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

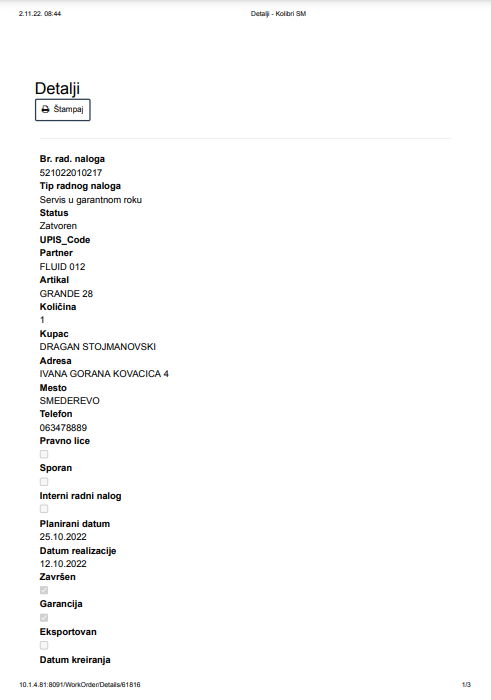
|  |  |  |  |
| --- | --- | --- | --- |
| **IME I PREZIME** | | **Anđela Kostić** | |
| Broj indeksa | | **2022/3010** | |
| Studijski modul | | Informacioni sistemi | |
| Stečena diploma osnovnih akademskih studija | | Diplomirani inženjer organizacionih nauka | |
| **NAZIV PROJEKTA** | | Informacioni sistem za servisiranje gotovih proizvoda | |
| Napomena | |  | |
| *Verzija* | *Datum promene* | | *Opis promene* | |
| 1 | 07.11.2022. | | **Verbalni opis i slike dokumenata.** | |
| 2 | 16.11.2022. | | **Prošireni model objekti – veze i relacioni model.** | |
| 3 | 30.11.2022. | | **Unete izmene za prethodnu fazu. Denormalizacija relacija.** | |
| 4 | 07.12.2022. | | **Korisnički definisani tipovi. Implementacija trigera.** | |
| 5 | 13.12.2022. | | **Indeksi. Particionisanje tabela. Logička optimizacija modela.** | |
| 6 | 10.01.2023. | | **Implementacija projekta.** | |

# OPIS PROJEKTA

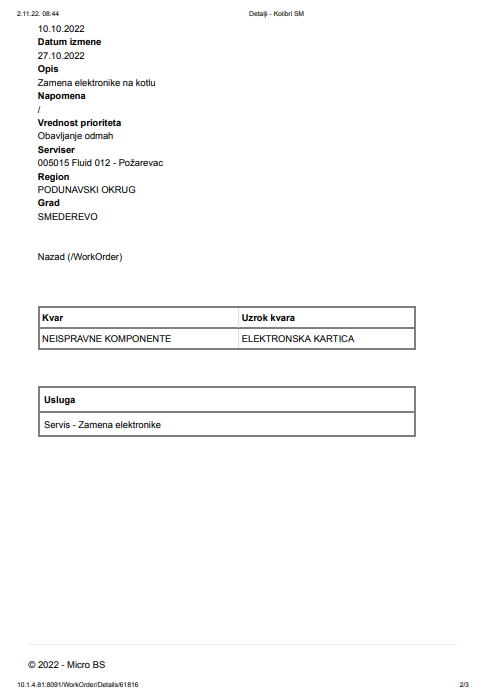
## Opis podsistema za koji se projektuje baza podataka

|  |
| --- |
| Potrebno je kreirati informacioni sistem za servisiranje gotovih proizvoda u kompaniji Alfa-Plam iz Vranja.  Najpre se radi godišnji ugovor sa eksternim serviserima koji obezbeđuju uslugu servisiranja gotovih proizvoda Alfa-Plama koji se nalaze u upotrebi kod krajnjeg potrošača, a po nalogu tehničke podrške (call centra) Alfa-Plama.  Proces servisiranja počinje prijemom reklamacije u call centru (telefonom, mejlom) i u tom trenutku se zavodi u bazi podataka Kolibri. Zatim se radni nalog dodeljuje serviseru (u ovom slučaju Fluid 012 Požarevac). Nakon toga serviser otvara sebi radni nalog i popunjava ga i potpisuje zajedno sa strankom gde je izvršeno servisiranje, gde konstatuju šta je urađeno. Na osnovu radnog naloga i zapisa kod stranke, gde mu i stranka potpisuje zapisnik, serviser radi izveštaj u Kolibriju. Isti se šalje call centru gde se vrši pregled svih podataka iz radnog naloga i fakture. Ukoliko je sve u redu izrađuje se prijemnica tog naloga koji je prihvaćen u virtuelnom magacinu servisera. Na osnovu prijemnice radi se trebovanje čime se pravda potrošnja tog dela.  Na osnovu svih prethodnih aktivnosti i dokumenata, serviser radi račun, odnosno otpremnicu ka Alfa-Plamu. Račun sadrži utrošeno vreme po radnom nalogu, troškove prevoza i količinu i vrednost delova koji su zamenjeni. Tako se dolazi do ukupne vrednosti fakture koju kompanija Alfa-Plam treba da plati. Potpisana faktura sa pratećom dokumentacijom (radni nalog, prijemnica) šalje se službi finansija na plaćanje.  U ovom podsistemu koriste se sledeći dokumenti:   1. Radni nalog 2. Terenski radni nalog 3. Izveštaj servisera 4. Prijemnica 5. Trebovanje 6. Račun – otpremnica   Postupak metodologije projektovanja baze podsistema koja će biti implementirana u ovom radu podrazumeva:   1. Modelovanje sistema u proširenom modelu objekti – veze 2. Prevođenje proširenog modela objekti – veze u relacioni model 3. Denormalizacija relacija uz očuvanje integriteta 4. Uvođenje korisnički definisanih tipova i trigera 5. Optimizacija baze podataka korišćenjem izvedenih tipova, indeksa i particionisanjem tabela |

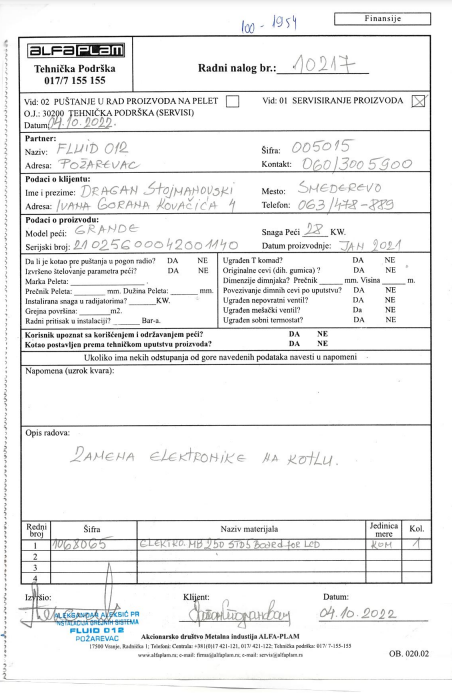
## Poslovna dokumenta koja se koriste u procesu



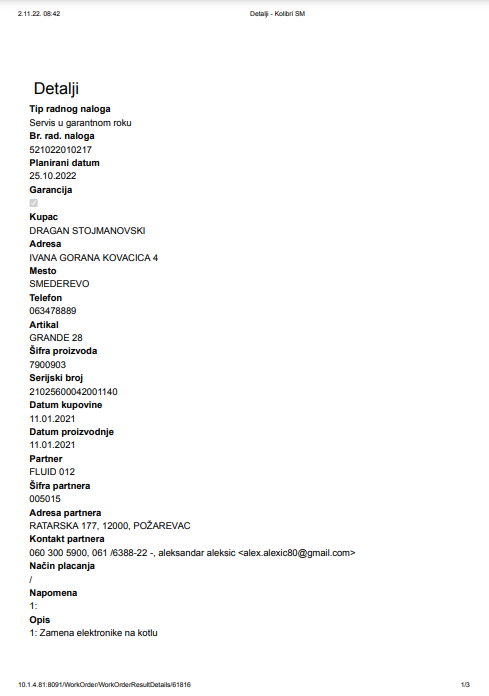
Slika Radni nalog



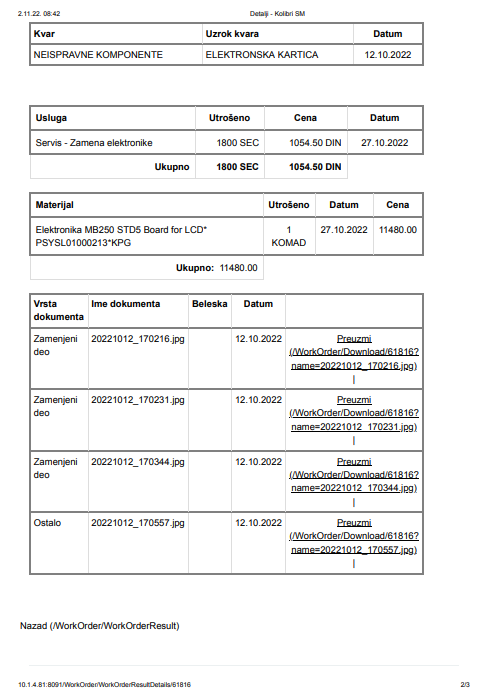
Slika Radni nalog



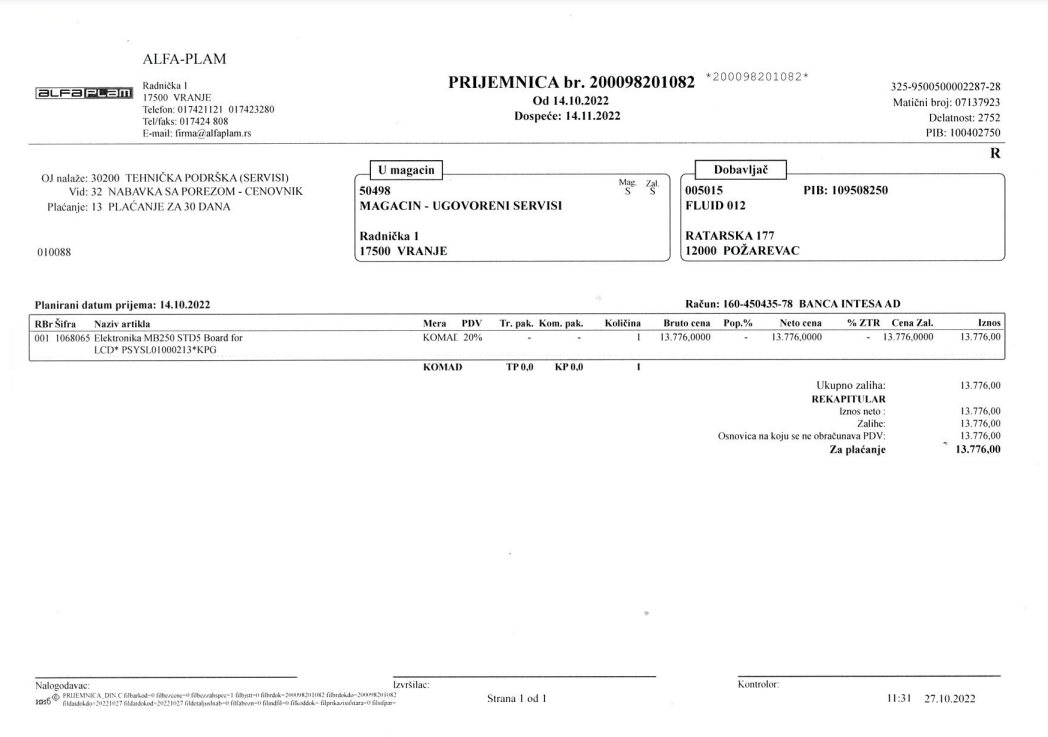
Slika 3 Potpisan radni nalog na terenu



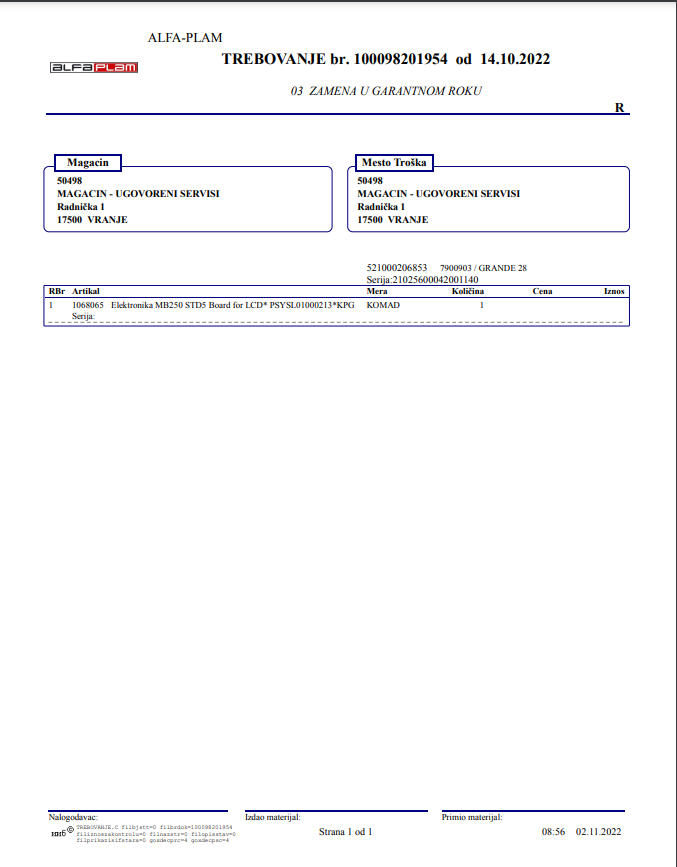
Slika Izveštaj servisera



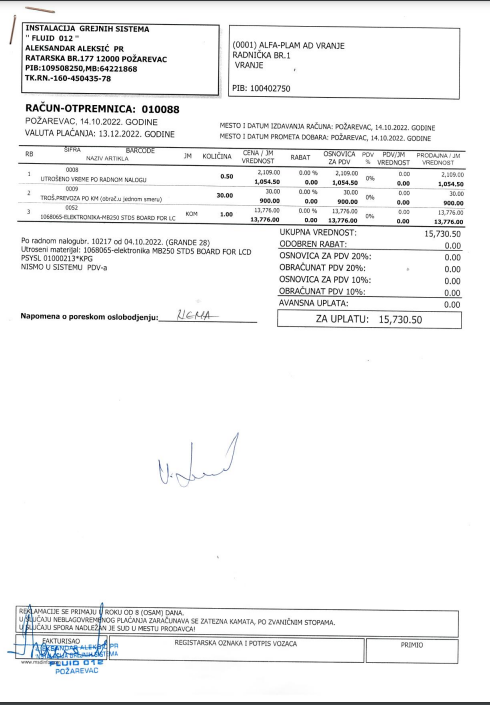
Slika Izveštaj servisera



Slika Prijemnica



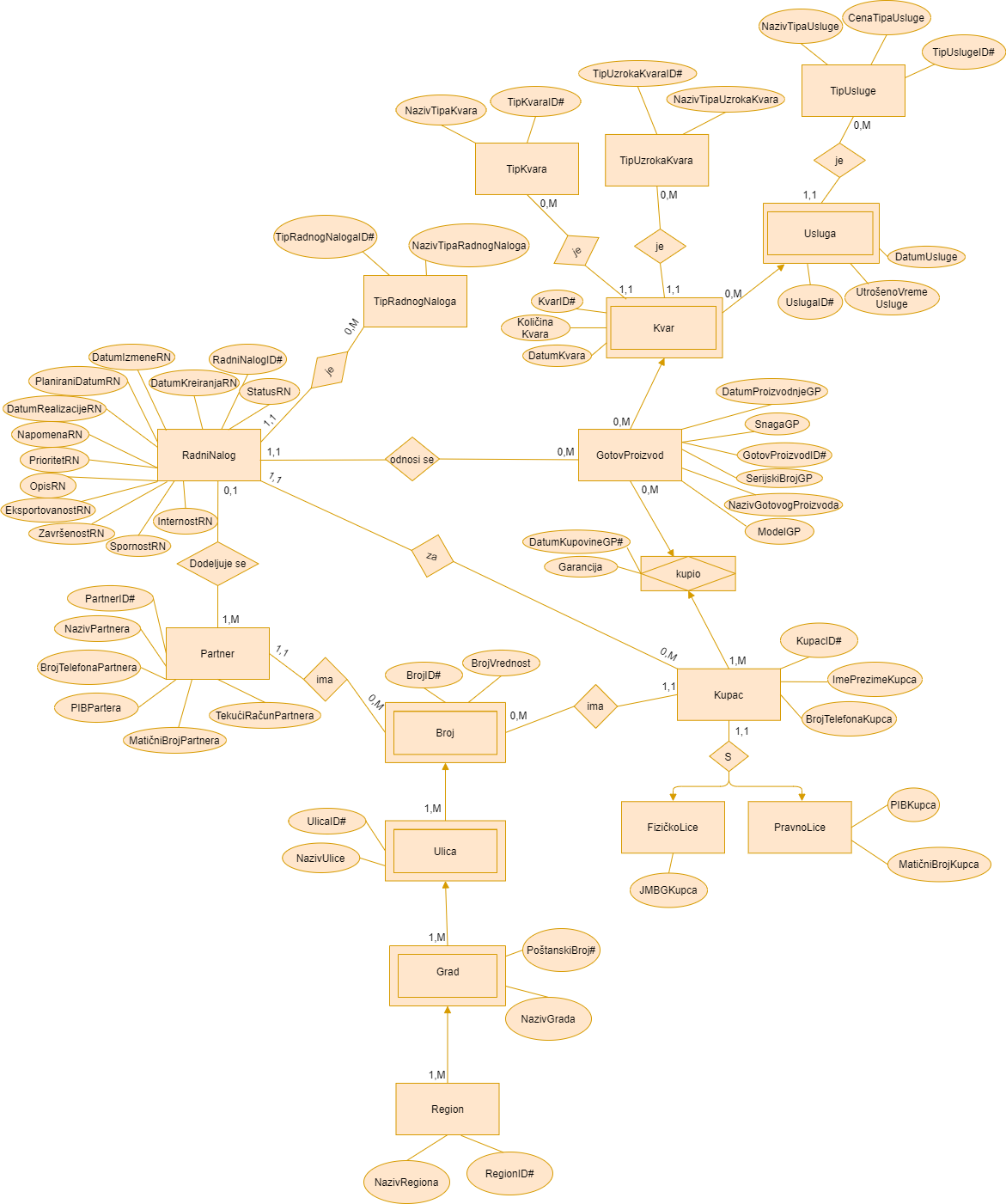
Slika Trebovanje



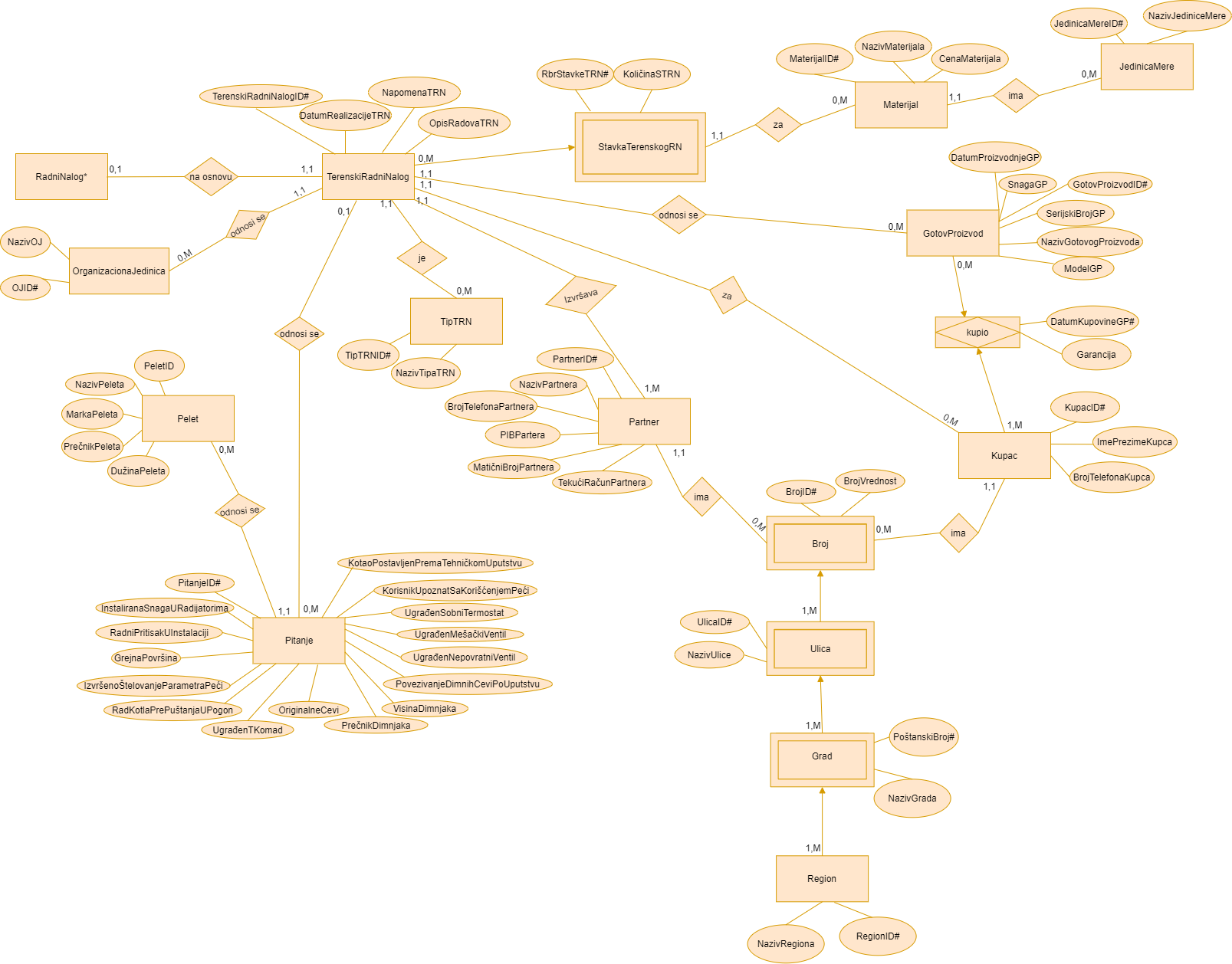
Slika Račun-otpremnica

# PROJEKTOVANJE BAZE PODATAKA

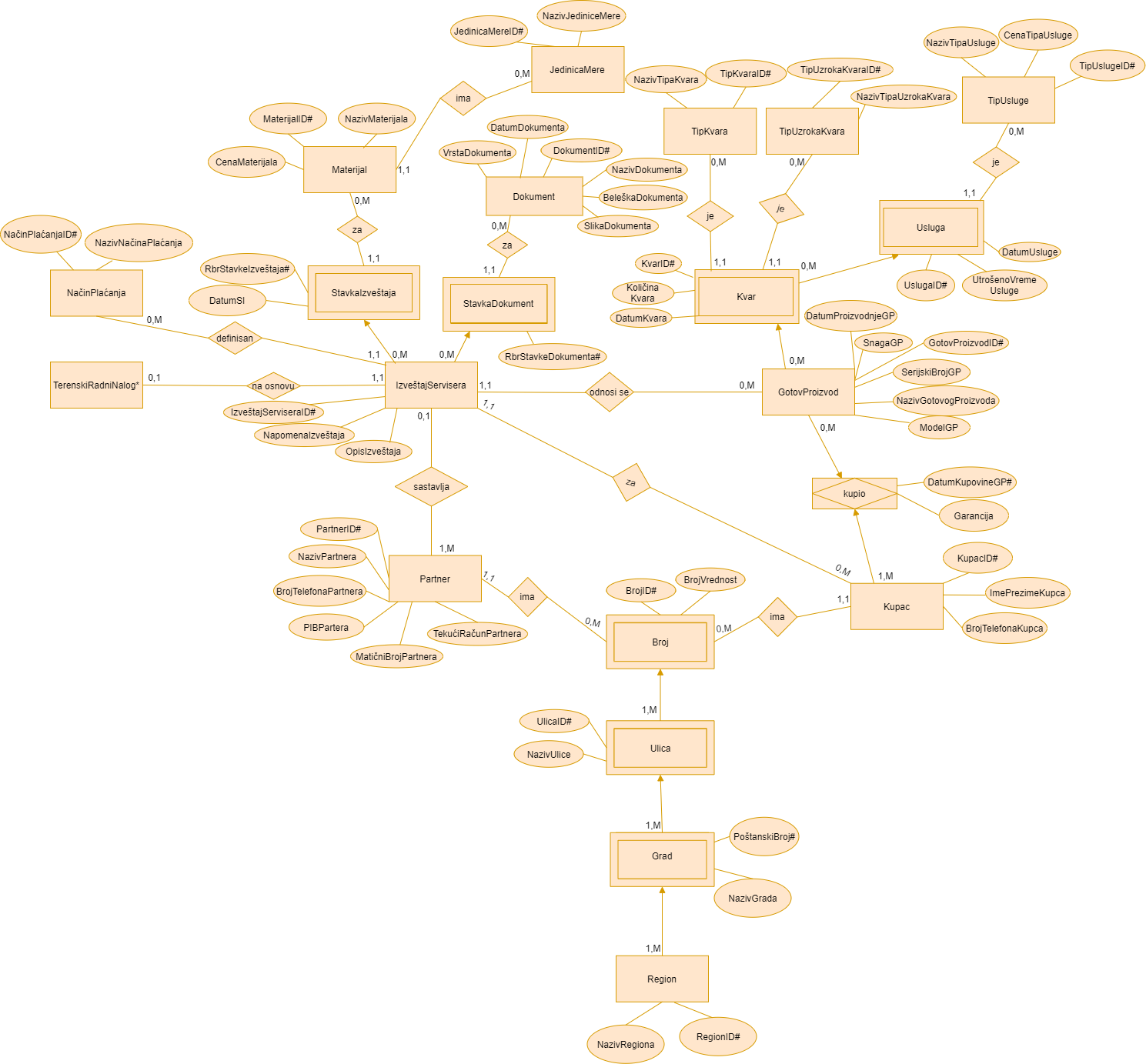
## Model podatka



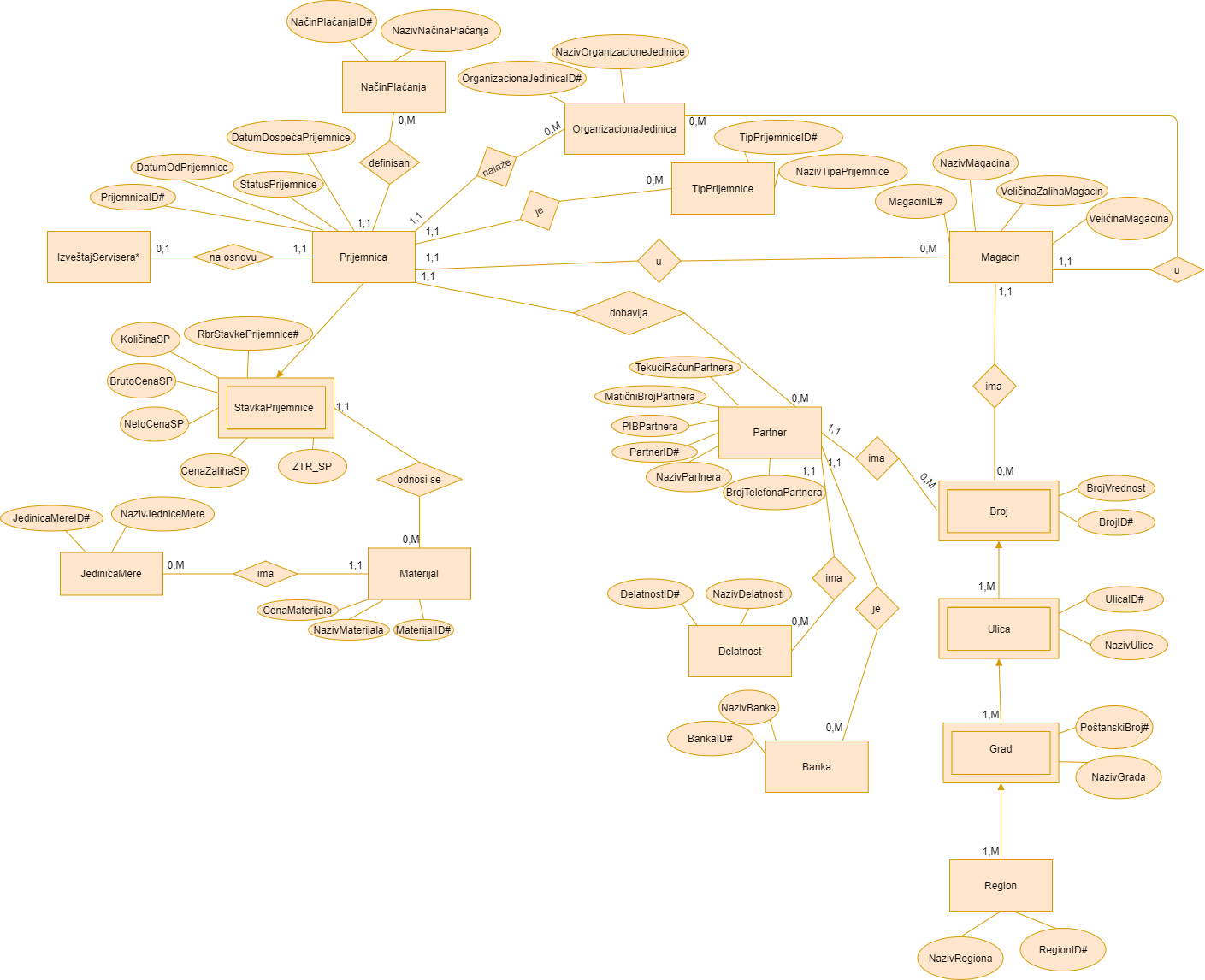
Slika PMOV Radni Nalog



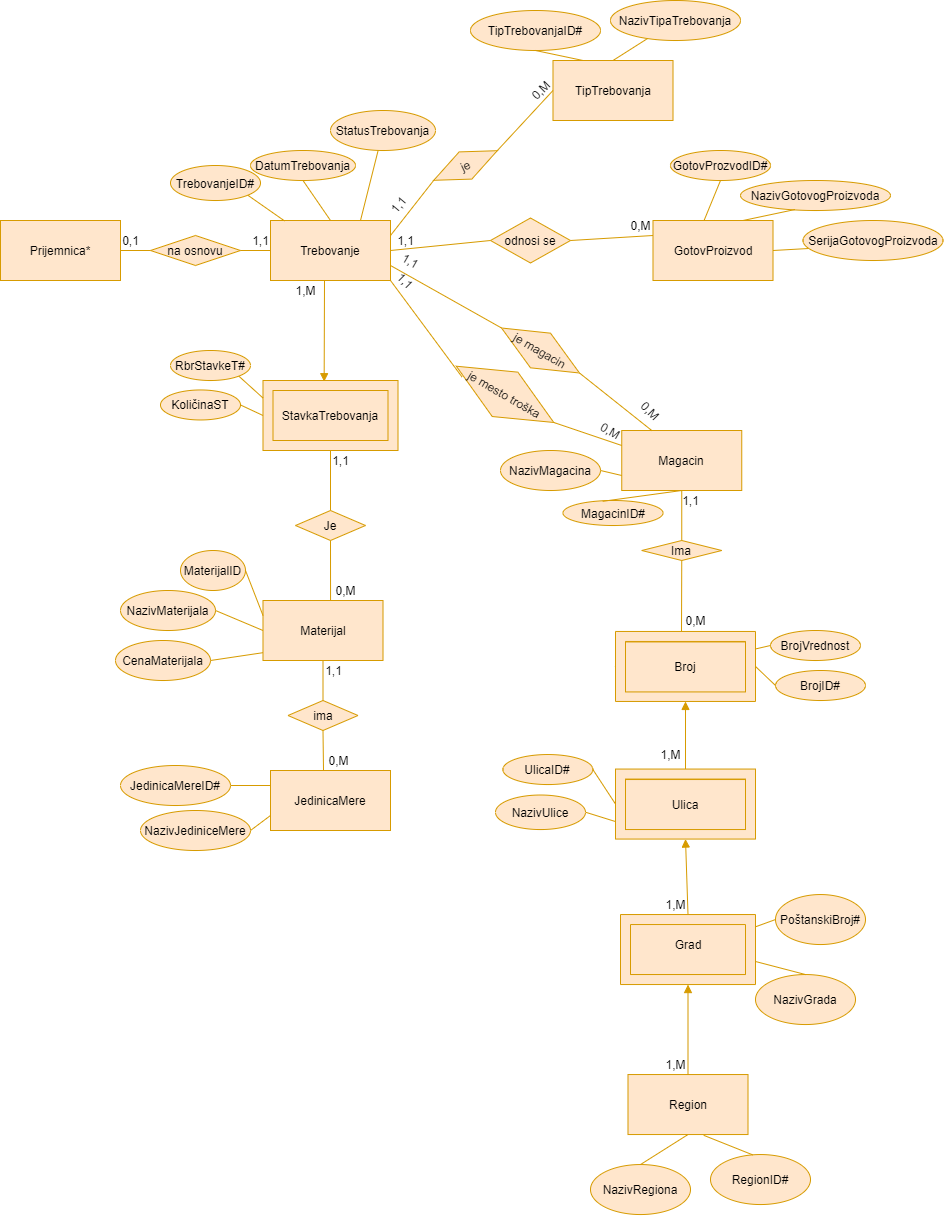
Slika 10 PMOV Terenski radni nalog



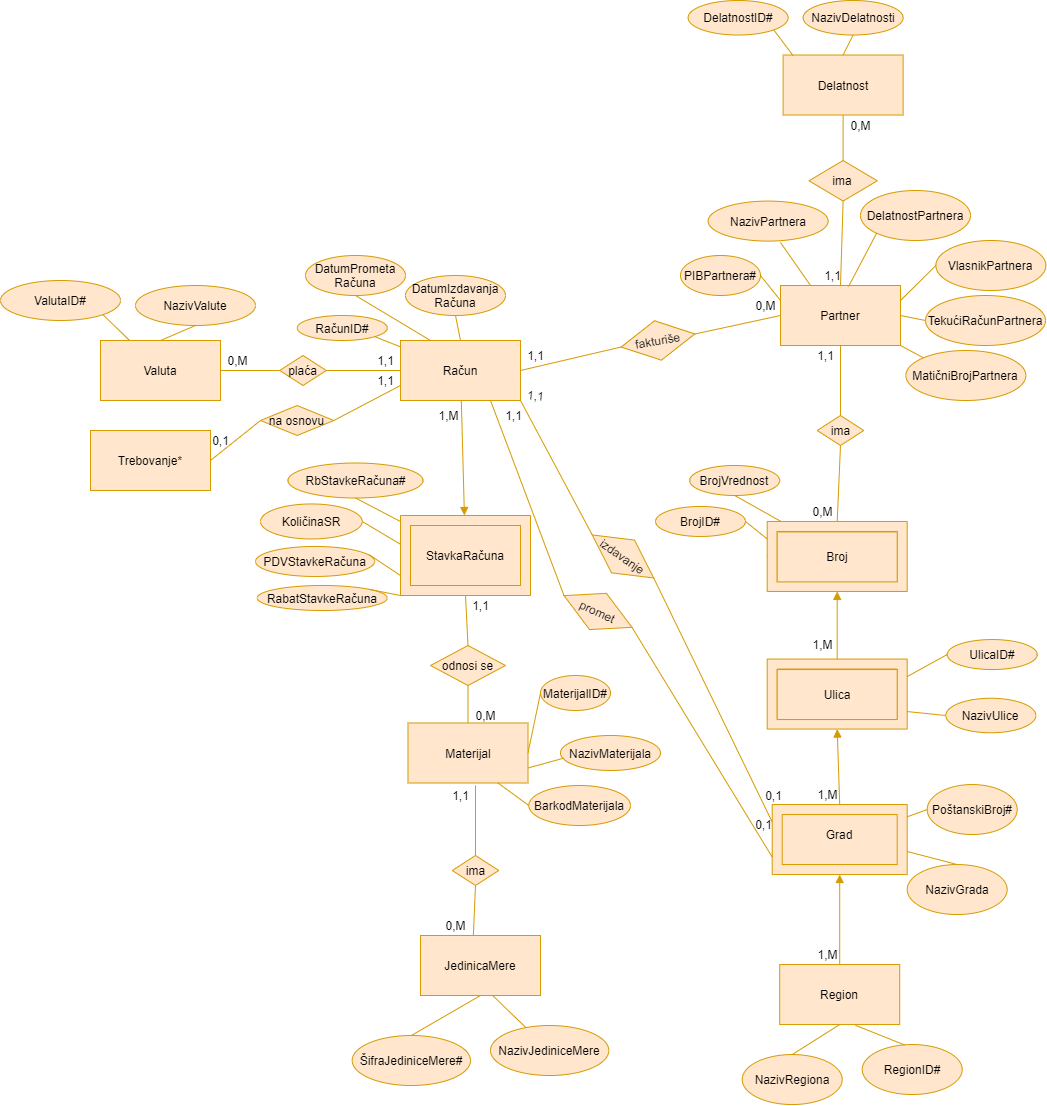
Slika PMOV Izveštaj servisera



Slika PMOV Prijemnica



Slika PMOV Trebovanje



Slika PMOV Račun - otpremnica

## Relacioni model

**RadniNalog** (RadniNalogID, StatusRadnogNaloga, DatumKreiranjaRadnogNaloga, DatumIzmeneRadnogNaloga, PlaniraniDatumRadnogNaloga, DatumRealizacijeRadnogNaloga, NapomenaRadnogNaloga, PrioritetRadnogNaloga, OpisRadnogNaloga, EksportovanostRadnogNaloga, ZavršenostRadnogNaloga, SpornostRadnogNaloga, InternostRadnogNaloga, *TipRadnogNalogaID, PartnerID, GotovProizvodID, KupacID* )

**TipRadnogNaloga** (TipRadnogNalogaID, NazivRadnogNaloga)

**GotovProizvod** (GotovProizvodID, NazivGotovogProizvoda, SnagaGotovogProizvoda, SerijskiBrojGotovogProizvoda, ModelGotovogProizvoda, DatumProizvodnjeGotovogProizvoda)

**Kvar** (KvarID, *GotovProizvodID*, KoličinaKvara, DatumKvara, *TipKvaraID, TipUzrokaKvaraID*)

**TipKvara** (TipKvaraID, NazivTipaKvara)

**TipUzrokaKvara** (TipUzrokaKvaraID, NazivTipaUzrokaKvara)

**Usluga** (UslugaID, *KvarID*, *GotovProizvodID*, DatumUsluge, UtrošenoVremeUsluge, *TipUslugeID*)

**TipUsluge** (TipUslugeID, NazivTipaUsluge, CenaTipaUsluge)

**Partner** (PartnerID, NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MatičniBrojPartnera, TekućiRačunPartnera, *BrojID,* *UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID*)

**Kupac** (KupacID, ImePrezimeKupca, BrojTelefonaKupca, *BrojID,* *UlicaID, PoštanskiBroj, RegionID*)

**FizičkoLice** (*KupacID*, JMBGKupca)

**PravnoLice** (*KupacID*, PIBKupca, MatičniBrojKupca)

**Region** (RegionID, NazivRegiona)

**Grad** (PoštanskiBroj, *RegionID*, NazivGrada)

**Ulica** (UlicaID, *PoštanskiBroj, RegionID*, NazivUlice)

**Broj** (BrojID, *UlicaID, PoštanskiBroj, RegionID,* BrojVrednost)

**Kupio** (DatumKupovineGotovogProizvoda, *GotovProizvodID, KupacID*, Garancija)

**TerenskiRadniNalog** (TerenskiRadniNalogID, DatumRealizacijeTerenskogRadnogNaloga, NapomenaTerenskogRadnogNaloga, OpisPoslovaTerenskogRadnogNaloga, *RadniNalogID,* *OrganizacionaJedinicaID, GotovProizvodID, KupacID, PartnerID, TipTerenskogRadnogNalogaID, PitanjeID*)

**TipTerenskogRadnogNaloga** (TipTerenskogRadnogNalogaID, NazivTipaTerenskogRadnogNaloga)

**StavkaTerenskogRadnogNaloga** (RbrStavkeTerenskogRadnogNaloga, *TerenskiRadniNalogID*, KoličinaStavkeTerenskogRadnogNaloga, *MaterijalID*)

**OrganizacionaJedinica** (OrganizacionaJedinicaID, NazivOrganizacioneJedinice)

**Materijal** (MaterijalID, NazivMaterijala, CenaMaterijala, BarkodMaterijala, *JedinicaMereID*)

**JedinicaMere** (JedinicaMereID, NazivJediniceMere)

**Pitanje** (PitanjeID, InstaliranaSnagaURadijatorima, RadniPritisakUInstalaciji, GrejnaPovršina, IzvršenoŠtelovanjeParametraPeći, RadKotlaPrePuštanjaUPogon, UgrađenTKomad, OriginalneCevi, PrečnikDimnjaka, VisinaDimnjaka, PovezivanjeDimnihCeviPoUputstvu, UgrađenNepovratniVentil, UgrađenMešačkiVentil, UgrađenSobniTermostat, KorisnikUpoznatSaKorišćenjemPeći, KotaoPostavljenPremaTehničkomUputstvu, *PeletID*)

**Pelet** (PeletID, NazivPeleta, MarkaPeleta, PrečnikPeleta, DužinaPeleta)

**IzveštajServisera** (IzveštajServiseraID, NapomenaIzveštaja, OpisIzveštaja, *TerenskiRadniNalogID, NačinPlaćanjaID, PartnerID, KupacID, GotovProizvodID*)

**StavkaIzveštaja** (RbrStavkeIzveštaja, *IzveštajServiseraID*, DatumStavkeIzveštaja, *MaterijalID*)

**StavkaDokument** (RbrStavkeDokumenta*, IzveštajServiseraID*, *DokumentID*)

**Dokument** (DokumentID, NazivDokumenta, BeleškaDokumenta, ŠifraDokumenta, DatumDokumenta, VrstaDokumenta)

**NačinPlaćanja** (NačinPlaćanjaID, NazivNačinaPlaćanja)

**Prijemnica** (PrijemnicaID, DatumOdPrijemnice, DatumDospećaPrijemnice, StatusPrijemnice, *IzveštajServiseraID, NačinPlaćanjaID, OrganizacionaJedinicaID, TipPrijemniceID, MagacinID, PartnerID*)

**StavkaPrijemnice** (RbrStavkePrijemnice, *PrijemnicaID*, KoličinaStavkePrijemnice, BrutoCenaStavkePrijemnice, NetoCenaStavkePrijemnice, CenaZalihaStavkePrijemnice, ZTRStavkePrijemnice*, MaterijalID*)

**TipPrijemnice** (TipPrijemniceID, NazivTipaPrijemnice)

**Magacin** (MagacinID, NazivMagacina, VeličinaZalihaMagacin, VeličinaMagacina, *BrojID,* *UlicaID, PoštanskiBroj, RegionID, OrganizacionaJedinicaID)*

**Delatnost** (DelatnostID, NazivDelatnosti)

**Banka** (BankaID, NazivBanke)

**Trebovanje** (TrebovanjeID, DatumTrebovanja, StatusTrebovanja, *PrijemnicaID, TipTrebovanjaID, GotovProizvodID, MagacinID, MestoTroškaID*)

**TipTrebovanja** (TipTrebovanjaID, NazivTipaTrebovanja)

**StavkaTrebovanja** (RbrStavkeTrebovanja, *TrebovanjeID*, KoličinaStavkeTrebovanja, *MaterijalID*)

**Račun** (RačunID, DatumPrometaRačuna, DatumIzdavanjaRačuna, *TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa*)

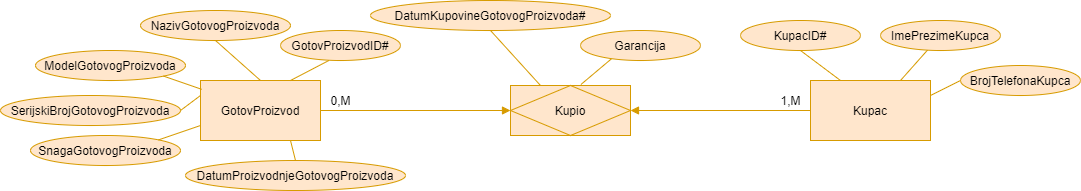
**StavkaRačuna** (RbrStavkeRačuna, *RačunID*, KoličinaStavkeRačuna, PDVStavkeRačuna, RabatStavkeRačuna, *MaterijalID*)

**Valuta** (ValutaID, NazivValute)

# DENORMALIZACIJA RELACIJA

## Denormalizacija relacija uz narušavanje 2NF – Pre – joining tehnika

**Denormalizacija 2NF – Stanje pre primene Pre – joining tehnike**



Slika Pre primene pre - joining tehnike denormalizacije

**GotovProizvod** (GotovProizvodID, NazivGotovogProizvoda, SnagaGotovogProizvida, SerijskiBrojGotovogProizvoda, ModelGotovogProizvoda, DatumProizvodnjeGotovogProizvoda)

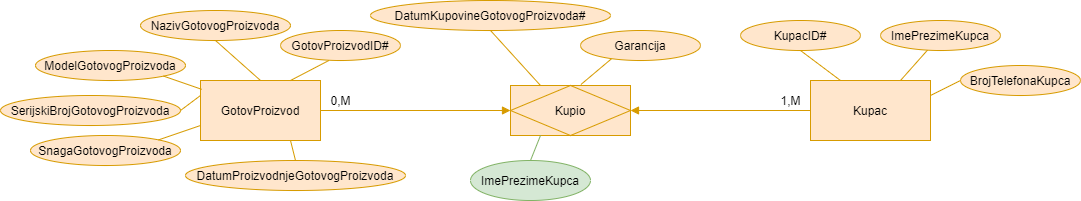
**Kupac** (KupacID, ImePrezimeKupca, BrojTelefonaKupca, *BrojID,* *UlicaID, PoštanskiBroj, RegionID*)

**Kupio** (DatumKupovineGotovogProizvoda, *GotovProizvodID, KupacID*, Garancija)

**Funkcionalne zavisnosti (Kupio):**

DatumKupovineGotovogProizvoda, GotovProizvodID, KupacID -> Garancija

**Denormalizacija 2NF – Stanje nakon primene Pre – joining tehnike**



Slika Nakon primene pre - joining tehnike denormalizacije

**GotovProizvod** (GotovProizvodID, NazivGotovogProizvoda, SnagaGotovogProizvida, SerijskiBrojGotovogProizvoda, ModelGotovogProizvoda, DatumProizvodnjeGotovogProizvoda)

**Kupac** (KupacID, ImePrezimeKupca, BrojTelefonaKupca, *BrojID,* *UlicaID, PoštanskiBroj, RegionID*)

**Kupio** (DatumKupovineGotovogProizvoda, *GotovProizvodID, KupacID*, Garancija, ImePrezimeKupca)

**Funkcionalne zavisnosti (Kupio):**

DatumKupovineGotovogProizvoda, GotovProizvodID, KupacID -> Garancija, ImePrezimeKupca

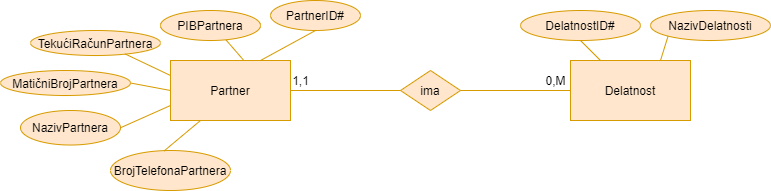
KupacID -> ImePrezimeKupca

Tabela specifikacije trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba da uradi? |
| Kupio | Insert |  | DA | U kolonu ImePrezimeKupca upisuje odgovarajuću vrednost za unetu vrednost KupacID |
| Update | ImePrezimeKupca | DA | Sprečava direktnu izmenu |
| Delete |  | NE |  |
| Kupac | Insert |  | NE |  |
| Update | ImePrezimeKupca | DA | Prilikom izmene vrednosti kolone ImePrezimeKupca u tabeli Kupac, izmenjena vrednost ažurira se u tabeli Kupio |
| Delete |  | NE |  |

## Denormalizacija relacija uz narušavanje 3NF – Pre -joining tehnika

**Denormalizacija 3NF – Stanje pre primene Pre – joining tehnike**



Slika Pre primene pre - joining tehnike denormalizacije

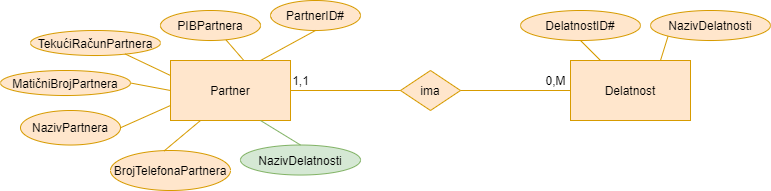
**Partner** (PartnerID, NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MatičniBrojPartnera, TekućiRačunPartnera, *BrojID,* *UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID*)

**Delatnost** (DelatnostID, NazivDelatnosti)

**Funkcionalne zavisnosti (Partner):**

PartnerID -> NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MatičniBrojPartnera, TekućiRačunPartnera, UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID

**Denormalizacija 3NF – Stanje nakon primene Pre – joining tehnike**



Slika Nakon primene pre - joining tehnike denormalizacije

**Partner** (PartnerID, NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MatičniBrojPartnera, TekućiRačunPartnera, NazivDelatnosti, *BrojID,* *UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID*)

**Delatnost** (DelatnostID, NazivDelatnosti)

**Funkcionalne zavisnosti (Partner):**

PartnerID -> NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MatičniBrojPartnera, TekućiRačunPartnera, BrojID, UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID, NazivDelatnosti

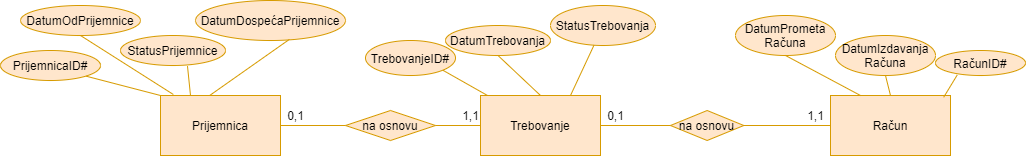
DelatnostID -> NazivDelatnosti

Tabela specifikacije trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba da uradi? |
| Delatnost | Insert |  | NE |  |
| Update | NazivDelatnosti | DA | Prilikom izmene vrednosti polja NazivDelatnosti u tabeli Delatnost, ažurira vrednost u tabeli Partner |
| Delete |  | NE |  |
| Partner | Insert |  | DA | Vrši ažuriranje kolone NazivDelatnosti na osnovu unete vrednosti atributa DelatnostID |
| Update | DelatnostID | DA | Vrši ažuriranje kolone NazivDelatnosti na osnovu izmenjene vrednosti kolone DelatnostID |
| NazivDelatnosti | DA | Zabraniti direktno ažuriranje ove kolone |
| Delete |  | NE |  |

## Denormalizacija relacija uz narušavanje 3NF – Short – circuit keys tehnika

**Denormalizacija 3NF – Stanje pre primene Short - circuit keys tehnike**



Slika Pre primene short - circuit keys tehnike denormalizacije

**Prijemnica** (PrijemnicaID, DatumOdPrijemnice, DatumDospećaPrijemnice, StatusPrijemnice, *IzveštajServiseraID, NačinPlaćanjaID, OrganizacionaJedinicaID, TipPrijemniceID, MagacinID, PartnerID*)

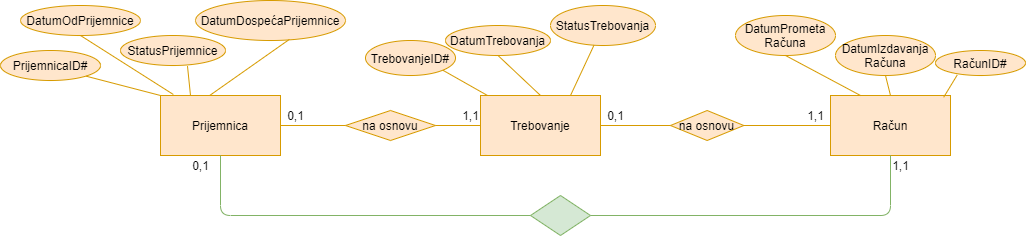
**Trebovanje** (TrebovanjeID, DatumTrebovanja, StatusTrebovanja, *PrijemnicaID, TipTrebovanjaID, GotovProizvodID, MagacinID, MestoTroškaID*)

**Račun** (RačunID, DatumPrometaRačuna, DatumIzdavanjaRačuna, *TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa*)

**Funkcionalne zavisnosti (Račun):**

RačunID -> DatumPrometaRačuna, DatumIzdavanjaRačuna, TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa

**Denormalizacija 3NF – Stanje nakon primene Short - circuit keys tehnike**



Slika 20 Nakon primene short - circuit keys tehnike denormalizacije

**Prijemnica** (PrijemnicaID, DatumOdPrijemnice, DatumDospećaPrijemnice, StatusPrijemnice, *IzveštajServiseraID, NačinPlaćanjaID, OrganizacionaJedinicaID, TipPrijemniceID, MagacinID, PartnerID*)

**Trebovanje** (TrebovanjeID, DatumTrebovanja, StatusTrebovanja, *PrijemnicaID, TipTrebovanjaID, GotovProizvodID, MagacinID, MestoTroškaID*)

**Račun** (RačunID, DatumPrometaRačuna, DatumIzdavanjaRačuna, *TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa, PrijemnicaID*)

**Funkcionalne zavisnosti (Partner):**

RačunID -> DatumPrometaRačuna, DatumIzdavanjaRačuna, TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa, PrijemnicaID

TrebovanjeID -> PrijemnicaID

Tabela specifikacije trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba da uradi? |
| Trebovanje | Insert |  | NE |  |
| Update | PrijemnicaID | DA | Ukoliko postoji Račun za Trebovanje, ažurira PrijemnicaID u tabeli Račun |
| Delete |  | NE |  |
| Račun | Insert |  | DA | U kolonu PrijemnicaID upisuje odgovarajuću vrednost za unetu vrednost TrebovanjeID |
| Update | TrebovanjeID | DA | U kolonu PrijemnicaID upisuje odgovarajuću vrednost na osnovu nove vrednosti za TrebovanjeID |
| PrijemnicaID | DA | Sprečava direktnu izmenu |
| Delete |  | NE |  |

# KORISNIČKI DEFINISANI TIPOVI I TRIGERI

## Definicija korisničkom tipa i njegovo korišćenje

**Struktuirani tip**

Kreiranje Object tipa:

CREATE OR REPLACE TYPE PELETPODACI AS OBJECT(

NazivPeleta VARCHAR2(100),

MarkaPeleta VARCHAR2(50),

PrecnikPeleta NUMBER,

DuzinaPeleta NUMBER,

MEMBER FUNCTION DAJ\_NAZIV RETURN VARCHAR2,

MEMBER FUNCTION DAJ\_MARKU RETURN VARCHAR2,

MEMBER FUNCTION DAJ\_PRECNIK RETURN NUMBER,

MEMBER FUNCTION DAJ\_DUZINU RETURN NUMBER)

INSTANTIABLE NOT FINAL;

Kreiranje body-ja:

CREATE OR REPLACE TYPE BODY PELETPODACI AS

MEMBER FUNCTION DAJ\_NAZIV RETURN VARCHAR2 IS

BEGIN

RETURN SELF.NazivPeleta;

END;

MEMBER FUNCTION DAJ\_MARKU RETURN VARCHAR2 IS

BEGIN

RETURN SELF.MarkaPeleta;

END;

MEMBER FUNCTION DAJ\_PRECNIK RETURN NUMBER IS

BEGIN

RETURN SELF.PrecnikPeleta;

END;

MEMBER FUNCTION DAJ\_DUZINU RETURN NUMBER IS

BEGIN

RETURN SELF.DuzinaPeleta;

END;

END;

Kreiran struktuirani tip je korišćen prilikom kreiranja tabele Pelet, kao što je prikazano u nastavku:

CREATE TABLE PELET(

PeletID NUMBER (7,0) NOT NULL,

Podaci PELETPODACI,

CONSTRAINT PELET\_PK PRIMARY KEY (PeletID)

);

Unos podataka u tabelu Pelet, koja sadrži kolonu sa korisnički definisanim tipom PELETPODACI:

INSERT INTO PELET VALUES (

1, PELETPODACI('Komo', 'Alfa-Plam', 60, 120)

);

**Distinkt tip**

Kreiranje Object tipa:

CREATE OR REPLACE TYPE NAZIVVREDNOST AS OBJECT(

Vrednost VARCHAR2(50),

CONSTRUCTOR FUNCTION NAZIVVREDNOST(Vrednost VARCHAR2)

RETURN SELF AS RESULT,

MEMBER FUNCTION DAJ\_VREDNOST RETURN VARCHAR2)

INSTANTIABLE FINAL;

Kreiranje body-ja:

CREATE OR REPLACE TYPE BODY NAZIVVREDNOST AS

CONSTRUCTOR FUNCTION NAZIVVREDNOST(Vrednost VARCHAR2)

RETURN SELF AS RESULT

AS

BEGIN

IF(Vrednost='ZAMENA ELEKTRONIKE 1') THEN SELF.Vrednost:=Vrednost;

ELSIF (Vrednost='ZAMENA ELEKTRONIKE 2') THEN SELF.Vrednost:=Vrednost;

ELSIF (Vrednost='ZAMENA ELEKTRONIKE 3') THEN SELF.Vrednost:=Vrednost;

ELSIF (Vrednost='ZAMENA ELEKTRONIKE 4') THEN SELF.Vrednost:=Vrednost;

ELSIF (Vrednost='ZAMENA ELEKTRONIKE 5') THEN SELF.Vrednost:=Vrednost;

ELSE RAISE\_APPLICATION\_ERROR(-20001,'Naziv tipa usluge ne postoji u bazi');

END IF;

RETURN;

END;

MEMBER FUNCTION DAJ\_VREDNOST RETURN VARCHAR2 IS

BEGIN

RETURN SELF.Vrednost;

END;

END;

Kreiran distinkt tip je korišćen prilikom kreiranja tabele TipUsluge, kao što je prikazano u nastavku:

CREATE TABLE TIPUSLUGE(

TipUslugeID NUMBER(7,0) NOT NULL,

NazivTipaUsluge NAZIVVREDNOST,

CenaTipaUsluge NUMBER,

CONSTRAINT TIPUSLUGE\_PK PRIMARY KEY (TIPUSLUGEID)

);

Unos podataka u tabelu TipUsluge, koja sadrži kolonu sa korisnički definisanim tipom NAZIVVREDNOST:

INSERT INTO TIPUSLUGE VALUES (

1, NAZIVVREDNOST('ZAMENA ELEKTRONIKE 1'), 1000

);

Prikaz podataka iz tabele TipUsluge koja sadrži korisnički definisani tip:

SELECT \* FROM TipUsluge;

SELECT tu.TipUslugeID,tu.NazivTipaUsluge.DAJ\_VREDNOST() AS Naziv\_tipa\_usluge, tu.CenaTipaUsluge FROM TIPUSLUGE tu;

## Realizacija proceduralne logike uz pomoć trigera

**2NF TRIGERI**

**Prilikom izmene vrednosti kolone ImePrezimeKupca u tabeli Kupac, izmenjena vrednost ažurira se u tabeli Kupio.**

Triger vrši izmenu kolone ImePrezimeKupca u tabeli Kupio na osnovu izmene kolone ImePrezimeKupca u tabeli Kupac tako da kolona ImePrezimeKupca u tabeli Kupio odgovara koloni ImePrezimeKupca u tabeli Kupac ako je deo primarnog ključa KupacID u tabeli Kupio jednak primarnom ključu KupacID u tabeli Kupac.

CREATE OR REPLACE TRIGGER IZMENA\_IMEPREZIMEKUPCA\_IZ\_KUPAC\_U\_KUPIO

AFTER UPDATE OF ImePrezimeKupca

ON Kupac

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER ZABRANA\_IZMENE\_IMEPREZIMEKUPCA DISABLE';

UPDATE Kupio

SET ImePrezimeKupca = :new.ImePrezimeKupca

WHERE KupacID = :NEW.KupacID;

COMMIT;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER ZABRANA\_IZMENE\_IMEPREZIMEKUPCA ENABLE';

END;

END;

Prikaz tabele Kupio pre pokretanja trigera:

SELECT \* FROM Kupio;

Test primer:

UPDATE Kupac SET ImePrezimeKupca='Ana Ivanovic' WHERE ImePrezimeKupca='Ana Vlahovic';

Prikaz tabele Kupio nakon pokretanja trigera:

SELECT \* FROM Kupio;

**Triger ažurira vrednost kolone ImePrezimeKupca na osnovu unete vrednosti kolone KupacID:**

CREATE OR REPLACE TRIGGER DODAVANJE\_IMEPREZIMEKUPCA\_U\_KUPIO

BEFORE INSERT

ON Kupio

FOR EACH ROW

DECLARE

v\_ImePrezimeKupca KUPAC.ImePrezimeKupca%TYPE;

BEGIN

SELECT ImePrezimeKupca

INTO v\_ImePrezimeKupca

FROM Kupac

WHERE KupacID=:NEW.KupacID;

:NEW.ImePrezimeKupca:=v\_ImePrezimeKupca;

END;

Prikaz tabele Kupio pre pokretanja trigera:

SELECT \* FROM Kupio;

Test primer:

INSERT INTO Kupio (DatumKupovineGotovogProizvoda, GotovProizvodID, KupacID, Garancija) VALUES (TO\_DATE('05/12/2022', 'DD/MM/YYYY'),1,1,1);

Prikaz tabele Kupio nakon pokretanja trigera:

SELECT \* FROM Kupio;

**Triger ne dozvoljava direktnu izmenu kolone ImePrezimeKupca u okviru tabele Kupio:**

CREATE TRIGGER ZABRANA\_IZMENE\_IMEPREZIMEKUPCA

BEFORE UPDATE OF ImePrezimeKupca

ON Kupio

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(-20000, 'Nije dozvoljena direktna izmena kolone ImePrezimeKupca u okviru tabele Kupio');

END;

Test da li radi:

UPDATE Kupio SET ImePrezimeKupca='Ana Kostic' WHERE ImePrezimeKupca='Ana Ivanovic';

**3NF TRIGERI – PRE JOINING**

**Prilikom izmene vrednosti polja NazivDelatnosti u tabeli Delatnost, ažurira vrednost u tabeli Partner:**

Kada se izmeni vrednost polja NazivDelatnosti u tabeli Delatnost, pokreće se triger koji ažurira vrednost u tabeli Partner tako da kolona NazivDelatnosti u tabeli Delatnost odgovara koloni NazivDelatnosti u tabeli Delatnost ako je spoljni ključ DelatnostID u tabeli Partner jednak primarnom ključu DelatnostID u tabeli Delatnost.

CREATE OR REPLACE TRIGGER IZMENA\_NAZIVDELATNOSTI\_IZ\_DELATNOST\_U\_PARTNER

AFTER UPDATE OF NazivDelatnosti

ON Delatnost

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER ZABRANA\_IZMENE\_NAZIVDELATNOSTI\_PARTNER DISABLE';

UPDATE Partner

SET NazivDelatnosti=:NEW.NazivDelatnosti

WHERE DelatnostID=:NEW.DelatnostID;

COMMIT;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER ZABRANA\_IZMENE\_NAZIVDELATNOSTI\_PARTNER ENABLE';

END;

END;

Prikaz tabele Partner pre pokretanja trigera:

SELECT \* FROM Partner;

Test primer:

UPDATE Delatnost SET NazivDelatnosti='Metalurska ind' WHERE NazivDelatnosti='Met ind';

Prikaz tabele Partner nakon pokretanja trigera:

SELECT \* FROM Partner;

**Triger vrši ažuriranje kolone NazivDelatnosti na osnovu unete vrednosti atributa DelatnostID:**

CREATE OR REPLACE TRIGGER DODAVANJE\_NAZIVADELATNOSTI\_U\_PARTNER

BEFORE INSERT ON Partner

FOR EACH ROW

DECLARE

v\_NazivDelatnosti DELATNOST.NazivDelatnosti%TYPE;

BEGIN

SELECT NazivDelatnosti INTO v\_NazivDelatnosti

FROM Delatnost

WHERE DelatnostID=:NEW.DelatnostID;

:NEW.NazivDelatnosti:= v\_NazivDelatnosti;

END;

Prikaz tabele Partner nakon pokretanja trigera:

SELECT \* FROM Partner;

Provera da li radi:

INSERT INTO Partner(PartnerID, NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MatičniBrojPartnera, TekućiRačunPartnera, BrojID, UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID) VALUES (1,'Fluid 012', '069333333', '123456', '123457', '120-123-212',1,1,17501,1,1,1);

Prikaz tabele Partner nakon pokretanja trigera:

SELECT \* FROM Partner;

**Triger vrši ažuriranje kolone NazivDelatnosti na osnovu izmenjene vrednosti kolone DelatnostID:**

CREATE OR REPLACE TRIGGER IZMENA\_DELATNOSTID\_U\_PARTNER

BEFORE UPDATE OF DelatnostID

ON Partner

FOR EACH ROW

DECLARE

v\_NazivDelatnosti DELATNOST.NazivDelatnosti%TYPE;

BEGIN

SELECT NazivDelatnosti INTO v\_NazivDelatnosti

FROM Delatnost

WHERE DelatnostID=:NEW.DelatnostID;

:NEW.NazivDelatnosti:= v\_NazivDelatnosti;

END;

Prikaz tabele Partner pre pokretanja trigera:

SELECT \* FROM Partner;

Test primer:

UPDATE Partner SET DelatnostID=2 WHERE DelatnostID=1;

Prikaz tabele Partner nakon pokretanja trigera:

SELECT \* FROM Partner;

**Triger ne dozvoljava direktnu izmenu kolone NazivDelatnosti u okviru tabele Partner:**

CREATE OR REPLACE TRIGGER ZABRANA\_IZMENE\_NAZIVDELATNOSTI\_PARTNER

AFTER UPDATE OF NazivDelatnosti

ON Partner

BEGIN

RAISE\_APPLICATION\_ERROR (-20000, 'Nije dozvoljena direktna izmena vrednosti kolone NazivDelatnosti u tabeli Partner');

END;

Test primer:

UPDATE Partner SET NazivDelatnosti='Metalurska industrija' WHERE NazivDelatnosti='Metalurska ind';

**3NF TRIGERI – SHORT-CIRCUIT KEYS**

**Ukoliko postoji Račun za Trebovanje, ažurira PrijemnicaID u tabeli Račun:**

CREATE OR REPLACE TRIGGER IZMENA\_PRIJEMNICAID\_U\_RACUN

AFTER UPDATE OF PrijemnicaID

ON Trebovanje

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER ZABRANA\_IZMENE\_PRIJEMNICAID\_U\_RACUN DISABLE';

UPDATE Racun

SET PrijemnicaID=:NEW.PrijemnicaID

WHERE TrebovanjeID=:NEW.TrebovanjeID;

COMMIT;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER ZABRANA\_IZMENE\_PRIJEMNICAID\_U\_RACUN ENABLE';

END;

END;

Prikaz tabele Račun pre pokretanja trigera:

SELECT \* FROM Racun;

Test primer:

UPDATE Trebovanje SET PrijemnicaID=2 WHERE PrijemnicaID=1;

Prikaz tabele Račun nakon pokretanja trigera:

SELECT \* FROM Racun;

**U kolonu PrijemnicaID upisuje odgovarajuću vrednost za unetu vrednost TrebovanjeID:**

CREATE OR REPLACE TRIGGER DODAVANJE\_PRIJEMNICAID\_U\_RACUN

BEFORE INSERT ON Racun

FOR EACH ROW

DECLARE

v\_PrijemnicaID PRIJEMNICA.PrijemnicaID%TYPE;

BEGIN

SELECT PrijemnicaID INTO v\_PrijemnicaID

FROM Trebovanje

WHERE TrebovanjeID=:NEW.TrebovanjeID;

:NEW.PrijemnicaID:=v\_PrijemnicaID;

END;

Prikaz tabele Račun nakon pokretanja trigera:

SELECT \* FROM Racun;

Test primer:

INSERT INTO Racun (RacunID, DatumPrometaRacuna, DatumIzdavanjaRacuna, TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa) VALUES (2,TO\_DATE('05/12/2022', 'DD/MM/YYYY'),TO\_DATE('05/12/2022', 'DD/MM/YYYY'), 1, 1, 1, 17501, 1, 17501, 1);

Prikaz tabele Račun nakon pokretanja trigera:

SELECT \* FROM Racun;

**U kolonu PrijemnicaID upisuje odgovarajuću vrednost na osnovu nove vrednosti za TrebovanjeID:**

CREATE OR REPLACE TRIGGER IZMENA\_TREBOVANJEID\_U\_RACUN

BEFORE UPDATE OF TrebovanjeID

ON Racun

FOR EACH ROW

DECLARE

v\_PrijemnicaID PRIJEMNICA.PrijemnicaID%TYPE;

BEGIN

SELECT PrijemnicaID INTO v\_PrijemnicaID

FROM Trebovanje

WHERE TrebovanjeID=:NEW.TrebovanjeID;

:NEW.PrijemnicaID:=v\_PrijemnicaID;

END;

Prikaz tabele Račun nakon pokretanja trigera:

SELECT \* FROM Racun;

Test primer:

UPDATE Racun SET TrebovanjeID=2 WHERE TrebovanjeID=1;

Prikaz tabele Račun nakon pokretanja trigera:

SELECT \* FROM Racun;

**Sprečava direktnu izmenu kolone PrijemnicaID u tabeli Račun:**

CREATE OR REPLACE TRIGGER ZABRANA\_IZMENE\_PRIJEMNICAID\_U\_RACUN

AFTER UPDATE OF PrijemnicaID

ON Racun

BEGIN

RAISE\_APPLICATION\_ERROR(-20000, 'Nije dozvoljena direktna izmena vrednosti kolone PrijemnicaID u tabeli Racun');

END;

Test primer:

UPDATE Racun SET PrijemnicaID=2 WHERE PrijemnicaID=1;

# OPTIMIZACIJA BAZE PODATAKA

## Definisanje i analiza indeksa

**Kreiranje indeksa i skeniranje indeksa po opsegu (range):**

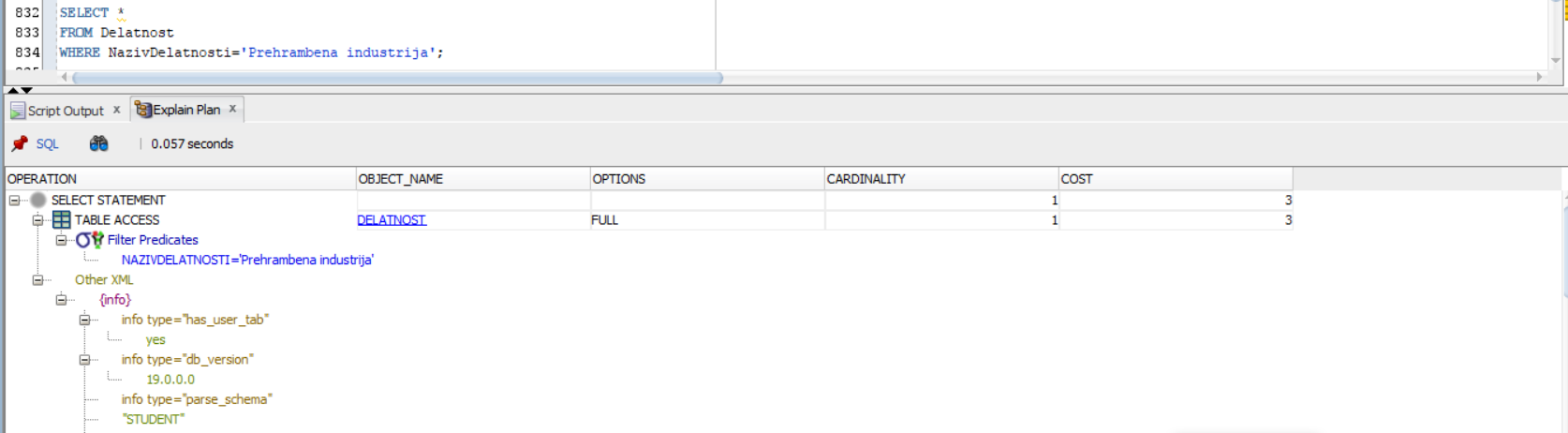
Operacija bez korišćenja indeksa:

SELECT \*

FROM Delatnost

WHERE NazivDelatnosti='Prehrambena industrija';

Pre kreiranja indeksa na tabeli Delatnost za naziv delatnosti, koštanje date operacije je 3.



Slika 21 Koštanje operacije pre kreiranja indeksa na tabeli Delatnost

Kreiramo indeks INDEX\_NAZIVDELATNOSTI i uz primenu skeniranja indeksa po opsegu (range) kreiramo isti upit kao na slici iznad.

CREATE INDEX INDEX\_NAZIVDELATNOSTI ON Delatnost (NazivDelatnosti);

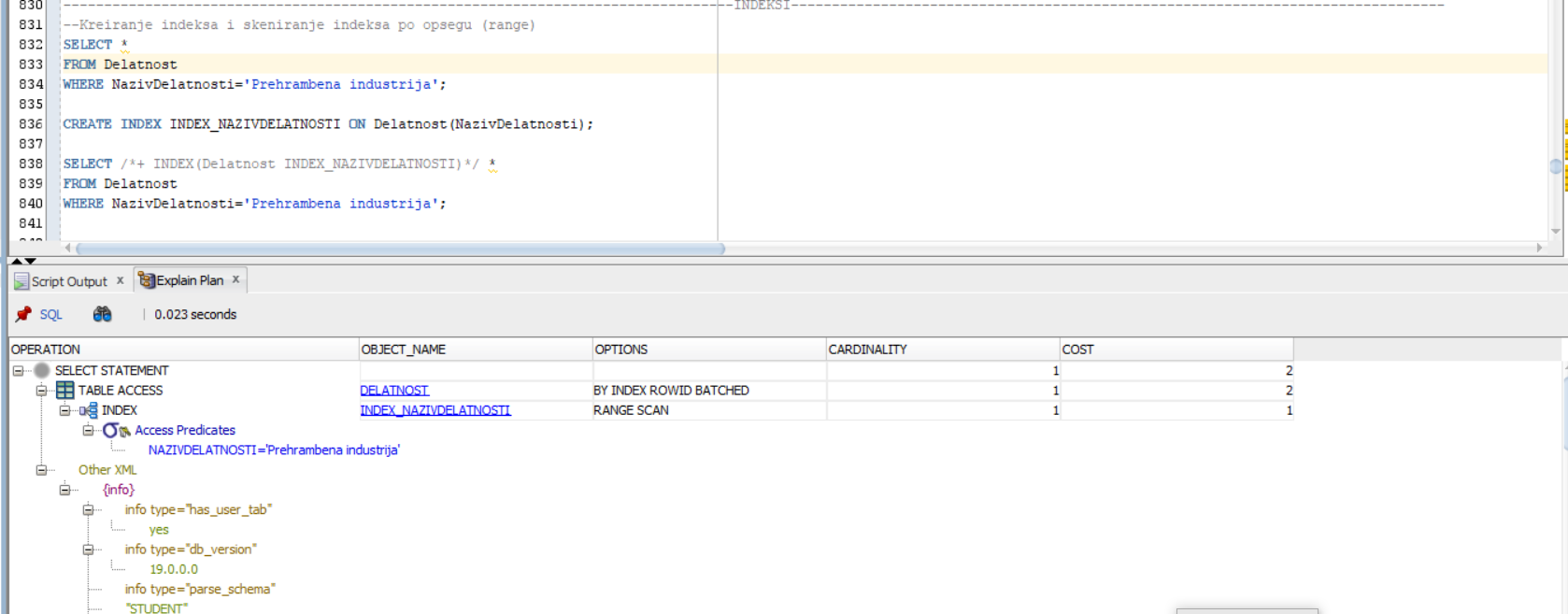
Operacija sa korišćenjem indeksa:

SELECT /\*+ INDEX(Delatnost INDEX\_NAZIVDELATNOSTI)\*/ \*

FROM Delatnost

WHERE NazivDelatnosti='Prehrambena industrija';

Koštanje date operacije nakon korišćenja indeksa može se videte na slici ispod. Da bi se videla prava razlika u koštanju date operacije nakon dodatog indeksa, neophodno je imati dosta veći broj rekorda.



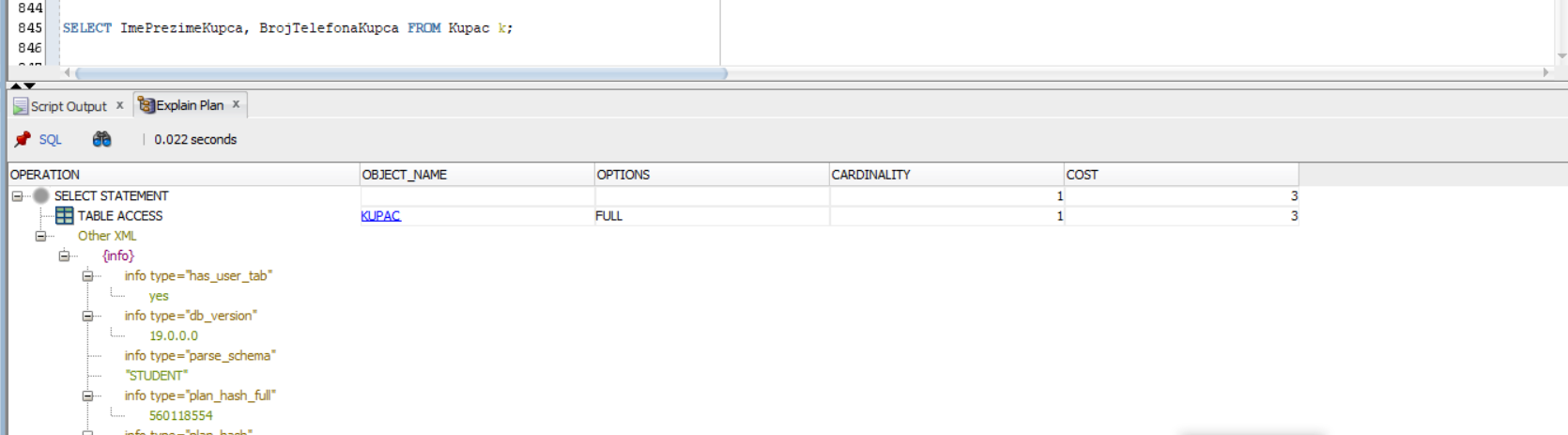
Slika 22 Koštanje operacije nakon kreiranja indeksa na tabeli Delatnost

**Kreiranje indeksa i skeniranje indeksa sa spajanjem (index join):**

Operacija bez korišćenja indeksa:

SELECT ImePrezimeKupca, BrojTelefonaKupca FROM Kupac k;

Pre kreiranja indeksa na tabeli Kupac za ImePrezimeKupca I BrojTelefona, koštanje date operacije je 3.



Slika 23 Koštanje operacije pre kreiranja indeksa na tabeli Kupac

Kreiramo indekse INDEX\_IMPREZIME i INDEX\_BROJTELEFONA i uz primenu skeniranje indeksa sa spajanjem (index join)kreiramo isti upit kao na slici iznad.

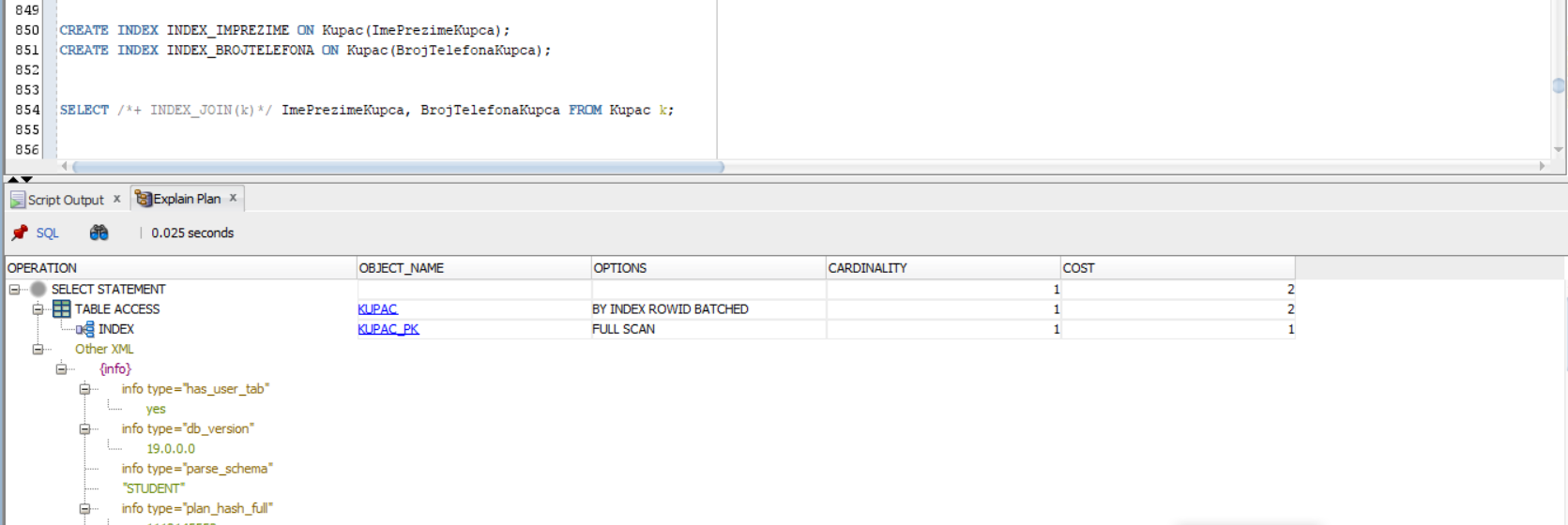
CREATE INDEX INDEX\_IMPREZIME ON Kupac(ImePrezimeKupca);

CREATE INDEX INDEX\_BROJTELEFONA ON Kupac(BrojTelefonaKupca);

Operacija sa korišćenjem indeksa:

SELECT /\*+ INDEX\_JOIN(k)\*/ ImePrezimeKupca, BrojTelefonaKupca FROM Kupac k;

Koštanje date operacije nakon korišćenja indeksa može se videte na slici ispod. Da bi se videla prava razlika u koštanju date operacije nakon dodatog indeksa, neophodno je imati dosta veći broj rekorda.



Slika 24 Koštanje operacije nakon kreiranja indeksa na tabeli Kupac

## Horizontalno particionisanje

Horizontalno particionisanje izvršeno je nad tabelom RadniNalog koja će imati particije realizovan i nerealizovan u zavisnosti od vrednosti atributa StatusRadnogNaloga. Koristi se List particionisanje, gde se zadaje lista vrednosti kao kriterijum za particionisanje.

Relacija RadniNalog na osnovu koje je kreirana tabela sa particijama:

**RadniNalog** (RadniNalogID, StatusRadnogNaloga, DatumKreiranjaRadnogNaloga, DatumIzmeneRadnogNaloga, PlaniraniDatumRadnogNaloga, DatumRealizacijeRadnogNaloga, NapomenaRadnogNaloga, PrioritetRadnogNaloga, OpisRadnogNaloga, EksportovanostRadnogNaloga, ZavršenostRadnogNaloga, SpornostRadnogNaloga, InternostRadnogNaloga*, TipRadnogNalogaID, PartnerID, GotovProizvodID, KupacID* )

Kreiranje tabele RADNINALOG\_PART sa particijama:

CREATE TABLE RADNINALOG\_PART(

RadniNalogID NUMBER(7,0),

StatusRadnogNaloga VARCHAR2(20),

DatumKreiranjaRadnogNaloga DATE,

DatumIzmeneRadnogNaloga DATE,

PlaniraniDatumRadnogNaloga DATE,

DatumRealizacijeRadnogNaloga DATE,

NapomenaRadnogNaloga VARCHAR2(150),

PrioritetRadnogNaloga VARCHAR2(40),

OpisRadnogNaloga VARCHAR2(150),

EksportovanostRadnogNaloga NUMBER(1),

ZavrsenostRadnogNaloga NUMBER(1),

SpornostRadnogNaloga NUMBER(1),

InternostRadnogNaloga NUMBER(1),

TipRadnogNalogaID NUMBER(7,0),

PartnerID NUMBER(7,0),

GotovProizvodID NUMBER(7,0),

KupacID NUMBER(7,0),

CONSTRAINT RADNINALOG\_PKK PRIMARY KEY (RadniNalogID),

CONSTRAINT TIPRADNOGNALOGA\_FKK FOREIGN KEY (TipRadnogNalogaID) REFERENCES TIPRADNOGNALOGA (TipRadnogNalogaID),

--CONSTRAINT PARTNERRN\_FKK FOREIGN KEY (PartnerID) REFERENCES PARTNER (PartnerID),

CONSTRAINT PARTNERRN\_FKK FOREIGN KEY (PartnerID) REFERENCES PARTNER\_BITNIDETALJI (PartnerID),

CONSTRAINT GOTOVPROIZVODRN\_FKK FOREIGN KEY (GotovProizvodID) REFERENCES GOTOVPROIZVOD (GotovProizvodID),

CONSTRAINT KUPACRN\_FKK FOREIGN KEY (KupacID) REFERENCES KUPAC (KupacID),

CONSTRAINT EKSPORTOVANOSTRN\_CKK CHECK (EksportovanostRadnogNaloga in (0,1)),

CONSTRAINT ZAVRSENOSTRN\_CKK CHECK (ZavrsenostRadnogNaloga in (0,1)),

CONSTRAINT SPORNOSTRN\_CKK CHECK (SpornostRadnogNaloga in (0,1)),

CONSTRAINT INTERNOSTRN\_CKK CHECK (InternostRadnogNaloga in (0,1))

)

PARTITION BY LIST(StatusRadnogNaloga)(

PARTITION realizovan VALUES('Realizovan'),

PARTITION nerealizovan VALUES('Kreiran','U toku')

);

Dodavanje novih podataka u tabelu RADNINALOG\_PART:

INSERT INTO RADNINALOG\_PART VALUES (1,'Realizovan', TO\_DATE('04/12/2022','DD/MM/YYYY'),TO\_DATE('16/12/2022','DD/MM/YYYY'),TO\_DATE('04/12/2022','DD/MM/YYYY'),TO\_DATE('16/12/2022','DD/MM/YYYY'), 'Napomena','Obavljanje odmah', 'Opis', 0,0,0,0,1,1,1,1);

INSERT INTO RADNINALOG\_PART VALUES (2,'U toku', TO\_DATE('04/12/2022','DD/MM/YYYY'),TO\_DATE('16/12/2022','DD/MM/YYYY'),TO\_DATE('04/12/2022','DD/MM/YYYY'),TO\_DATE('16/12/2022','DD/MM/YYYY'), 'Napomena','Obavljanje odmah', 'Opis', 0,0,0,0,1,1,1,1);

INSERT INTO RADNINALOG\_PART VALUES (3,'Kreiran', TO\_DATE('04/12/2022','DD/MM/YYYY'),TO\_DATE('16/12/2022','DD/MM/YYYY'),TO\_DATE('04/12/2022','DD/MM/YYYY'),TO\_DATE('16/12/2022','DD/MM/YYYY'), 'Napomena','Obavljanje odmah', 'Opis', 0,0,0,0,1,1,1,1);

Pregled svih realizovanih radnih naloga:

SELECT \* FROM RADNINALOG\_PART PARTITION(realizovan);

## Vertikalno particionisanje

**Partner** (PartnerID, NazivPartnera, BrojTelefonaPartnera, PIBPartnera, MaticniBrojPartnera, TekuciRacunPartnera, NazivDelatnosti, *BrojID, UlicaID, PoštanskiBroj, RegionID, DelatnostID, BankaID*)

Ovu tabelu delimo na dve: Partner\_BitniDetalji i Partner\_SporedniDetalji. U prvu smeštamo kolone koje se često koriste u upitima, a u drugu one kolone koje se retko koriste u upitima.

Kreiranje tabele Partner\_BitniDetalji:

**Partner\_BitniDetalji** (PartnerID, NazivPartnera, BrojTelefonaPartnera)

CREATE TABLE PARTNER\_BITNIDETALJI(

PartnerID NUMBER(7,0),

NazivPartnera VARCHAR2(30),

BrojTelefonaPartnera VARCHAR2(13),

CONSTRAINT PARTNER\_BITNIDETALJI\_PK PRIMARY KEY (PartnerID)

);

Kreiranje tabele Partner\_SporedniDetalji:

**Partner\_SporedniDetalji** (PartnerID, PIBPartnera, MaticniBrojPartnera, TekuciRacunPartnera, NazivDelatnosti, *BrojID, UlicaID, PostanskiBroj, RegionID, DelatnostID, BankaID*)

CREATE TABLE PARTNER\_SPOREDNIDETALJI(

PartnerID NUMBER(7,0),

PIBPartnera VARCHAR2(30),

MaticniBrojPartnera VARCHAR2(30),

TekuciRacunPartnera VARCHAR2(30),

NazivDelatnosti VARCHAR2(30),

BrojID NUMBER(7,0),

UlicaID NUMBER(7,0),

PostanskiBroj NUMBER(7,0),

RegionID NUMBER(7,0),

DelatnostID NUMBER(7,0),

BankaID NUMBER(7,0),

CONSTRAINT PARTNER\_SPOREDNIDETALJI\_PK PRIMARY KEY (PartnerID),

CONSTRAINT BROJPSD\_FK FOREIGN KEY (BrojID, UlicaID, PostanskiBroj, RegionID) REFERENCES BROJ (BrojID, UlicaID, PostanskiBroj, RegionID),

CONSTRAINT DELATNOSTPSD\_FK FOREIGN KEY (DelatnostID) REFERENCES DELATNOST (DelatnostID),

CONSTRAINT BANKAPSD\_FK FOREIGN KEY (BankaID) REFERENCES BANKA (BankaID)

);

Budući da je nad tabelom Partner rađena denormalizacija i ona sadrži redundantan podatak NazivDelatnosti koji se sada nalaze u tabeli Partner\_SporedniDetalji, potrebno je sve trigere za očuvanje referencijalnog integriteta koji su implementirani nad tabelom Partner sada implementirati nad tabelom Partner\_SporedniDetalji.

Nakon što su implementirani trigeri, kreira se pogled koji vraća sve kolone iz podeljenih tabela.

CREATE OR REPLACE VIEW PARTNER\_VIEW

AS

SELECT pb.PartnerID, pb.NazivPartnera, pb.BrojTelefonaPartnera,

ps.PIBPartnera, ps.MaticniBrojPartnera, ps.TekuciRacunPartnera, ps.NazivDelatnosti, ps.BrojID, ps.UlicaID, ps.PostanskiBroj, ps.RegionID, ps.DelatnostID, ps.BankaID

FROM Partner\_BitniDetalji pb,Partner\_SporedniDetalji ps

WHERE pb.PartnerID=ps.PartnerID;

Pregled svih podataka:

SELECT \* FROM PARTNER\_VIEW;

Triger za unos novog partnera (instead of trigger):

CREATE OR REPLACE TRIGGER PARTNER\_INSERT\_VIEW

INSTEAD OF INSERT ON PARTNER\_VIEW

FOR EACH ROW

BEGIN

INSERT INTO Partner\_BitniDetalji (PartnerID, NazivPartnera, BrojTelefonaPartnera) VALUES (:NEW.PartnerID , :NEW.NazivPartnera, :NEW.BrojTelefonaPartnera);

INSERT INTO Partner\_SporedniDetalji (PartnerID, PIBPartnera, MaticniBrojPartnera, TekuciRacunPartnera, BrojID, UlicaID, PostanskiBroj, RegionID, DelatnostID, BankaID)

VALUES (:NEW.PartnerID, :NEW.PIBPartnera, :NEW.MaticniBrojPartnera, :NEW.TekuciRacunPartnera, :NEW.BrojID, :NEW.UlicaID, :NEW.PostanskiBroj, :NEW.RegionID, :NEW.DelatnostID, :NEW.BankaID);

END;

Unos novog partnera preko pogleda:

INSERT INTO PARTNER\_VIEW (PartnerID, NazivPartnera, BrojTelefonaPartnera,PIBPartnera, MaticniBrojPartnera, TekuciRacunPartnera, BrojID, UlicaID, PostanskiBroj, RegionID, DelatnostID, BankaID) VALUES (1,'Vitez', '065223344',1, '1111111', '2222222', '1234556', 1,1,17501,1,1,1);

INSERT INTO PARTNER\_VIEW (PartnerID, NazivPartnera, BrojTelefonaPartnera,PIBPartnera, MaticniBrojPartnera, TekuciRacunPartnera, BrojID, UlicaID, PostanskiBroj, RegionID, DelatnostID, BankaID) VALUES (2,'Fluid', '065223344',2, '1111111', '2222222', '1234556', 1,1,17501,1,1,1);

Triger za ažuriranje Partnera (instead of triger):

CREATE OR REPLACE TRIGGER PARTNER\_UPDATE\_VIEW

INSTEAD OF UPDATE ON Partner\_View

FOR EACH ROW

BEGIN

UPDATE Partner\_BitniDetalji

SET PartnerID=:NEW.PartnerID,NazivPartnera=:NEW.PartnerID,BrojTelefonaPartnera=:NEW.BrojTelefonaPartnera

WHERE Partner\_BitniDetalji.PartnerID=:NEW.PartnerID;

UPDATE Partner\_SporedniDetalji

SET PartnerID=:NEW.PartnerID, PIBPartnera=:NEW.PIBPartnera, MaticniBrojPartnera=:NEW.MaticniBrojPartnera, TekuciRacunPartnera=:NEW.TekuciRacunPartnera, BrojID=:NEW.BrojID, UlicaID=:NEW.UlicaID, PostanskiBroj=:NEW.PostanskiBroj,RegionID=:NEW.RegionID, DelatnostID=:NEW.DelatnostID, BankaID=:NEW.BankaID

WHERE Partner\_SporedniDetalji.PartnerID=:NEW.PartnerID;

END;

Ažuriranje partnera preko pogleda:

UPDATE Partner\_View SET NazivPartnera='Fluid' WHERE PartnerID=1;

Triger za brisanje partnera:

CREATE OR REPLACE TRIGGER PARTNER\_DELETE\_VIEW

INSTEAD OF DELETE ON PARTNER\_VIEW

FOR EACH ROW

BEGIN

DELETE FROM Partner\_BitniDetalji WHERE Partner\_BitniDetalji.PartnerID=:OLD.PartnerID;

DELETE FROM Partner\_SporedniDetalji WHERE Partner\_SporedniDetalji.PartnerID=:OLD.PartnerID;

END;

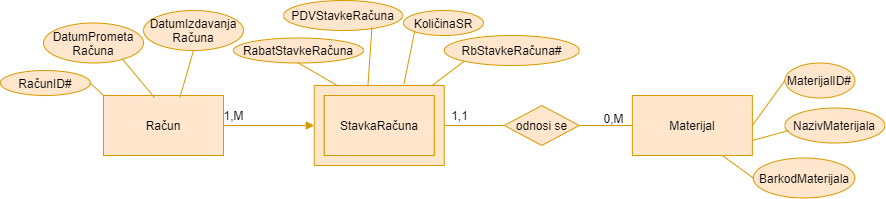
Brisanje partnera preko pogleda:

DELETE Partner\_view WHERE PartnerID=2;

## Primena drugih optimizacionih tehnika

**1. Storing Derivable Values tehnika optimizacije**

Konceptualni model pre optimizacije:



Slika Konceptualni model pre primene Storing Derivable Values tehnike optimizacije

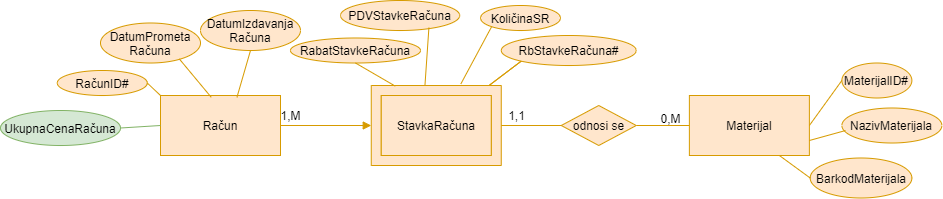
Relacioni model pre optimizacije:

**Račun** (RačunID, DatumPrometaRačuna, DatumIzdavanjaRačuna, *TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa*)

**StavkaRačuna** (RbrStavkeRačuna, *RačunID*, KoličinaStavkeRačuna, PDVStavkeRačuna, RabatStavkeRačuna, *MaterijalID*)

**Materijal** (MaterijalID, NazivMaterijala, CenaMaterijala, BarkodMaterijala, *JedinicaMereID*)

Konceptualni model nakon optimizacije:



Slika Konceptualni model nakon primene Storing Derivable Values tehnike optimizacije

Relacioni model nakon optimizacije:

**Račun** (RačunID, DatumPrometaRačuna, DatumIzdavanjaRačuna, UkupnaCenaRačuna, *TrebovanjeID, ValutaID, PartnerID, GradIzdavanja, RegionIzdavanja, GradPrometa, RegionPrometa*)

**StavkaRačuna** (RbrStavkeRačuna, *RačunID*, KoličinaStavkeRačuna, PDVStavkeRačuna, RabatStavkeRačuna, *MaterijalID*)

**Materijal** (MaterijalID, NazivMaterijala, CenaMaterijala, BarkodMaterijala, *JedinicaMereID*)

Tabela specifikacije trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba da uradi? |
| Račun | Insert |  | NE |  |
| Update | UkupnaCenaRačuna | DA | Zabrana direktnog ažuriranja polja UkupnaCenaRačuna |
| Delete |  | NE |  |
| StavkaRačuna | Insert |  | DA | Prilikom unosa nove StavkeRačuna, poziva se procedura za ažuriranje polja UkupnaCenaRačuna na Računu |
| Update | KoličinaSR | DA | Prilikom ažuriranja nove StavkeRačuna, poziva se procedura za ažuriranje polja UkupnaCenaRačuna na Računu |
| Delete |  | DA | Prilikom brisanja StavkeRačuna, poziva se procedura za ažuriranje polja UkupnaCenaRačuna na Računu |

Dodavanje kolone UkupnaCenaRačuna u tabelu Račun:

ALTER TABLE RACUN ADD UkupnaCenaRacuna NUMBER;

Kreiranje paketa Račun\_Paket:

CREATE OR REPLACE PACKAGE RACUN\_PAKET AS

PROCEDURE postaviID(RacunSifra NUMBER);

FUNCTION vratiID RETURN NUMBER;

END RACUN\_PAKET;

Kreiranje tela paketa Račun\_Paket:

CREATE OR REPLACE PACKAGE BODY RACUN\_PAKET IS

RacunID NUMBER;

PROCEDURE postaviID(RacunSifra NUMBER) IS

BEGIN

RacunID:=RacunSifra;

END;

FUNCTION vratiID RETURN NUMBER IS

BEGIN

RETURN RacunID;

END;

END;

Triger TB\_UKUPNA\_CENA\_RACUNA: Svaki put kada se menja nešto u tabeli StavkaRačuna, hoćemo da se to odrazi i na jak objekat Račun, odnosno da to upišemo u polje UkupnaCenaRačuna na Računu.

CREATE OR REPLACE TRIGGER TB\_UKUPNA\_CENA\_RACUNA

BEFORE INSERT OR UPDATE OR DELETE ON StavkaRacuna

FOR EACH ROW

BEGIN

IF(INSERTING OR UPDATING)

THEN

BEGIN RACUN\_PAKET.postaviID(:NEW.RacunID); END;

ELSE

BEGIN RACUN\_PAKET.postaviID(:OLD.RacunID); END;

END IF;

END;

Procedura za određivanje ukupnog iznosa računa:

CREATE OR REPLACE PROCEDURE UKUPNACENARACUNA(RacunSifra IN NUMBER) AS

suma NUMBER:=0;

BEGIN

SELECT SUM (sr.KolicinaStavkeRacuna \* (m.CenaMaterijala + (m.CenaMaterijala \* sr.PDVStavkeRacuna ) / 100)) INTO suma FROM Materijal m join StavkaRacuna sr on (m.MaterijalID=sr.MaterijalID)

WHERE RacunID=RacunSifra;

UPDATE Racun

SET UkupnaCenaRacuna=suma

WHERE RacunID=RacunSifra;

END;

Triger TA\_UKUPNA\_CENA\_RACUNA: služi za pozivanje procedure. Nakon što se desi insert, update ili delete, poziva se procedura koja nam omogućava izvršavanje logike, odnosno izračunavanje ukupne cene računa.

CREATE OR REPLACE TRIGGER TA\_UKUPNA\_CENA\_RACUNA

AFTER INSERT OR UPDATE OR DELETE ON StavkaRacuna

DECLARE

RacunSifra NUMBER:=RACUN\_PAKET.vratiID();

BEGIN

UKUPNACENARACUNA(RacunSifra);

END;

Triger TBU\_ZABRANA\_UKUPNA\_CENA\_RACUNA: sprečava da se direktno menja UkupnaCenaRačuna u tabeli Račun:

CREATE OR REPLACE TRIGGER TBU\_ZABRANA\_UKUPNA\_CENA\_RACUNA

BEFORE UPDATE OF UkupnaCenaRacuna

ON Racun

FOR EACH ROW

BEGIN

IF NOT (DBMS\_UTILITY.FORMAT\_CALL\_STACK LIKEC '%TA\_UKUPNA\_CENA\_RACUNA%')

THEN

RAISE\_APPLICATION\_ERROR(-20000,'Nije dozvoljena direktna izmena ukupne cene racuna u okviru tabele Racun.');

END IF;

END;

Test primeri:

INSERT INTO StavkaRacuna VALUES (1,1,2,20,0,1);

INSERT INTO StavkaRacuna VALUES (2,3,2,20,0,1);

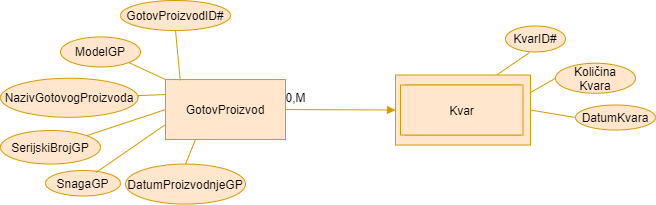
INSERT INTO StavkaRacuna VALUES (3,1,2,20,0,1);

Pregled svih podataka iz tabele Račun kako bismo videli vrednosti kolone UkupnaCenaRačuna:

SELECT \* FROM RACUN;

**2.Repeating Single Detail with Master tehnika optimizacije**

Konceptualni model pre optimizacije:



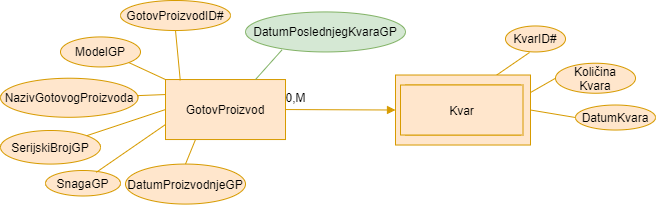
Slika Konceptualni model pre primene Repeating Single Detail with Master tehnike optimizacije

Relacioni model pre optimizacije:

**GotovProizvod** (GotovProizvodID, NazivGotovogProizvoda, SnagaGotovogProizvogotoda, SerijskiBrojGotovogProizvoda, ModelGotovogProizvoda, DatumProizvodnjeGotovogProizvoda)

**Kvar** (KvarID, *GotovProizvodID*, KoličinaKvara, DatumKvara, *TipKvaraID, TipUzrokaKvaraID*)

Konceptualni model pre optimizacije:



Slika Konceptualni model nakon primene Repeating Single Detail with Master tehnike optimizacije

Relacioni model nakon optimizacije:

**GotovProizvod** (GotovProizvodID, NazivGotovogProizvoda, SnagaGotovogProizvoda, SerijskiBrojGotovogProizvoda, ModelGotovogProizvoda, DatumProizvodnjeGotovogProizvoda, DatumPoslednjegKvaraGotovogProizvoda)

**Kvar** (KvarID, *GotovProizvodID*, KoličinaKvara, DatumKvara, *TipKvaraID, TipUzrokaKvaraID*)

Tabela specifikacije trigera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba da uradi? |
| GotovProizvod | Insert |  | NE |  |
| Update | DatumPoslednjeg  KvaraGP | DA | Zabrana direktnog ažuriranja polja DatumPoslednjegKvaraGP u tabeli GotovProizvod |
| Delete |  | NE |  |
| Kvar | Insert |  | DA | Prilikom unosa novog Kvara, poziva se procedura za ažuriranje polja DatumPoslednjegKvaraGP na GotovomProizvodu |
| Update | DatumKvara | DA | Prilikom ažuriranja novog Kvara, poziva se procedura za ažuriranje polja DatumPoslednjegKvaraGP na GotovomProizvodu |
| Delete |  | DA | Prilikom brisanja Kvara, poziva se procedura za ažuriranje polja DatumPoslednjegKvaraGP na GotovomProizvodu |

Dodavanje kolone DatumPoslednjegKvaraGotovogProizvoda u tabelu GotovProizvod:

ALTER TABLE GOTOVPROIZVOD ADD DatumPoslednjegKvaraGotovogProizvoda DATE;

Kreiranje paketa GotovProizvod\_Paket:

CREATE OR REPLACE PACKAGE GOTOVPROIZVOD\_PAKET AS

PROCEDURE postaviID(GPSifra NUMBER);

FUNCTION vratiID RETURN NUMBER;

END GOTOVPROIZVOD\_PAKET;

Kreiranje tela paketa GotovProizvod\_Paket:

CREATE OR REPLACE PACKAGE BODY GOTOVPROIZVOD\_PAKET IS

GotovProizvodID NUMBER;

PROCEDURE postaviID(GPSifra NUMBER) IS

BEGIN

GotovProizvodID:=GPSifra;

END;

FUNCTION vratiID RETURN NUMBER IS

BEGIN

RETURN GotovProizvodID;

END;

END;

Triger TB\_DATUM\_POSLEDNJEG\_KVARA: Svaki put kada se menja nešto u tabeli Kvar, hoćemo da se to odrazi i na jak objekat GotovProizvod, odnosno da to upišemo u polje Datum Poslednjeg Kvara Gotovog Proizvoda na GotovomProizvodu.

CREATE OR REPLACE TRIGGER TB\_DATUM\_POSLEDNJEG\_KVARA

BEFORE INSERT OR UPDATE OR DELETE ON Kvar

FOR EACH ROW

BEGIN

IF(INSERTING OR UPDATING)

THEN

BEGIN GOTOVPROIZVOD\_PAKET.postaviID(:NEW.GotovProizvodID); END;

ELSE

BEGIN GOTOVPROIZVOD\_PAKET.postaviID(:OLD.GotovProizvodID); END;

END IF;

END;

Procedura DATUMPOSLEDNJEGKVARA za odredjivanje datuma poslednjeg kvara:

CREATE OR REPLACE PROCEDURE DATUMPOSLEDNJEGKVARA(GPSifra IN NUMBER) AS

datumpk KVAR.DATUMKVARA%TYPE;

BEGIN

BEGIN

SELECT MAX(DatumKvara) INTO datumpk

FROM Kvar

WHERE GotovProizvodID=GPSifra AND DatumKvara<=sysdate;

EXCEPTION

WHEN no\_data\_found THEN datumpk:=NULL;

END;

UPDATE GotovProizvod

SET DatumPoslednjegKvaraGotovogProizvoda=datumpk

WHERE GotovProizvodID=GPSifra;

END DATUMPOSLEDNJEGKVARA;

Triger TA\_DATUM\_POSLEDNJEG\_KVARA: služi za pozivanje procedure. Nakon što se desi insert, update ili delete, poziva se procedura koja nam omogućava izvršavanje logike, odnosno izračunavanje poslednjeg datuma kvara gotovog proizvoda.

CREATE OR REPLACE TRIGGER TA\_DATUM\_POSLEDNJEG\_KVARA

AFTER INSERT OR UPDATE OR DELETE ON Kvar

DECLARE

GPSifra NUMBER:=GOTOVPROIZVOD\_PAKET.vratiID();

BEGIN

DATUMPOSLEDNJEGKVARA(GPSifra);

END;

Triger TBU\_DATUM\_POSLEDNJEG\_KVARA: sprečava da se direktno menja Datum Poslednjeg Kvara Gotovog Proizvoda u tabeli GotovProizvod:

CREATE OR REPLACE TRIGGER TBU\_DATUM\_POSLEDNJEG\_KVARA

BEFORE UPDATE OF DatumPoslednjegKvaraGotovogProizvoda

ON GotovProizvod

FOR EACH ROW

BEGIN

IF NOT (DBMS\_UTILITY.FORMAT\_CALL\_STACK LIKEC '%TA\_DATUM\_POSLEDNJEG\_KVARA%')

THEN

RAISE\_APPLICATION\_ERROR(-20000,'Nije dozvoljena direktna izmena datuma poslednjeg kvara u okviru tabele GotovProizvod.');

END IF;

END;

Test primer:

INSERT INTO Kvar VALUES (2,1,1,TO\_DATE('05/12/2022','DD/MM/YYYY'),1,1);

INSERT INTO Kvar VALUES (3,1,1,TO\_DATE('06/12/2022','DD/MM/YYYY'),1,1);

INSERT INTO Kvar VALUES (4,1,1,TO\_DATE('07/12/2022','DD/MM/YYYY'),1,1);

Pregled svih podataka iz tabele GotovProizvod kako bismo videli vrednosti kolone Datum Poslednjeg Kvara Gotovog Proizvoda:

SELECT \* FROM GotovProizvod;

# TEHNOLOGIJE ZA IMPLEMENTACIJU PROJEKTA

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

Za implementaciju baze podataka korišćen je Oracle 19c Enterprise Edition sistem za upravljanje bazama podataka. Ovaj sistem pruža veliki izbor alata za manipulaciju podacima i kreiranje svih stvari neophodnih za rad – trigeri, procedure, funkcije, paketi.

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

Aplikacija je razvijena u NetBeans-u 15, u Java programskom jeziku. NetBeans je integrisano razvojno okruženje koje je namenjeno pre sve svega razvoju Java tehnologija. Prilikom razvoja aplikacije korišćen je MVC (Model - View - Controller) patern koji podrazumeva da se komunikacija između formi i baze obavlja preko kontrolera. Putem formi je omogućena osnovna manipulacija podacima – prikaz, dodavanje, izmena i brisanje.