION;  
BEGIN  
 EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_SIFRAMARKE\_ZABRANA DISABLE';  
 UPDATE RadniNalog SET SifraMarke = :NEW.SifraMarke WHERE SifraUredjaja = :NEW.SifraUredjaja;  
BEGIN  
 EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_SIFRAMARKE\_ZABRANA ENABLE';  
END;  
END;

# OPTIMIZACIJA BAZE PODATAKA

## Definisanje i analiza indeksa

**Definisanje indeksa nad tekstualnim poljem**

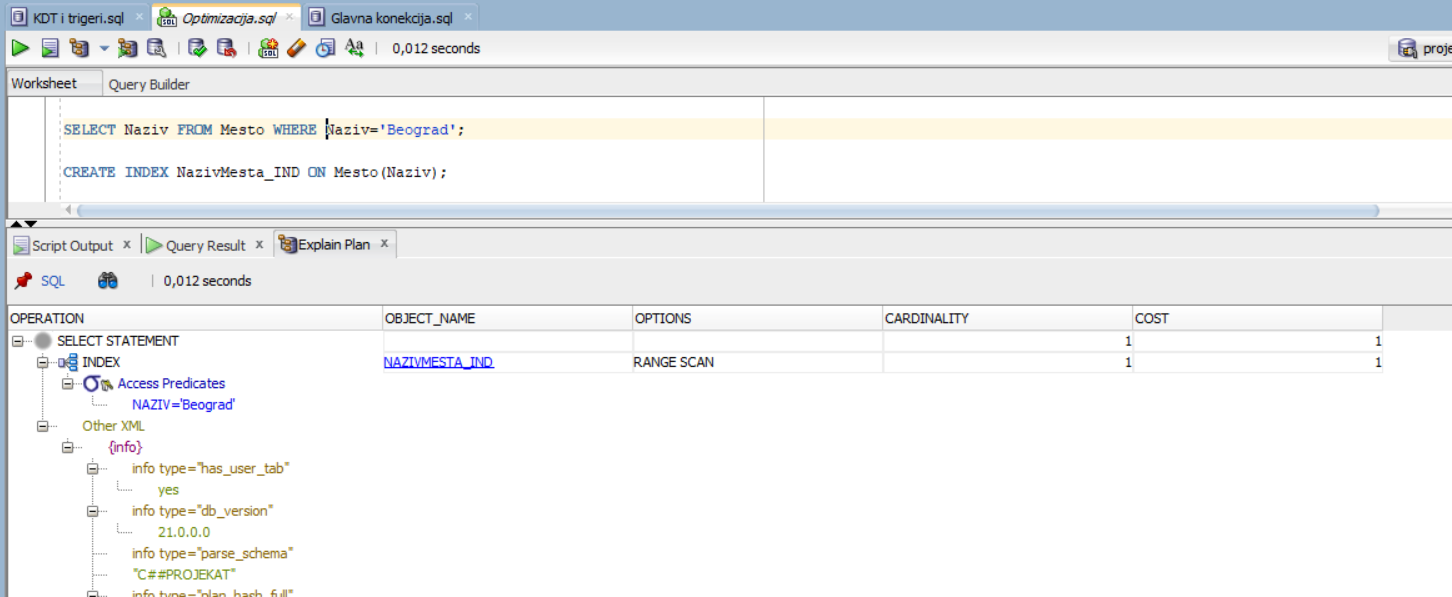
Plan izvršenja pre definisanja indeksa:

****

Definisanje indeksa:

CREATE INDEX NazivMesta\_IND ON Mesto(Naziv);

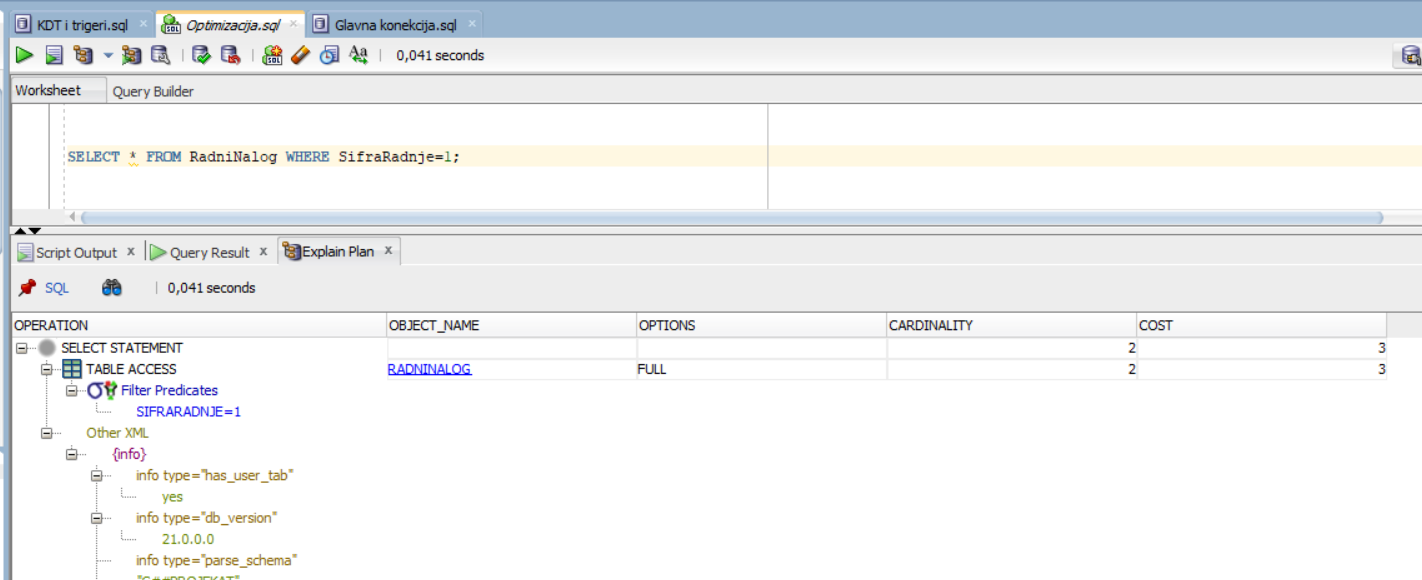
Plan izvršenja nakon definisanja indeksa:



Ovaj indeks ubrzava upite koji u WHERE kluzuli imaju kolonu Naziv iz tabele Mesto.

**Definisanje indeksa nad spoljnim ključem**

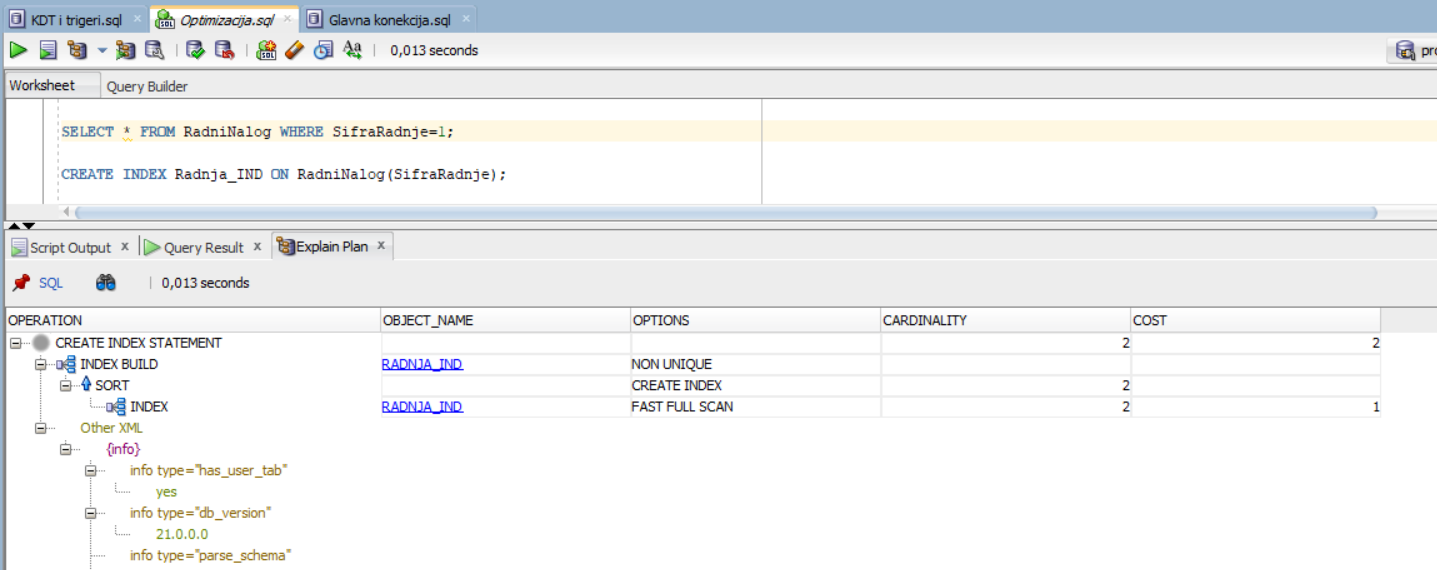
Plan izvršenja pre definisanja indeksa:



Definisanje indeksa:

CREATE INDEX Radnja\_IND ON RadniNalog(SifraRadnje);

Plan izvršenja nakon definisanja indeksa:



## 

Ovaj indeks ubrzava upite koji u WHERE kluzuli imaju kolonu SifraRadnje iz tabele Radnja. Ubrzani su upiti ako na primer proveravamo, koja radnja ima najvise posla.

## Horizontalno particionisanje

Horizontalno particionisanje je odrađeno na tabeli RadniNalog, gde je izvršena podela na dve particije koje se odnose na poslovanje radnji po godinama na osnovu kolone DatumIzdavanja.

ALTER TABLE RadniNalog  
MODIFY PARTITION BY RANGE (DatumIzdavanja) (  
PARTITION pruzene\_usluge\_2022 VALUES LESS THAN (TO\_DATE('01/01/2023', 'MM/DD/YYYY')),  
PARTITION pruzene\_usluge\_2023 VALUES LESS THAN (TO\_DATE('01/01/2024', 'MM/DD/YYYY'))  
);

**Naredba za prikaz pruzenih usluga u 2022. godino:**

SELECT \* FROM RadniNalog PARTITION(pruzene\_usluge\_2022);

## Vertikalno particionisanje

**Skladište** (ŠifraSkladišta, PIB, NazivFirme, Naziv, Adresa, *ŠifraMesta*)  
**SkladišteOsnovno** (ŠifraSkladišta, Naziv, Adresa, *ŠifraMesta*)  
**SkladišteDetalji** (ŠifraSkladišta, PIB, NazivFirme)

**Kreiranje pogleda**

CREATE OR REPLACE VIEW Skladiste AS   
SELECT so.SifraSkladista, so.Naziv, so.Adresa, so.SifraMesta, sd. PIB, sd.NazivFirme  
FROM SkladisteOsnovno so, SkladisteDetalji sd   
WHERE so.SifraSkladista=SD.SifraSkladista;

**Kreiranje INSTEAD OF trigera za sve naredbe**

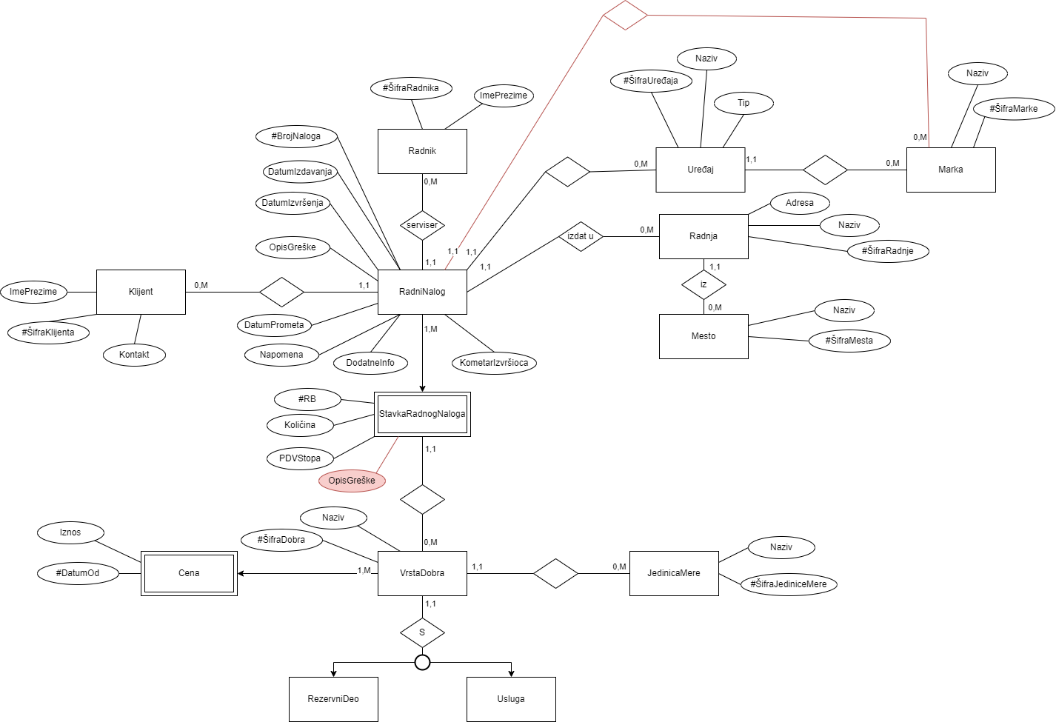
CREATE OR REPLACE TRIGGER INSTEAD\_INS\_SKLADISTE  
INSTEAD OF INSERT ON Skladiste  
FOR EACH ROW  
BEGIN  
 INSERT INTO SkladisteOsnovno(SifraSkladista, Naziv, Adresa, SifraMesta)  
 VALUES (:NEW.SifraSkladista, :NEW.Naziv, :NEW.Adresa, :NEW.SifraMesta);  
 INSERT INTO SkladisteDetalji(SifraSkladista, PIB, NazivFirme)  
 VALUES (:NEW.SifraSkladista, :NEW.PIB, :NEW.NazivFirme);  
END;

CREATE OR REPLACE TRIGGER INSTEAD\_UPD\_SKLADISTE  
INSTEAD OF UPDATE ON Skladiste  
FOR EACH ROW  
BEGIN  
 UPDATE SkladisteOsnovno   
 SET Naziv = :NEW.Naziv, Adresa = :NEW.Adresa, SifraMesta = :NEW.SifraMesta  
 WHERE SifraSkladista = :NEW.SifraSkladista;  
 UPDATE SkladisteDetalji   
 SET PIB = :NEW.PIB, NazivFirme = :NEW.NazivFirme  
 WHERE SifraSkladista = :NEW.SifraSkladista;  
END;

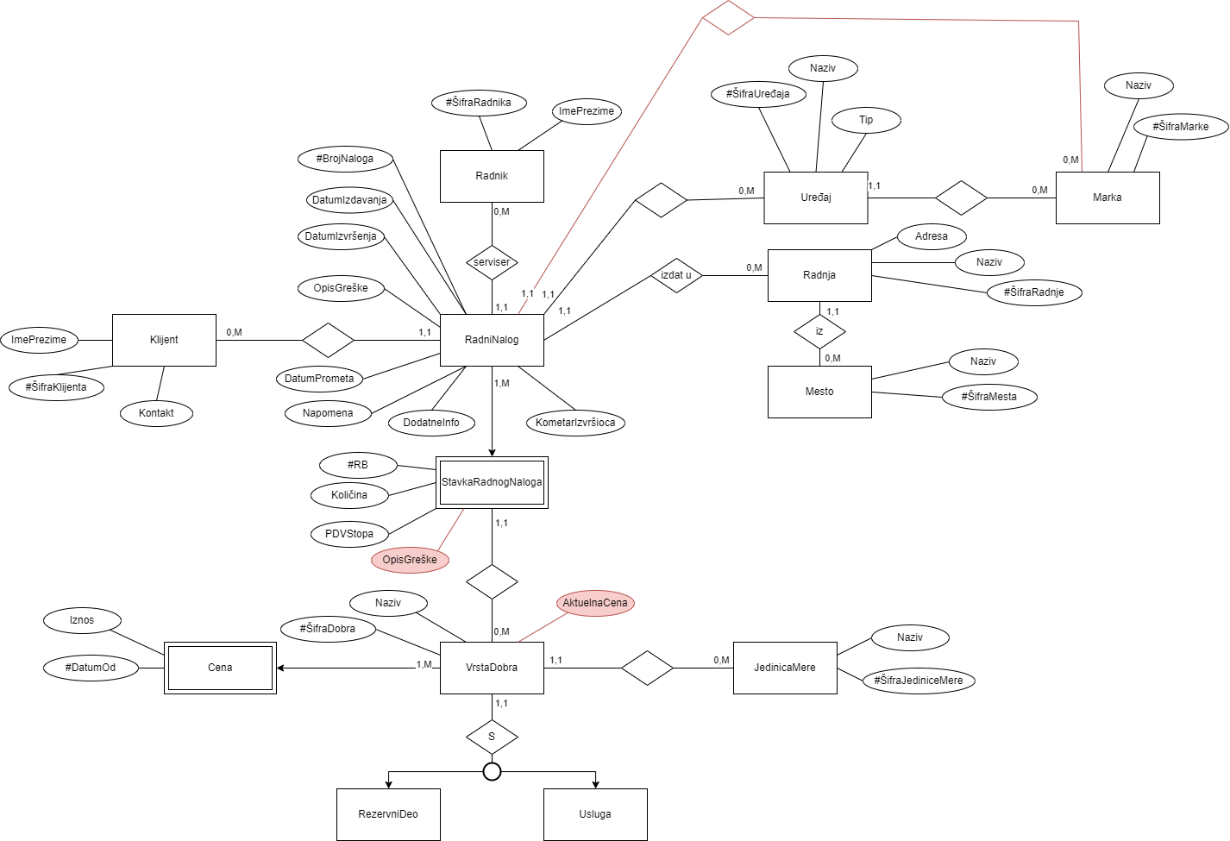
CREATE OR REPLACE TRIGGER INSTEAD\_DEL\_SKLADISTE  
INSTEAD OF DELETE ON Skladiste  
FOR EACH ROW  
BEGIN  
 DELETE FROM SkladisteOsnovno WHERE SifraSkladista = :OLD.SifraSkladista;  
 DELETE FROM SkladisteDetalji WHERE SifraSkladista = :OLD.SifraSkladista;  
END;

## Primena drugih optimizacionih tehnika

**Repeating Single Detail with Master**U ovom primeru primenjena je Repeating Single Detail with Mastertehnika optimizacije tako što je tabeli VrstaDobra dodat atribut AktuelnaCena kako bismo pratili njenu promenu kroz vreme. Na ovj način izbegli smo spajanje tabela kod upita kojima je potrebna samo specifična vrednost odnoso aktuelna cena robe. **Pre optimizacije:**

****

**Posle optimizacije:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba uraditi?** |
| **VrstaDobra** | insert |  | NE |  |
| update | AktuelnaCena | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |
| **Cena** | insert |  | DA | Okida se triger koji poziva proceduru koja ažurira vrednost kolone AktuelnaCena u tabeli VrstaDobra |
| update | Iznos | DA |
| delete |  | DA |

**Triger koji odredjuje sifru dobra i cuva u globalnu promenljivu**

CREATE OR REPLACE NONEDITIONABLE PACKAGE VrstaDobraPaket AS   
SifraDobra NUMBER(10);  
END VrstaDobraPaket;

CREATE OR REPLACE TRIGGER VRSTADOBRA\_CENA  
BEFORE INSERT OR UPDATE OR DELETE ON Cena  
FOR EACH ROW  
BEGIN  
 IF (INSERTING OR UPDATING)  
 THEN  
 BEGIN VrstaDobraPaket.SifraDobra := :NEW.SifraDobra; END;  
 ELSE  
 BEGIN VrstaDobraPaket.SifraDobra := :OLD.SifraDobra; END;  
 END IF;  
END;

**Triger koji sprecava direktnu izmenu AktuelneCene u tabeli VrstaDobra**

CREATE OR REPLACE TRIGGER UPD\_AKTUELNA\_CENA\_ZABRANA  
BEFORE UPDATE OF AktuelnaCena ON VrstaDobra  
FOR EACH ROW  
BEGIN  
 RAISE\_APPLICATION\_ERROR(NUM => -20000,MSG => 'Ne mozete direktno menjati vrednost AktuelnaCena!');  
END;

**Procedura koja racuna aktuelnu cenu robe**

CREATE OR REPLACE PROCEDURE AKTUELNA\_CENA\_DOBRA (SDobra IN NUMBER) AS  
Aktuelna VrstaDobra.AktuelnaCena%type :=0;  
BEGIN  
 BEGIN  
 SELECT Iznos INTO Aktuelna   
 FROM Cena  
 WHERE SifraDobra = SDobra AND DatumOd=(SELECT MAX(DatumOd) FROM Cena WHERE SifraDobra = SDobra AND DatumOd <= SYSDATE);  
 EXCEPTION WHEN no\_data\_found THEN Aktuelna := null;  
 END;  
 UPDATE VrstaDobra  
 SET AktuelnaCena = Aktuelna  
 WHERE SifraDobra = SDobra;  
END;

**Procedure koje iskljucuju i ukljucuju triger koji vrsi zabranu izmene aktuelne cene, da bi se promena prenela, ukoliko se izmena izvrsi u tabeli Cena**

CREATE OR REPLACE PROCEDURE ISKLJUCI\_TRIGER\_CENA AS   
PRAGMA AUTONOMOUS\_TRANSACTION;   
BEGIN   
EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_AKTUELNA\_CENA\_ZABRANA DISABLE';   
END ISKLJUCI\_TRIGER\_CENA;

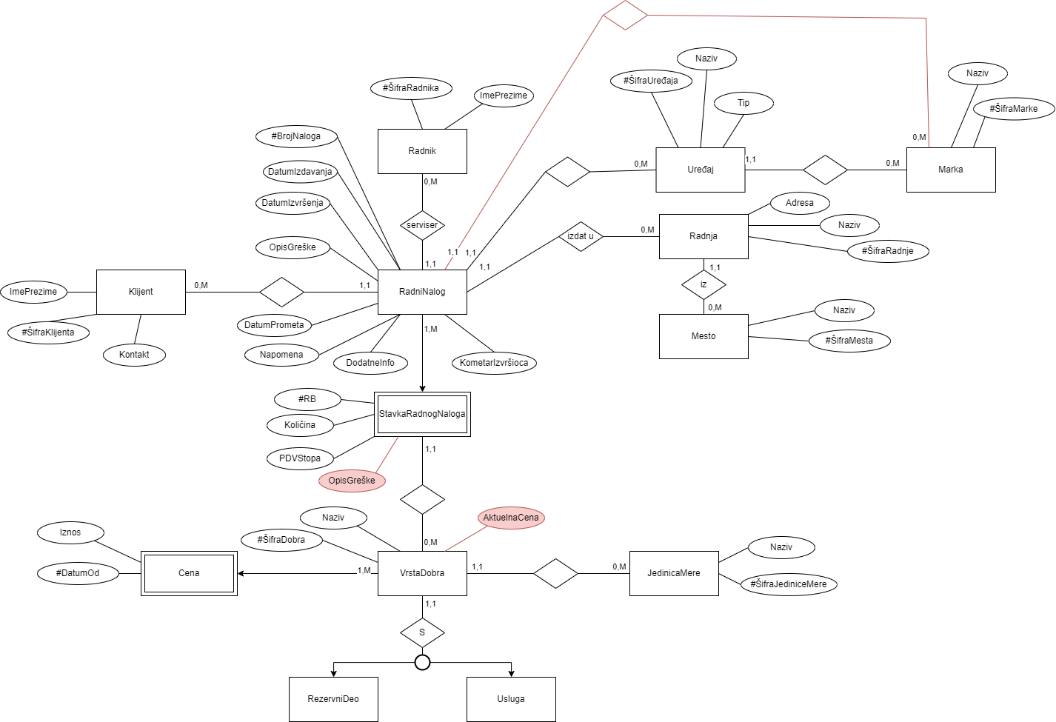
CREATE OR REPLACE PROCEDURE UKLJUCI\_TRIGER\_CENA AS   
PRAGMA AUTONOMOUS\_TRANSACTION;   
BEGIN   
EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_AKTUELNA\_CENA\_ZABRANA ENABLE';   
END UKLJUCI\_TRIGER\_CENA;

**Triger koji poziva proceduru koja azurira aktuelnu cenu**

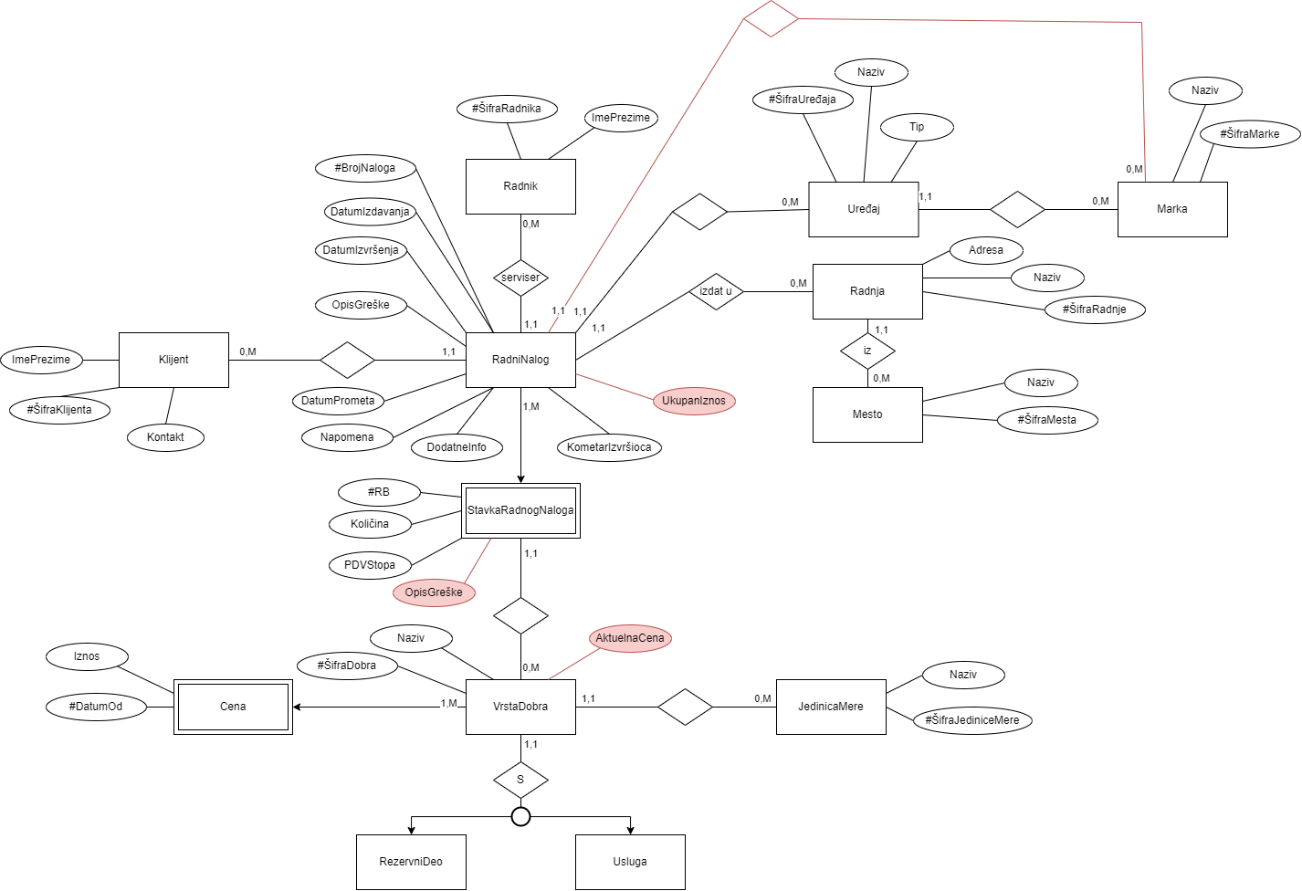
CREATE OR REPLACE TRIGGER ROBA\_CENA2  
AFTER INSERT OR UPDATE OR DELETE ON Cena  
DECLARE  
 SDobra NUMBER(10) := VrstaDobraPaket.SifraDobra;  
BEGIN  
 ISKLJUCI\_TRIGER\_CENA();  
 AKTUELNA\_CENA\_DOBRA(SDobra);  
 UKLJUCI\_TRIGER\_CENA();  
END;

**Storing Derivable Values**U prvom primeru primenjena je Storing Derivable Values tehnika optimizacije tako što je dodat atribut UkupanIznos u tabelu RadniNalog koji predstavlja ukupan iznos svih stavki. Na ovaj način omogućeno je da ne moramo pristupati osnovnim podacima ukoliko nam je potrebna izvedena vrednost.

**Pre optimizacije:**

****

**Posle optimizacije:**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba uraditi?** |
| **RadniNalog** | insert |  | NE |  |
| update | UkupanIznos | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |
| **StavkaRadnogNaloga** | insert |  | DA | Okida se triger koji poziva proceduru koja ažurira vrednost kolone UkupanIznos u tabeli RadniNalog |
| update | Kolicina | DA |
| delete |  | DA |

**Triger koji cuva BrojNaloga u globalnu promenljivu**

CREATE OR REPLACE NONEDITIONABLE PACKAGE StavkaPaket AS   
BrojNaloga NUMBER(10);  
END StavkaPaket;

CREATE OR REPLACE TRIGGER UKUPAN\_IZNOS\_NALOG  
BEFORE INSERT OR UPDATE OR DELETE ON StavkaRadnogNaloga   
FOR EACH ROW  
BEGIN  
 IF (INSERTING OR UPDATING)  
 THEN  
 BEGIN StavkaPaket.BrojNaloga := :NEW.BrojNaloga; END;  
 ELSE  
 BEGIN StavkaPaket.BrojNaloga := :OLD.BrojNaloga; END;  
 END IF;  
END;

**Triger koji sprecava direktnu izmenu Ukupnogiznosa u tabeli RadniNalog**

CREATE OR REPLACE TRIGGER UPD\_UKUPNO\_ZABRANA  
BEFORE UPDATE OF UkupanIznos ON RadniNalog  
FOR EACH ROW  
BEGIN RAISE\_APPLICATION\_ERROR(NUM => -20000,MSG => 'Ne mozete direktno menjati vrednost UkupanIznos!');  
END;

**Procedura koja racuna ukupan iznos radnog naloga**

CREATE OR REPLACE PROCEDURE UKUPAN\_IZNOS\_SUM (BrNal IN NUMBER) AS  
Ukupno VrstaDobra.AktuelnaCena%type :=0;  
BEGIN  
 SELECT SUM(vd.AktuelnaCena\*s.Kolicina+vd.AktuelnaCena\*s.PDVStopa\*0.01) INTO  
Ukupno   
 FROM StavkaRadnogNaloga s JOIN VrstaDobra vd ON (s.SifraDobra=vd.SifraDobra)  
 WHERE s.BrojNaloga = BrNal;  
 UPDATE RadniNalog  
 SET UkupanIznos = Ukupno  
 WHERE BrojNaloga = BrNal;  
END;

**Procedure koje iskljucuju i ukljucuju triger koji vrsi zabranu izmene ukupne cene, da bi se promena prenela, ukoliko se izmena izvrsi u tabeli Stavke, tj ako se promeni kolicina**

CREATE OR REPLACE PROCEDURE ISKLJUCI\_TRIGER\_UKUPNO AS   
PRAGMA AUTONOMOUS\_TRANSACTION; BEGIN   
EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_UKUPNO\_ZABRANA DISABLE';   
END ISKLJUCI\_TRIGER\_UKUPNO;

CREATE OR REPLACE PROCEDURE UKLJUCI\_TRIGER\_UKUPNO AS   
PRAGMA AUTONOMOUS\_TRANSACTION;   
BEGIN   
EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_UKUPNO\_ZABRANA ENABLE';   
END UKLJUCI\_TRIGER\_UKUPNO;

**Triger koji poziva proceduru koja azurira ukupanIznos**

CREATE OR REPLACE TRIGGER UKUPAN\_IZNOS\_NALOG2  
AFTER INSERT OR UPDATE OR DELETE ON StavkaRadnogNaloga  
DECLARE  
 BrojNaloga NUMBER(10) := StavkaPaket.BrojNaloga;  
BEGIN  
 ISKLJUCI\_TRIGER\_UKUPNO();  
 UKUPAN\_IZNOS\_SUM(BrojNaloga);  
 UKLJUCI\_TRIGER\_UKUPNO();  
END;

# TEHNOLOGIJE ZA IMPLEMENTACIJU PROJEKTA

## SUBP korišćen za implementaciju baze podatka (kratak opis)

Za implementaciju baze podataka korišćen je sistem za upravljanje bazama podataka kompanije Oracle. Oracle SUBP je sistem za upravljanje relacionim bazama podataka koji se koristi za rad sa transakcijama (OLTP) i skladištenje podataka (DW) kao i njihove kombinacije. Za implementaciju baze podataka koja je prikazana u ovom radu koršćena je verzija Oracle 19c.

Razvojno okruženje koje je korišćeno u ovom radu je SQL Developer takođe kompanije Oracle. To je integrisano razvojno okruženje koje pojednostavljuje rad i upravljanje Oracle bazama podataka i omogućava kompletan razvoj PL/SQL aplikacija. On predstavlja grafički korisnički interfejs koji omogućava izvršavanje različitih upita i skripti, prikazivanje izveštaja, modelovanje podataka i tome slično.

## Programsko okruženje za razvoj korisničkog interfejsa ( kratak opis)

Za implemetaciju korisničkog interfejsa korišćeno je Netbeans IDE razvojno okruženje i programski jezik Java. Netbeans je integrisano razvojno okruženje otvorenog koda za razvoj aplikacije u Windows, Mac, Linux i Solaris operativnim sistemima. On pojednostavljuje razvoj veb, poslovnih, desktop i mobilnih aplikacija. Za sam razvoj aplikacije korišćen je MVC patern koji se zasniva na ideji da postoje tri dela softvera i to model podataka, poglede i kontrolera pri čemu je komponenta za prikaz podataka odvojena od interakcije korisnika sa tim informacijama.