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

|  |  |  |  |
| --- | --- | --- | --- |
| **IME I PREZIME** | | **Jana Mitrović** | |
| Broj indeksa | | **2022/3012** | |
| Studijski modul | | Informacioni sistemi | |
| Stečena diploma osnovnih akademski studija | | Diplomirani inženjer organizacionih nauka | |
| **NAZIV PROJEKTA** | | Informacioni sistem za evidenciju podataka o poreskim kontrolama | |
| Napomena | |  | |
| *Verzija* | *Datum promene* | | *Opis promene* | |
| 1 | 09.11.2022. | | **Prijava teme** | |
| 2 | 17.11.2022. | | **PMOV i Relacioni model** | |
| 3 | 01.12.2022. | | **Denormalizacija relacija** | |
| 4 | 08.12.2022. | | **KDT i implementacija trigera** | |
| 5 | 15.12.2022. | | **Optimizacija baze podataka** | |
| 6 | 09.01.2023. | | **Implementacija aplikacije** | |

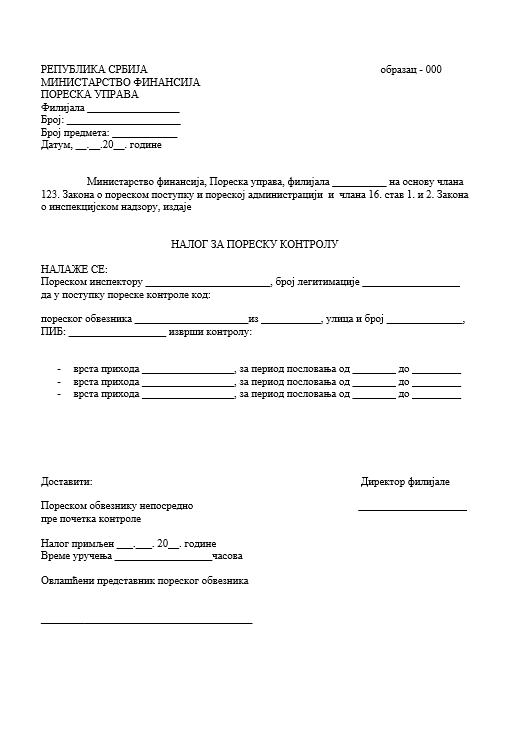
# OPIS PROJEKTA

## Opis podsistema za koji se projektuje baza podataka

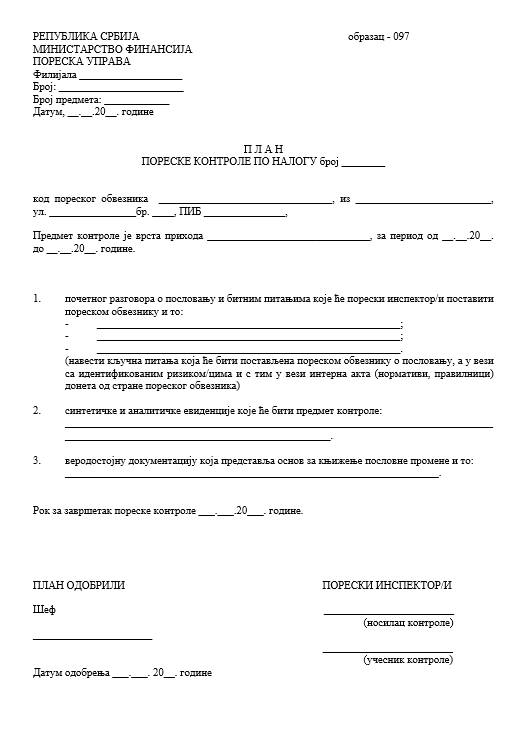
|  |
| --- |
| U okviru svake filijale poreske uprave ministarstva finansija vrše se poreske kontrole. Jedna poreska kontrola se odnosi na jednog poreskog obveznika koji može imati više zaposlenih. Postupak vršenja poreske kontrole započinje izdavanjem **naloga za kontrolu** jednom ili više inspektora od strane šefa. Nakon izdavanja naloga inspektor kreira **plan kontrole** i vrši samu kontrolu. Po završenoj kontroli kreira se **zapisnik** koji sadrži sve značajne informacije vezane za odrađenu kontrolu. Na samom kraju na osnovu zapisnika se donosi **rešenje** u kome se navode radnje koje treba preduzeti i po potrebi **rešenje o oduzimanju robe** koje se odnosi na robu koju je neophodno oduzeti od poreskog obveznika. |

## Poslovna dokumenta koja se koriste u procesu

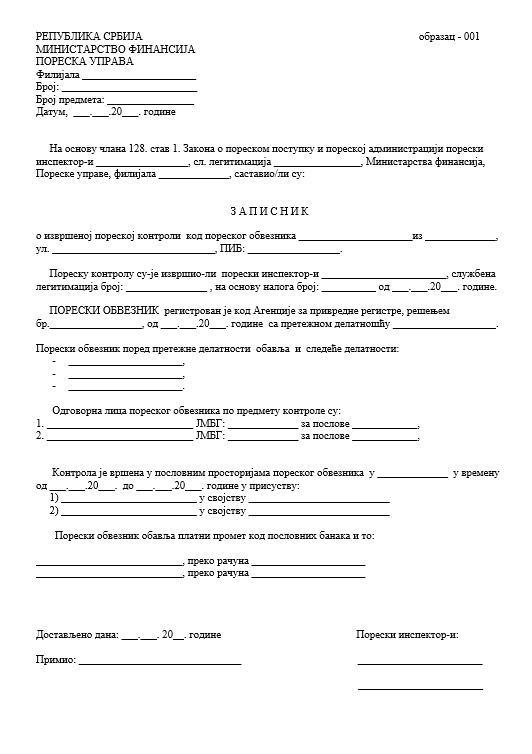
*Nalog*



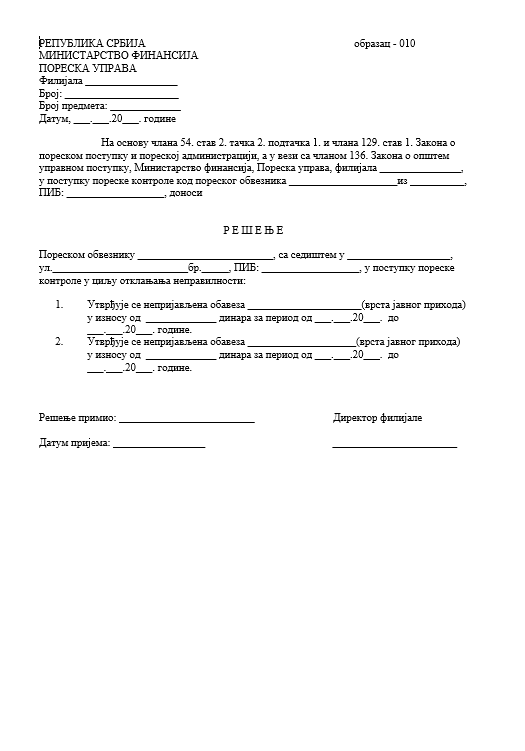
*Plan*



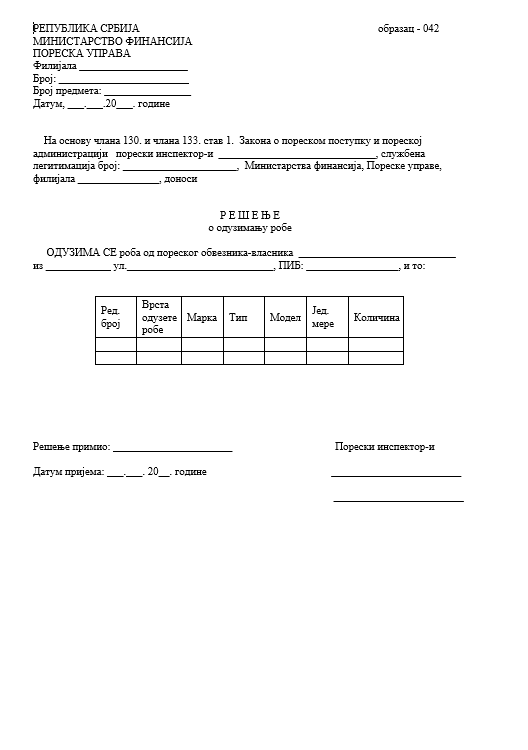
*Zapisnik*

**

*Rešenje*

**

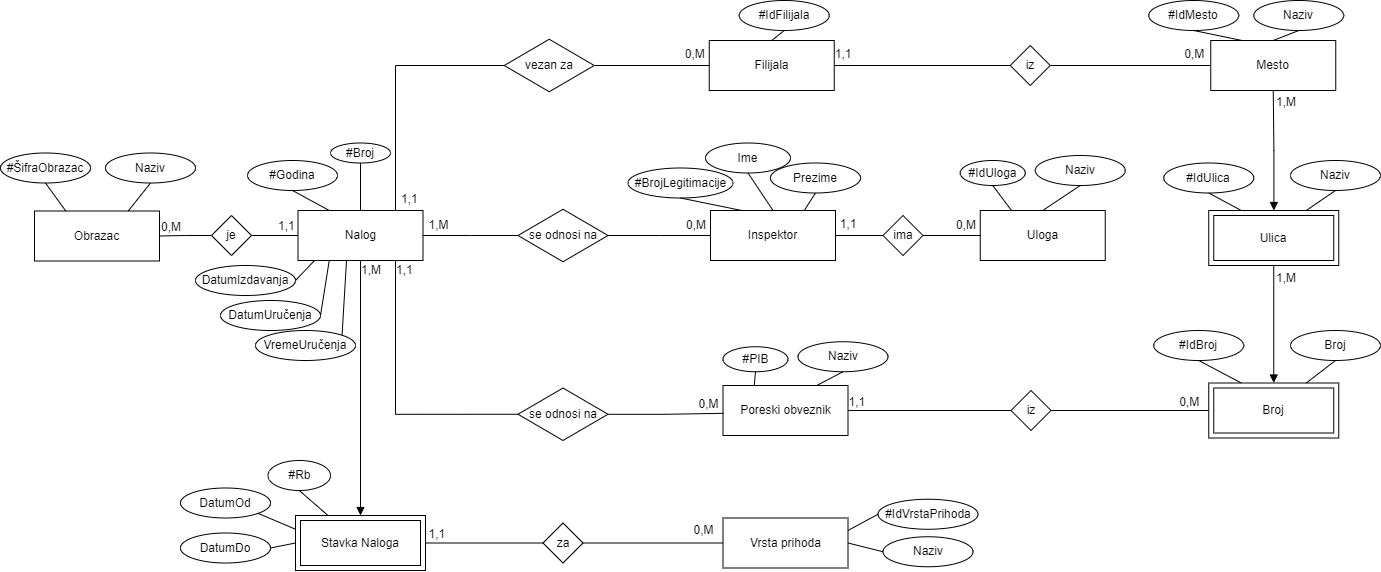
*Rešenje o oduzimanju robe*

**

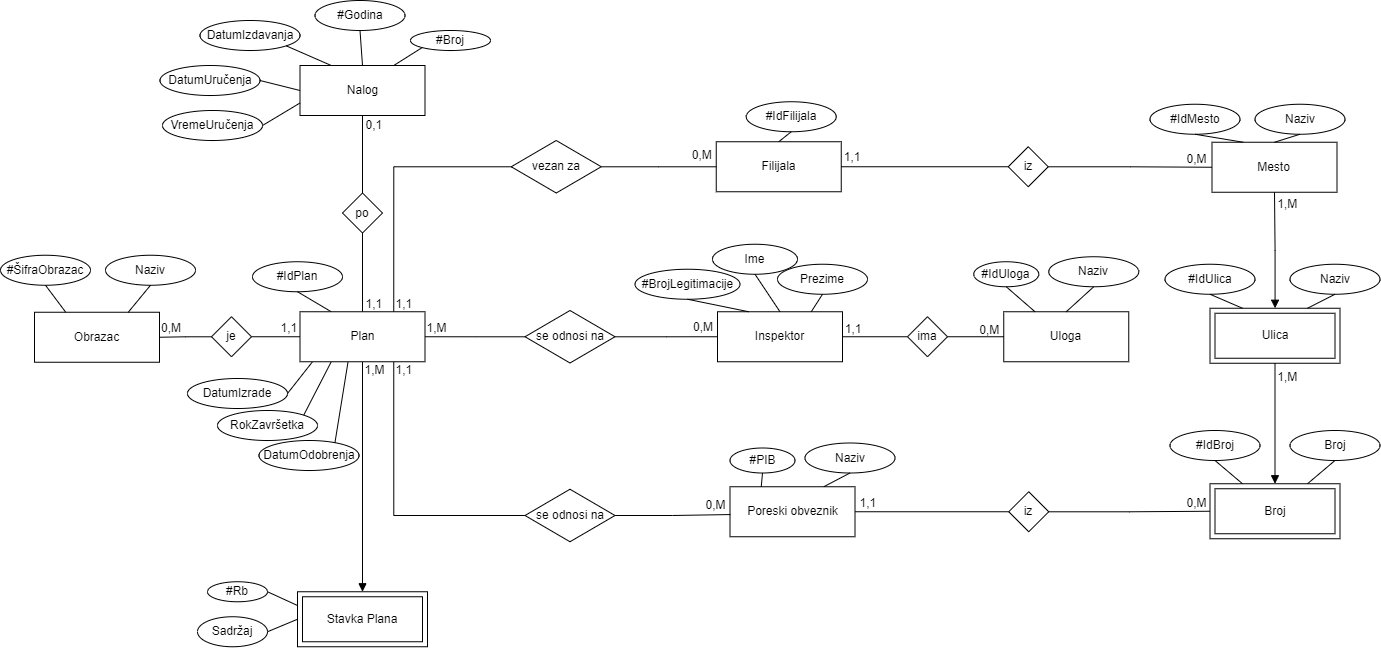
# PROJEKTOVANJE BAZE PODATAKA

## Model podatka

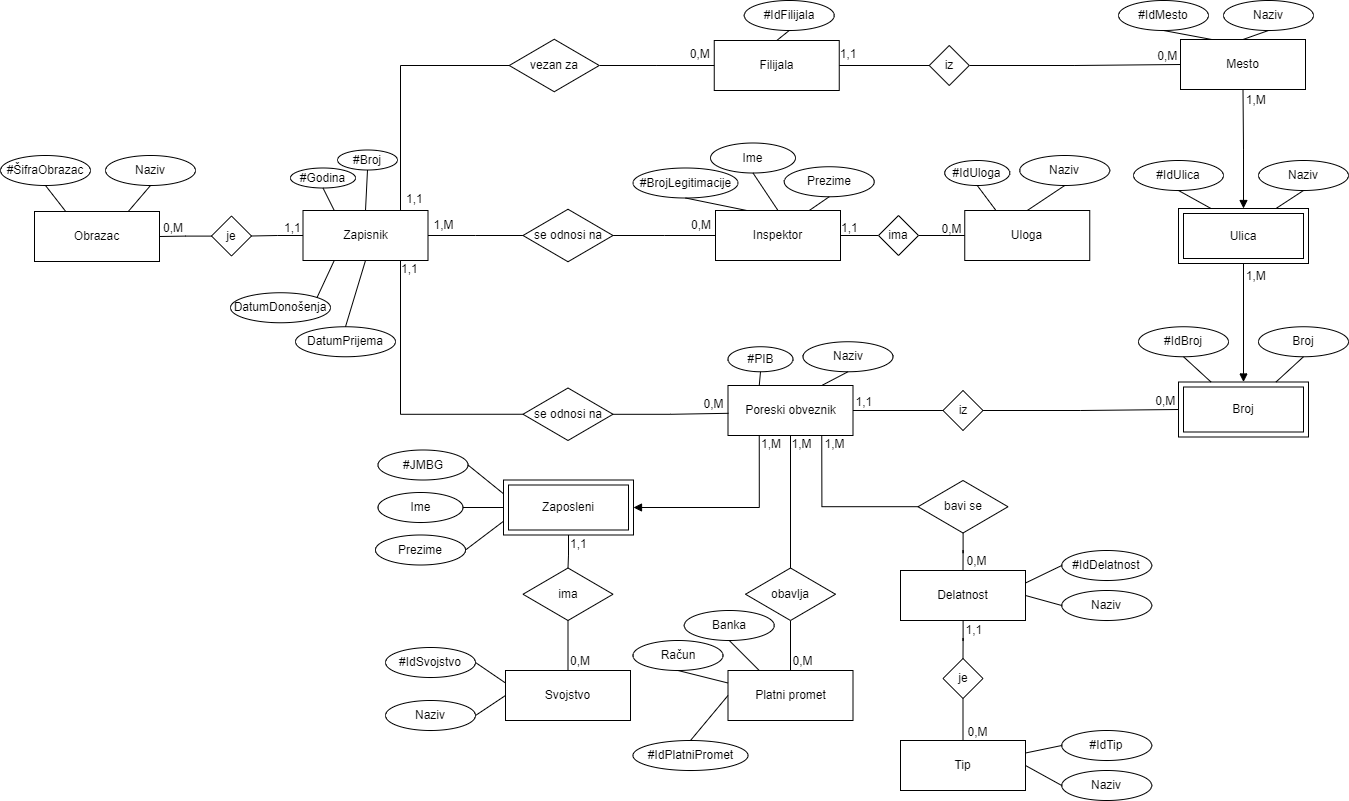
*Nalog*



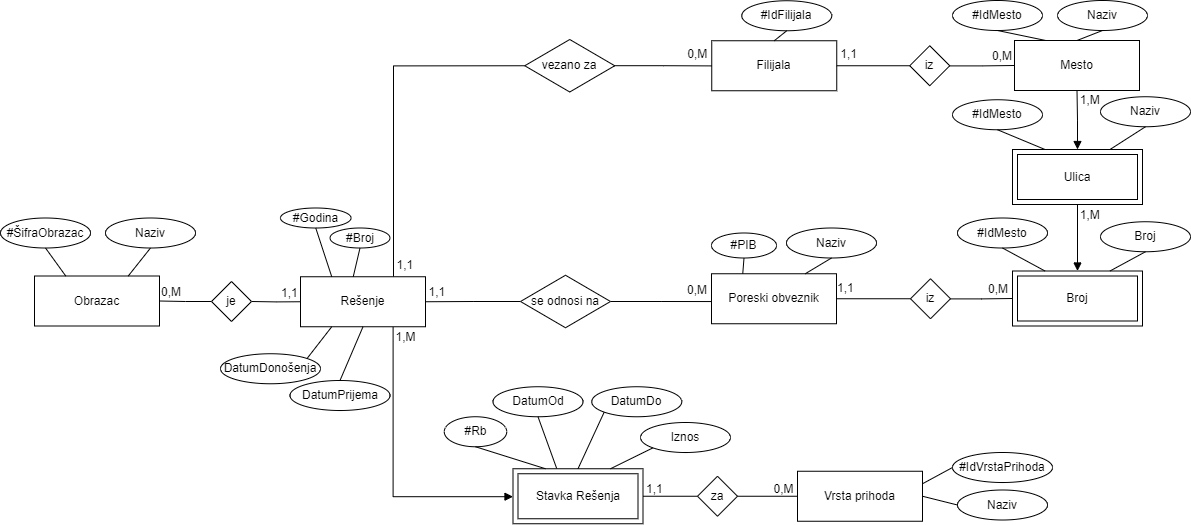
*Plan*



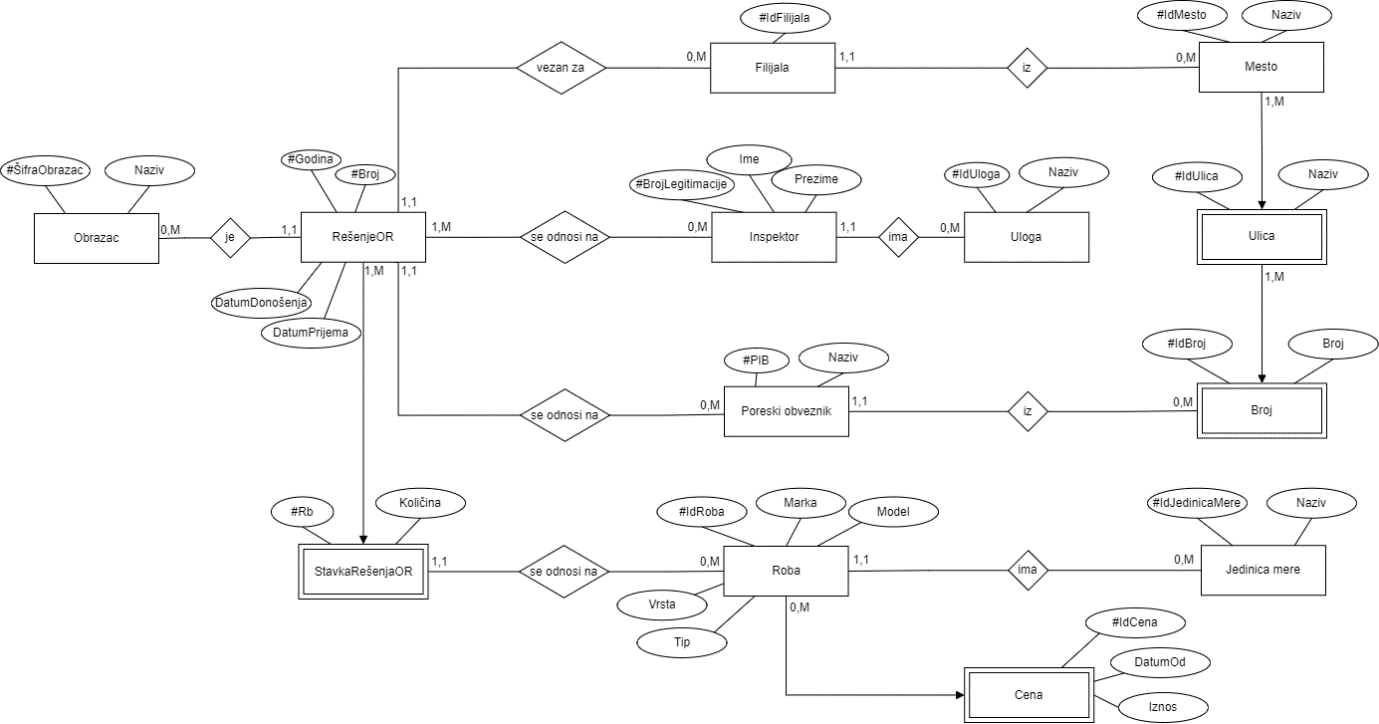
*Zapisnik*



*Rešenje*



*Rešenje o oduzimanju robe*



## Relacioni model

**Mesto**(IdMesto, Naziv)

**Ulica**(IdMesto, IdUlica, Naziv)

**Broj**(IdMesto, IdUlica, IdBroj, Broj)

**Filijala**(IdFilijala, *IdMesto*)

**PoreskiObveznik**(PIB, Naziv, *IdMesto, IdUlica, IdBroj*)

**Uloga**(IdUloga, Naziv)

**Inspektor**(BrojLegitimacije, Ime, Prezime, *IdUloga*)

**Obrazac**(ŠifraObrazac, Naziv)

**Nalog**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumIzdavanja, DatumUručenja, VremeUručenja*, PIB*)

**VrstaPrihoda**(IdVrstaPrihoda, Naziv)

**StavkaNaloga**(Rb, *Broj, Godina,* DatumOd, DatumDo, *IdVrstaPrihoda*)

**Plan**(IdPlan, *Broj, Godina, IdFilijala*, *ŠifraObrazac,* DatumIzrade, RokZavršetka, DatumOdobrenja*, PIB*)

**StavkaPlana**(Rb, *IdPlan*, Sadržaj)

**Zapisnik**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumDonošenja, DatumPrijema*, PIB*)

**Svojstvo**(IdSvojstvo, Naziv)

**Zaposleni**(PIB, JMBG, Ime, Prezime, *IdSvojstvo*)

**PlatniPromet**(IdPlatniPromet, Banka, Račun)

**Tip**(IdTip, Naziv)

**Delatnost**(IdDelatnost, Naziv, *IdTip*)

**Rešenje**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumDonošenja, DatumPrijema*, PIB*)

**StavkaRešenja**(Rb, *Broj, Godina*, DatumOd, DatumDo, Iznos, *IdVrstaPrihoda*)

**RešenjeOR**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumDonošenja, DatumPrijema*, PIB*)

**JedinicaMere**(IdJedinicaMere, Naziv)

**Roba**(IdRoba, Vrsta, Marka, Model, Tip, *IdJedinicaMere*)

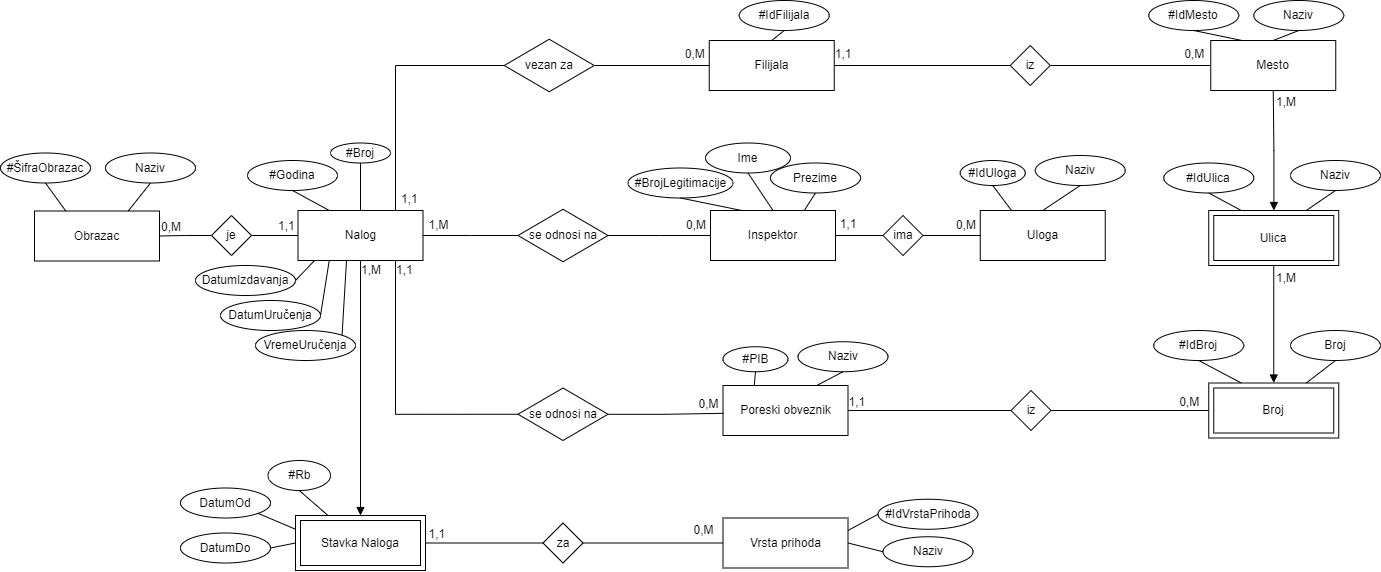
**Cena**(IdCena, *IdRoba,* DatumOd, Iznos)

**StavkaRešenjaOR**(Rb, *Broj, Godina,* Količina, *IdRoba*)

# DENORMALIZACIJA RELACIJA

## Denormalizacija relacija uz narušavanje 2NF

***Normalizovani model podataka***

****

**Nalog**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumIzdavanja, DatumUručenja, VremeUručenja, *PIB*)

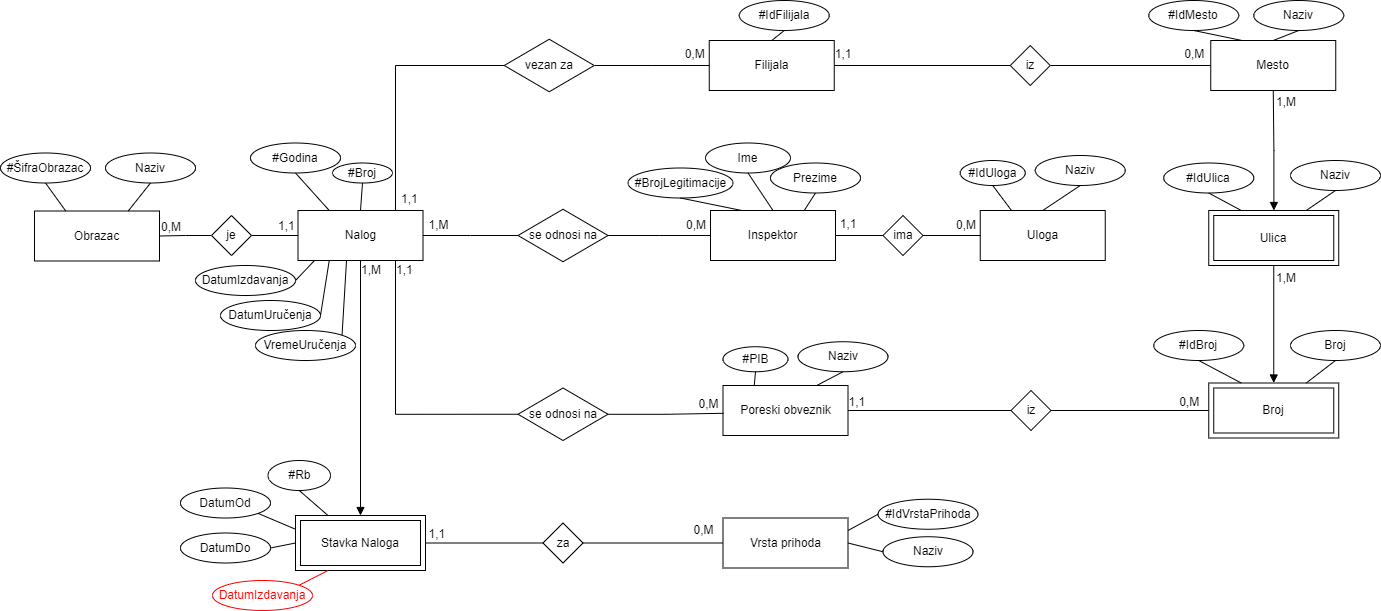
**StavkaNaloga**(Rb, *Broj, Godina,* DatumOd, DatumDo, *IdVrstaPrihoda*)

***Funkcionalne zavisnosti pre denormalizacije***

Broj, Godina 🡪 IdFilijala, ŠifraObrazac, DatumIzdavanja, DatumUručenja, VremeUručenja, PIB

Rb, Broj, Godina 🡪 DatumOd, DatumDo, IdVrstaPrihoda

***Denormalizovani model podataka***



**Nalog**(Broj, Godina, *IdFilijala*, *ŠifraObrazac,* DatumIzdavanja, DatumUručenja, VremeUručenja*, PIB*)

**StavkaNaloga**(Rb, *Broj, Godina*, DatumOd, DatumDo, DatumIzdavanja, *IdVrstaPrihoda*)

***Funkcionalne zavisnosti posle denormalizacije***

Broj, Godina 🡪 IdFilijala, ŠifraObrazac, DatumIzdavanja, DatumUručenja, VremeUručenja, PIB

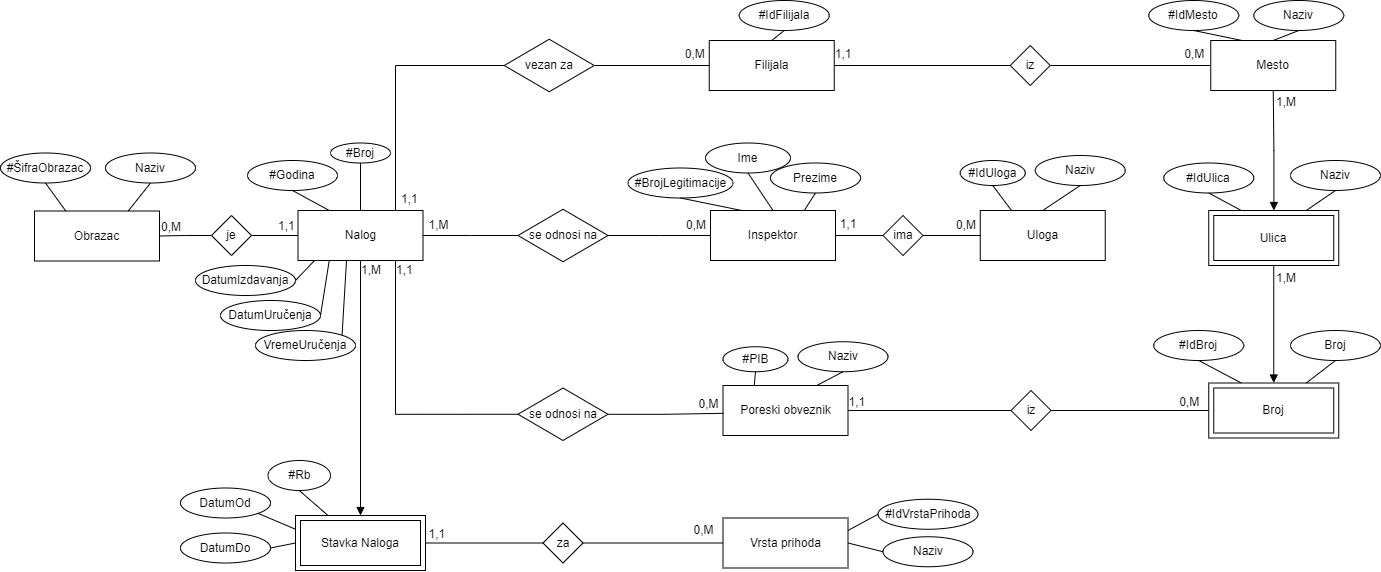
Rb, Broj, Godina 🡪 DatumOd, DatumDo, DatumIzdavanja, IdVrstaPrihoda

***Identifikacija trigera***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba uraditi?** |
| **Nalog** | insert |  | NE |  |
| update | DatumIzdavanja | DA | Prilikom izmene vrednosti kolone DatumIzdavanja u tabeli Nalog, pokreće se triger koji izmenjenu vrednost ažurira u tabeli StavkaNaloga |
| delete |  | NE |  |
| **StavkaNaloga** | insert |  | DA | U kolonu DatumIzdavanja tabele StavkaNaloga upisuje odgovarajuću vrednost za unetu vrednost kolona Broj i Godina |
| update | DatumIzdavanja | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |

## Denormalizacija relacija uz narušavanje 3NF

***Normalizovani model***



**Nalog**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumIzdavanja, DatumUručenja, VremeUručenja, *PIB*)

**Filijala**(IdFilijala, *IdMesto*)

**Mesto**(IdMesto, Naziv)

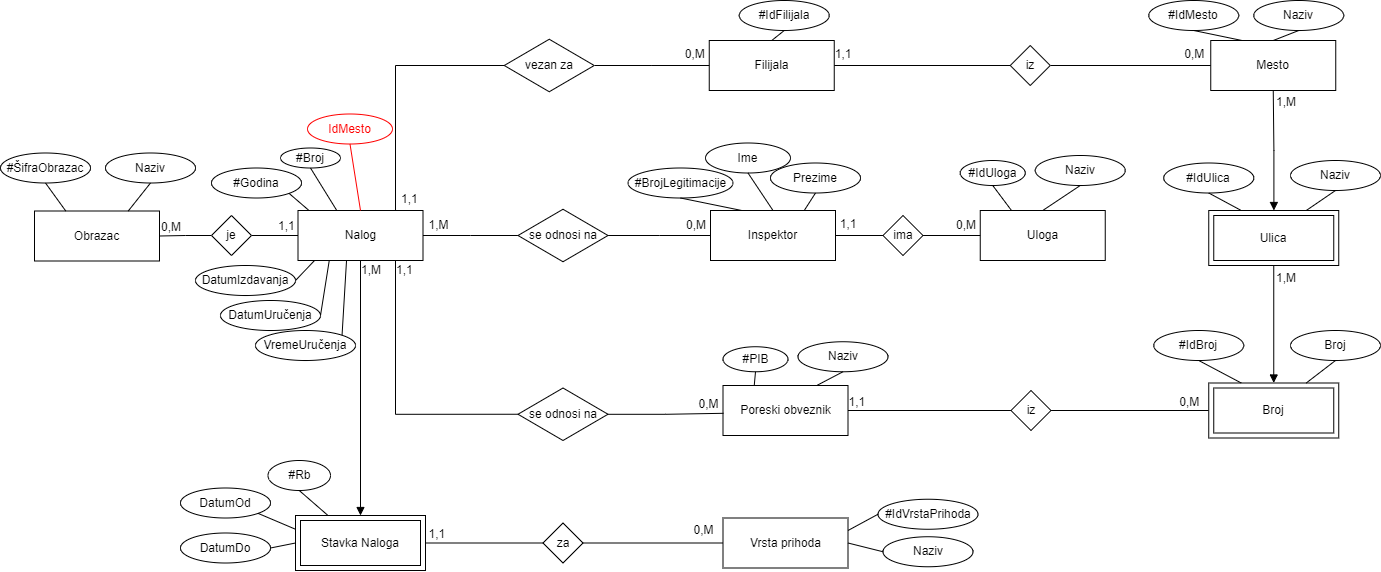
***Funkcionalne zavisnosti pre denormalizacije***

Broj, Godina 🡪 IdFilijala, ŠifraObrazac, DatumIzdavanja, DatumUručenja, VremeUručenja, IdFilijala, PIB

IdFilijala 🡪 IdMesto

IdMesto 🡪 Naziv

***Denoramlizovani model – Short-circuit keys***



**Nalog**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumIzdavanja, DatumUručenja, VremeUručenja, *PIB, IdMesto*)

**Filijala**(IdFilijala, *IdMesto*)

***Funkcionalne zavisnosti pre denormalizacije***

Broj, Godina 🡪 IdFilijala, ŠifraObrazac, DatumIzdavanja, DatumUručenja, VremeUručenja, IdFilijala, PIB, IdMesto

IdFilijala 🡪 IdMesto

***Identifikacija trigera***

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

# KORISNIČKI DEFINISANI TIPOVI I TRIGERI

## Definicija korisničkom tipa i njegovo korišćenje

**Distinct tip – Racun**

CREATE OR REPLACE TYPE Racun AS OBJECT(

brojRacuna NUMBER(20),

MEMBER FUNCTION get\_brojRacuna RETURN NUMBER,

CONSTRUCTOR FUNCTION Racun(brojRacuna NUMBER)

RETURN SELF AS RESULT)

INSTANTIABLE NOT FINAL;

CREATE OR REPLACE TYPE BODY Racun AS

MEMBER FUNCTION get\_brojRacuna RETURN NUMBER IS

BEGIN

RETURN SELF.brojRacuna;

END;

CONSTRUCTOR FUNCTION Racun(brojRacuna NUMBER)

RETURN SELF AS RESULT

AS

BEGIN

IF brojRacuna >= 100000000000000000 AND brojRacuna < 1000000000000000000

THEN SELF.brojRacuna := brojRacuna;

ELSE

BEGIN

RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Broj racuna mora imati tacno 18 cifara!');

END;

END IF;

RETURN;

END;

END;

**Naredba za kreiranje tabele Platni Promet**

CREATE TABLE platniPromet(

IdPlatniPromet NUMBER(9) PRIMARY KEY,

Banka VARCHAR2(30),

Racun Racun);

**Naredba za unos podataka u tabelu Poreski Obveznik**

INSERT INTO platniPromet VALUES (1, 'Unicredit Bank', Racun(100236154895214782));

**Naredba za prikaz podataka iz tabele Poreski Obveznik**

SELECT pp.IdPlatniPromet AS Id, pp.Banka, pp.Racun. get\_brojRacuna() AS Broj\_racuna FROM platniPromet pp;

**Naredba za izmenu podataka u tabeli Poreski Obveznik**

UPDATE platniPromet pp SET pp.Racun.brojRacuna = 123652199947511366 WHERE pp.IdPlatniPromet = 1;

**Naredba za brisanje podataka iz tabele Poreski Obveznik**

DELETE platniPromet pp WHERE pp.Racun.brojRacuna = 123652199947511366;

**Struktuirani tip – NazivPO**

## CREATE OR REPLACE TYPE NazivPO AS OBJECT(

## ime VARCHAR2(20),

## tip VARCHAR2(10),

## MEMBER FUNCTION get\_ime RETURN VARCHAR2,

## MEMBER FUNCTION get\_tip RETURN VARCHAR2,

## CONSTRUCTOR FUNCTION NazivPO(ime VARCHAR2, tip VARCHAR2)

## RETURN SELF AS RESULT)

## INSTANTIABLE NOT FINAL;

## CREATE OR REPLACE TYPE BODY NazivPO 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 NazivPO(ime VARCHAR2, tip VARCHAR2)

## RETURN SELF AS RESULT

## AS

## BEGIN

## SELF.ime := ime;

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

## THEN BEGIN

## RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Tip preduzaca moze uzati samo neku od sledecih vrednosti: doo, ad, od, kd!');

## END;

## ELSE SELF.tip := tip;

## END IF;

## RETURN;

## END;

## END;

**Naredba za kreiranje tabele Poreski Obveznik**

CREATE TABLE poreskiObveznik(

PIB NUMBER(9) PRIMARY KEY,

IdMesto NUMBER(10) NOT NULL,

IdUlica NUMBER(10) NOT NULL,

IdBroj NUMBER(10) NOT NULL,

naziv NazivPO,

CONSTRAINT adresa\_fk FOREIGN KEY (IdMesto, IdUlica, IdBroj) REFERENCES

Broj(IdMesto, IdUlica, IdBroj));

**Naredba za unos podataka u tabelu Poreski Obveznik**

INSERT INTO poreskiObveznik VALUES (123456789, 1, 1, 1, NazivPO('Atlantic Štark', 'doo'));

**Naredba za prikaz podataka iz tabele Poreski Obveznik**

SELECT po.PIB, po.naziv.get\_ime() AS NazivPO, po.naziv.get\_tip() AS TipPreduzeca FROM poreskiObveznik po;

**Naredba za izmenu podataka u tabeli Poreski Obveznik**

UPDATE poreskiObveznik po SET po.naziv.ime = 'Štark' WHERE po.PIB = 123456789;

**Naredba za brisanje podataka iz tabele Poreski Obveznik**

DELETE poreskiObveznik po WHERE po.naziv.get\_ime() = 'Štark';

## Realizacija proceduralne logike uz pomoć trigera

## **Trigeri koji se odnose na naraušavanje 2NF**

**INSERT StavkaNaloga 🡪 Na osnovu unetih kolona Broj i Godina upisuje odgovarajuću vrednost kolone DatumIzdavanja**

CREATE OR REPLACE TRIGGER INS\_STAVKA\_NALOGA

BEFORE INSERT ON StavkaNaloga

FOR EACH ROW

DECLARE v\_datumIzdavanja DATE;

BEGIN

SELECT DatumIzdavanja INTO v\_datumIzdavanja FROM Nalog WHERE Broj =: NEW.Broj AND Godina = :NEW.Godina;

:NEW.DatumIzdavanja := v\_datumIzdavanja;

END;

**UPDATE DatumIzdavanja u StavkaNaloga 🡪 Zabrana direktne izmene kolone DatumIzdavanja u tabeli StavkaNaloga**

CREATE OR REPLACE TRIGGER UPD\_DATUM\_ZABRANA

BEFORE UPDATE OF DatumIzdavanja ON StavkaNaloga

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Nije moguce direktno menjati Datum Izdavanja!');

END;

**UPDATE DatumIzdavanja u Nalog 🡪 Ažuriranje kolone DatumIzdavanja u tabeli StavkaNaloga nakon izmene vrednosti kolone DatumIzdavanja u tabeli Nalog**

CREATE OR REPLACE TRIGGER UPD\_DATUM\_IZDAVANJA

AFTER UPDATE OF DatumIzdavanja ON Nalog

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_DATUM\_ZABRANA DISABLE';

UPDATE StavkaNaloga

SET DatumIzdavanja = :NEW.DatumIzdavanja

WHERE Broj = :NEW.Broj AND Godina = :NEW.Godina;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_DATUM\_ZABRANA ENABLE';

END;

END;

**Trigeri koji se odnose na naraušavanje 3NF**

**INSERT Nalog 🡪 Unosi vrednost kolone IdMesto na osnovu unete vrednosti kolone IdFilijala**

CREATE OR REPLACE TRIGGER INS\_NALOG

BEFORE INSERT ON Nalog

FOR EACH ROW

DECLARE v\_mesto NUMBER(10);

BEGIN

SELECT IdMesto INTO v\_mesto FROM Filijala WHERE IdFilijala = :NEW.IdFilijala;

:NEW.IdMesto := v\_mesto;

END;

**UPDATE IdFilijala u Nalog 🡪 Na osnovu nove vrednosti kolone IdFilijala ažurira vrednost kolone IdMesto**

CREATE OR REPLACE TRIGGER UPD\_FILIJALA

BEFORE UPDATE OF IdFilijala ON Nalog

FOR EACH ROW

DECLARE

v\_novo\_mesto NUMBER(10);

BEGIN

SELECT IdMesto INTO v\_novo\_mesto FROM Filijala WHERE IdFilijala = :NEW.IdFilijala;

:NEW.IdMesto := v\_novo\_mesto;

END;

**UPDATE IdMesto u Nalog 🡪 Zabrana direktne izmene kolone IdMesto u tabeli Nalog**

CREATE OR REPLACE TRIGGER UPD\_MESTO\_ZABRANA

BEFORE UPDATE OF IdMesto ON Nalog

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(NUM => -20000, MSG => 'Nije moguca direktno menjati IdMesto!');

END;

**UPDATE IdMesto u Filijala 🡪 Na osnovu nove vrednosti kolone IdMesto u tabeli Filijala ažurira vrednost kolone IdMesto u tabeli Nalog koji se odnosi na tu Filijalu**

CREATE OR REPLACE TRIGGER UPD\_MESTO

AFTER UPDATE OF IdMesto ON Filijala

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_MESTO\_ZABRANA DISABLE';

UPDATE Nalog

SET IdMesto = :NEW.IdMesto

WHERE IdFilijala = :NEW.IdFilijala;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_DATUM\_ZABRANA ENABLE';

END;

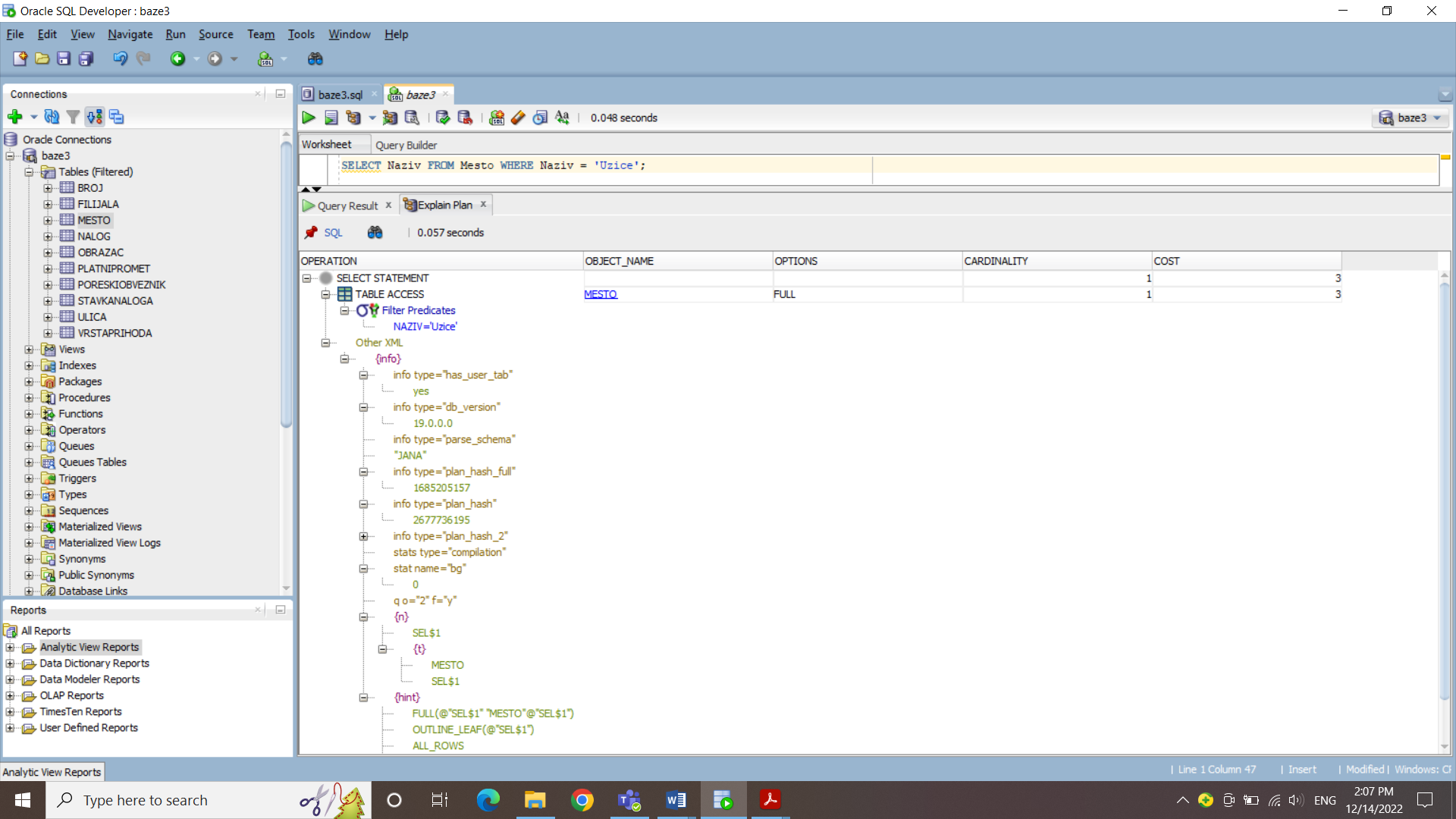
END;

# OPTIMIZACIJA BAZE PODATAKA

## Definisanje i analiza indeksa

**Definisanje indeksa nad tekstualnim poljem**

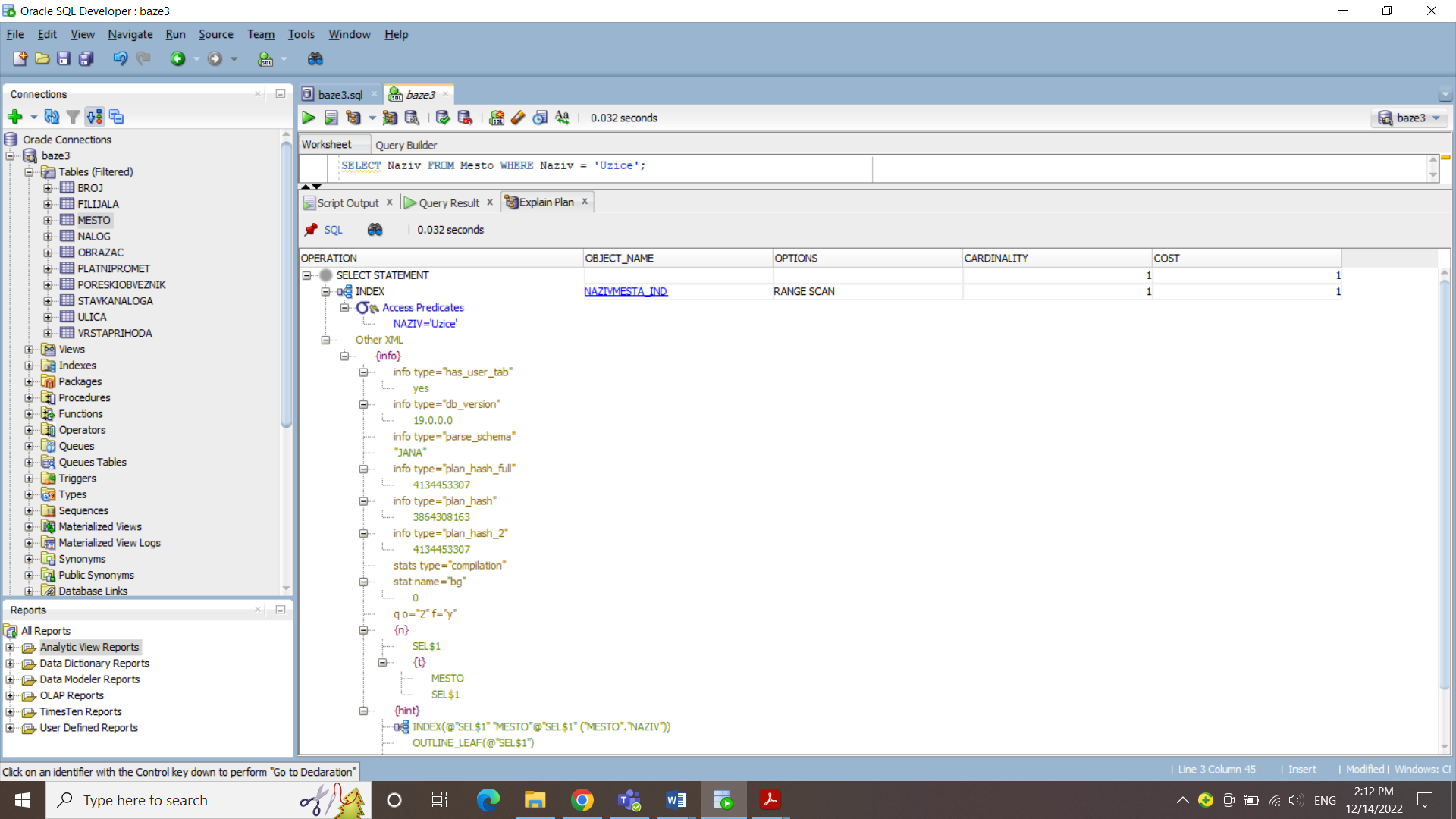
Plan izvršenja pre definisanja indeksa:



Definisanje indeksa:

CREATE INDEX NazivMesta\_IND ON Mesto(Naziv);

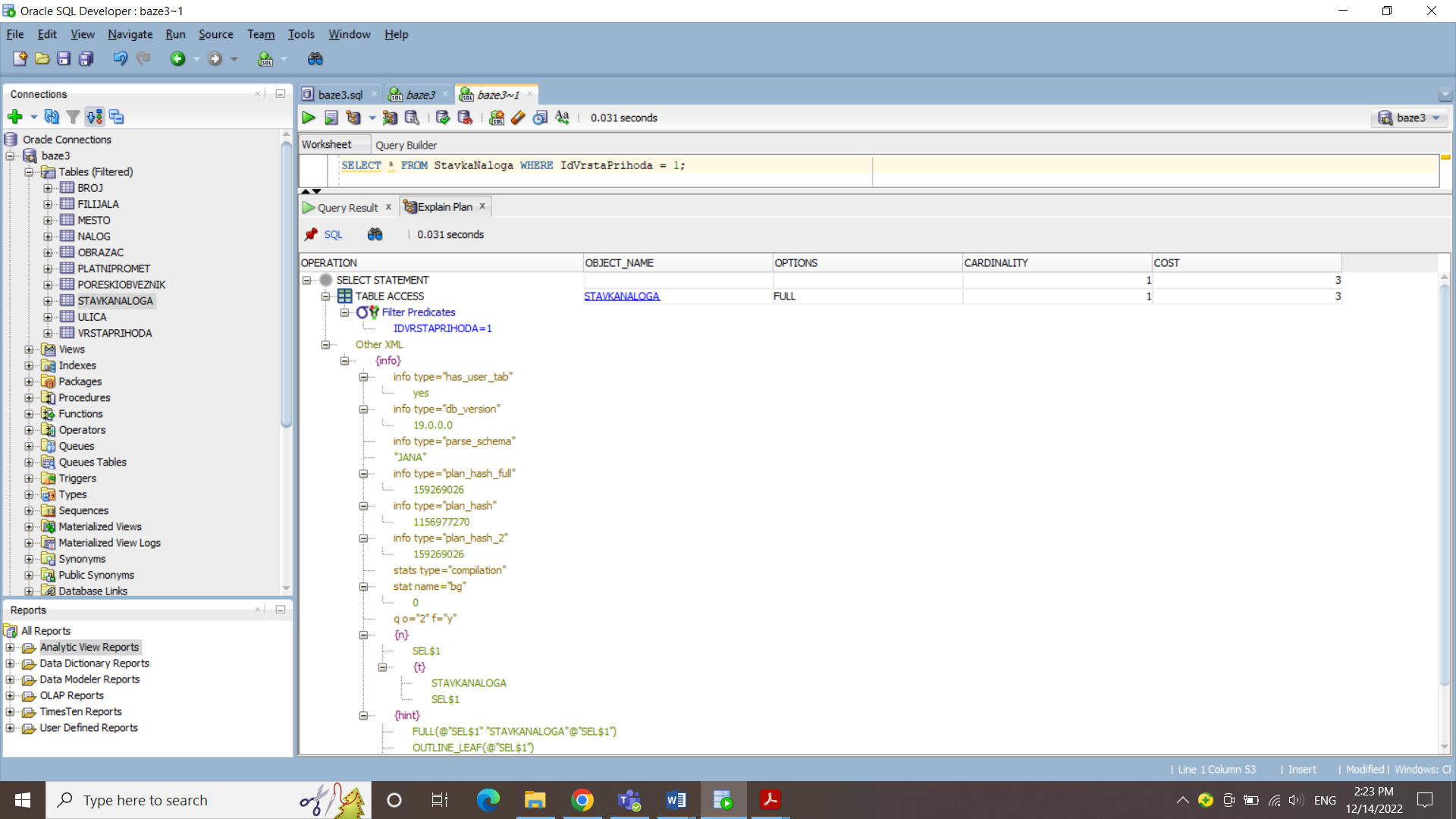
Plan izvršenja nakon definisanja indeksa:

––

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

**Definisanje indeksa nad spoljnim ključem**

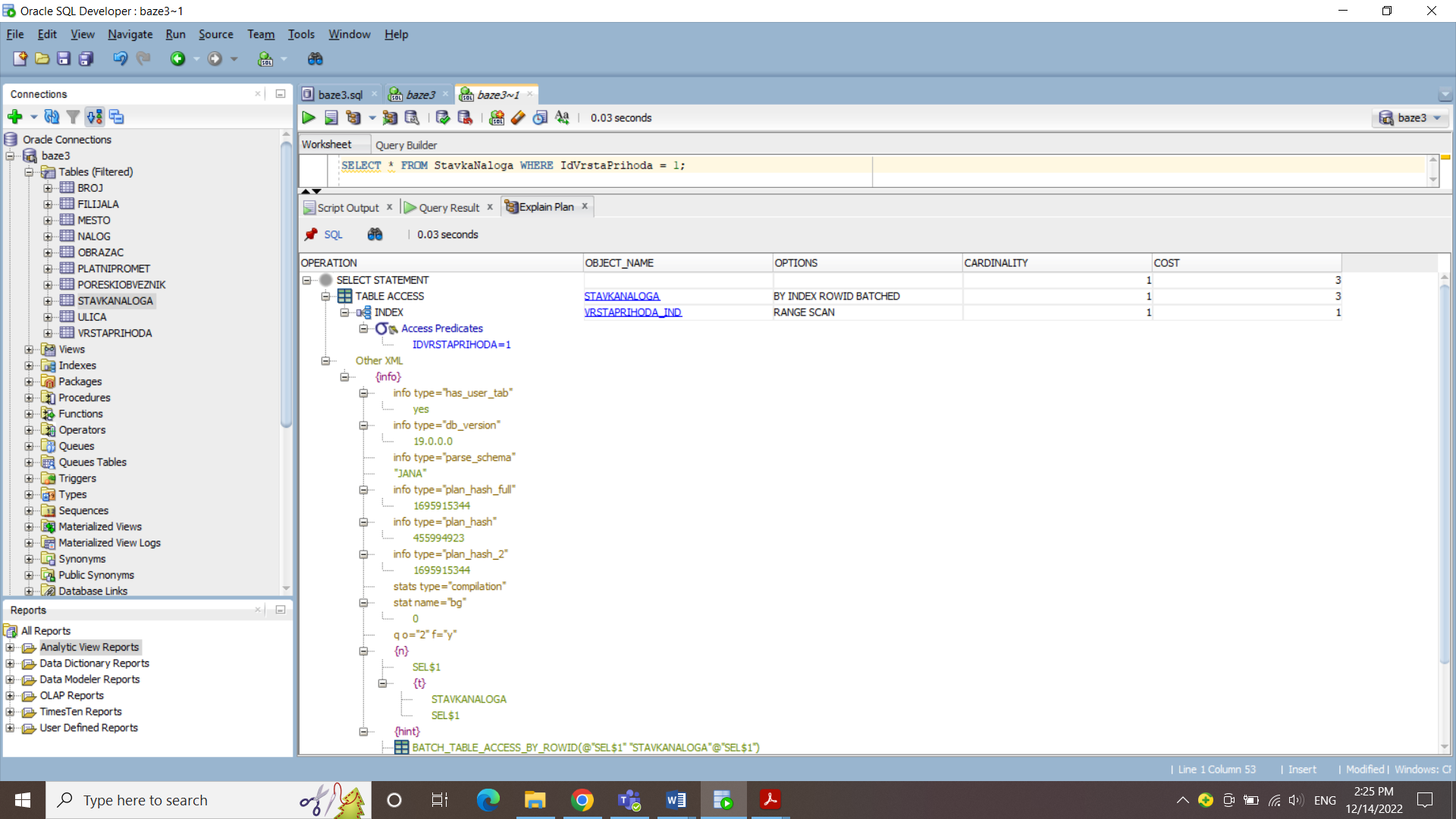
Plan izvršenja pre definisanja indeksa:



Definisanje indeksa:

CREATE INDEX VrstaPrihoda\_IND ON StavkaNaloga(IdVrstaPrihoda);

Plan izvršenja nakon definisanja indeksa:



## 

Ovaj indeks ubrzava upite koji u WHERE kluzuli imaju kolonu IdVrstaPrihoda iz tabele VrstaPrihoda.

## Horizontalno particionisanje

Horizontalno particionisanje je odrađeno na tabeli Rešenje, gde je izvršena podela na četiri particije koje se odnose na kvartale u toku godine na osnovu kolone DatumDonošenja.

CREATE TABLE Resenje(

Broj NUMBER(10) NOT NULL,

Godina NUMBER(4) NOT NULL,

IdFilijala NUMBER(10) NOT NULL,

SifraObrazac VARCHAR2(3) NOT NULL,

DatumDonosenja DATE,

DatumPrijema DATE,

PIB NUMBER(9) NOT NULL,

UkupanIznos NUMERIC(30),

CONSTRAINT filijala\_r\_fk FOREIGN KEY (IdFilijala) REFERENCES Filijala(IdFilijala),

CONSTRAINT obrazac\_r\_fk FOREIGN KEY (SifraObrazac) REFERENCES Obrazac(SifraObrazac),

CONSTRAINT po\_r\_fk FOREIGN KEY (PIB) REFERENCES poreskiObveznik(PIB),

CONSTRAINT resenje\_pk PRIMARY KEY (Broj, Godina))

PARTITION BY RANGE (DatumDonosenja)(

PARTITION prvi\_kvartal2022 VALUES LESS THAN(TO\_DATE('04/01/2022','MM/DD/YYYY')),

PARTITION drugi\_kvartal2022 VALUES LESS THAN(TO\_DATE('07/01/2022','MM/DD/YYYY')),

PARTITION treci\_kvartal2022 VALUES LESS THAN(TO\_DATE('10/01/2022','MM/DD/YYYY')),

PARTITION cetvrti\_kvartal2022 VALUES LESS THAN(TO\_DATE('01/01/2023','MM/DD/YYYY'))

);

**Naredba za prikaz rešenja iz prvog kvartala 2022 godine:**

SELECT \* FROM Resenje PARTITION(prvi\_kvartal2022);

## 

## Vertikalno particionisanje

**Zapisnik**(Broj, Godina, *IdFilijala*, *ŠifraObrazac*, DatumDonošenja, DatumPrijema*, PIB*)

**Zapisnik\_osnovno**(Broj, Godina, *IdFilijala, PIB*)

**Zapisnik\_detalji**(Broj, Godina, *ŠifraObrazac*, DatumDonošenja, DatumPrijema)

**Kreiranje pogleda**

CREATE OR REPLACE VIEW Zapisnik AS

SELECT zo.Broj, zo.Godina, zo.IdFilijala, zo.PIB, zd.SifraObrazac, zd.DatumDonosenja, zd.DatumPrijema

FROM Zapisnik\_osnovno zo, Zapisnik\_detalji zd

WHERE zo.Broj = zd.Broj AND zo.Godina = zd.Godina;

**Unos Zapisnika**

CREATE OR REPLACE TRIGGER INSTEAD\_INS\_ZAPISNIK

INSTEAD OF INSERT ON Zapisnik

FOR EACH ROW

BEGIN

INSERT INTO Zapisnik\_osnovno(Broj, Godina, IdFilijala, PIB)

VALUES (:NEW.Broj, :NEW.Godina, :NEW.IdFilijala, :NEW.PIB);

INSERT INTO Zapisnik\_detalji(Broj, Godina, SifraObrazac, DatumDonosenja, DatumPrijema)

VALUES (:NEW.Broj, :NEW.Godina, :NEW.SifraObrazac, :NEW.DatumDonosenja,:NEW.DatumPrijema);

END;

**Izmena zapisnika**

CREATE OR REPLACE TRIGGER INSTEAD\_UPD\_ZAPISNIK

INSTEAD OF UPDATE ON Zapisnik

FOR EACH ROW

BEGIN

UPDATE Zapisnik\_osnovno

SET IdFilijala = :NEW.IdFilijala, PIB = :NEW.PIB

WHERE Broj = :NEW.Broj AND Godina = :NEW.Godina;

UPDATE Zapisnik\_detalji

SET SifraObrazac = :NEW.SifraObrazac, DatumDonosenja = :NEW.DatumDonosenja, DatumPrijema = :NEW.DatumPrijema

WHERE Broj = :NEW.Broj AND Godina = :NEW.Godina;

END;

**Brisanje zapisnika**

CREATE OR REPLACE TRIGGER INSTEAD\_DEL\_ZAPISNIK

INSTEAD OF DELETE ON Zapisnik

FOR EACH ROW

BEGIN

DELETE FROM Zapisnik\_osnovno WHERE Broj = :OLD.Broj AND Godina = :OLD.Godina;

DELETE FROM Zapisnik\_detalji WHERE Broj = :OLD.Broj AND Godina = :OLD.Godina;

END;

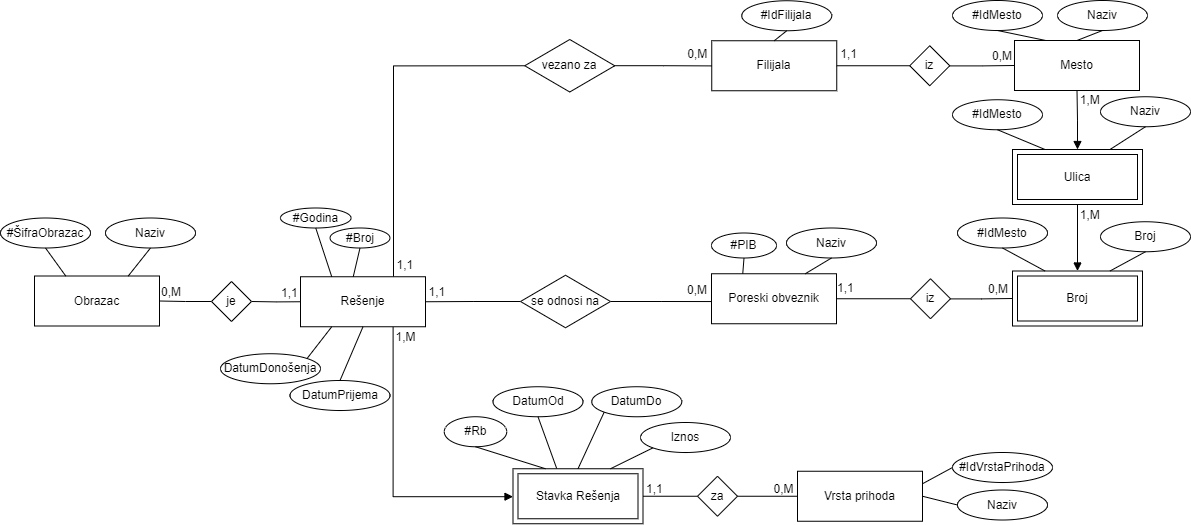
## 

## Primena drugih optimizacionih tehnika

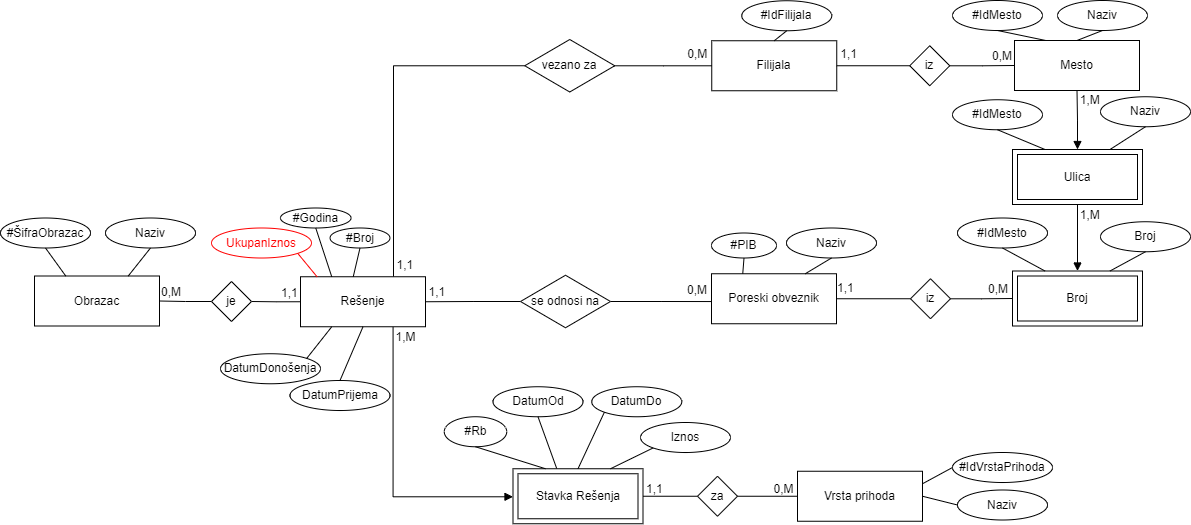
**Storing Derivable Values**

U prvom primeru primenjena je Storing Derivable Values tehnika optimizacije tako što je dodat atribut UkupanIznos u tabelu Rešenje koji predstavlja ukupan iznos svih stavki rešenja. 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?** |
| **Rešenje** | insert |  | NE |  |
| update | UkupanIznos | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |
| **StavkaRešenja** | insert |  | DA | Poziva proceduru koja ažurira vrednost kolone UkupanIznos u tabeli Rešenje |
| update | Iznos | DA |
| delete | Iznos | DA |

**Triger koji postavlja broj i godinu rešenja u promenljivu:**

CREATE OR REPLACE NONEDITIONABLE PACKAGE RStavka AS

Broj NUMBER(10);

Godina NUMBER(4);

END RStavka;

CREATE OR REPLACE TRIGGER UKUPAN\_IZNOS\_RESENJE1

BEFORE INSERT OR UPDATE OR DELETE ON StavkaResenja

FOR EACH ROW

BEGIN

IF (INSERTING OR UPDATING)

THEN

BEGIN

RStavka.Broj := :NEW.Broj;

RStavka.Godina := :NEW.Godina;

END;

ELSE

BEGIN

RStavka.Broj := :OLD.Broj;

RStavka.Godina := :OLD.Godina;

END;

END IF;

END;

**Triger za zabranu direktne izmene atributa UkupanIznos u tabeli Rešenje:**

CREATE OR REPLACE TRIGGER UPD\_RESENJE\_UKUPNO\_ZABRANA

BEFORE UPDATE OF UkupanIznos ON Resenje

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(NUM => -20000,MSG => 'Ne mozete direktno menjati vrednost UkupanIznos!');

END;

**Procedura koja ukljucuje triger za zabranu direktne izmene kolone UkupanIznos u tabeli Resenje:**

CREATE OR REPLACE PROCEDURE ISKLJUCI\_TRIGER AS

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_RESENJE\_UKUPNO\_ZABRANA DISABLE';

END ISKLJUCI\_TRIGER;

**Procedura koja iskljucuje triger za zabranu direktne izmene kolone UkupanIznos u tabeli Resenje:**

CREATE OR REPLACE PROCEDURE UKLJUCI\_TRIGER AS

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER UPD\_RESENJE\_UKUPNO\_ZABRANA ENABLE';

END UKLJUCI\_TRIGER;

**Procedura koja izračunava Ukupan iznos Rešenja:**

CREATE OR REPLACE PROCEDURE UKUPAN\_IZNOS\_RESENJA

(Br IN NUMBER, God IN NUMBER) AS

Ukupno NUMBER := 0;

BEGIN

SELECT SUM(sr.iznos) INTO Ukupno

FROM StavkaResenja sr

WHERE sr.Broj = Br AND sr.Godina = God;

UPDATE Resenje

SET UkupanIznos = Ukupno

WHERE Broj = Br AND Godina = God;

END;

**Triger za pozivanje procedure:**

CREATE OR REPLACE TRIGGER UKUPAN\_IZNOS\_RESENJE2

AFTER INSERT OR UPDATE OR DELETE ON StavkaResenja

DECLARE

Broj NUMBER(10) := RStavka.Broj;

Godina NUMBER(10) := RStavka.Godina;

BEGIN

ISKLJUCI\_TRIGER();

UKUPAN\_IZNOS\_RESENJA(Broj, Godina);

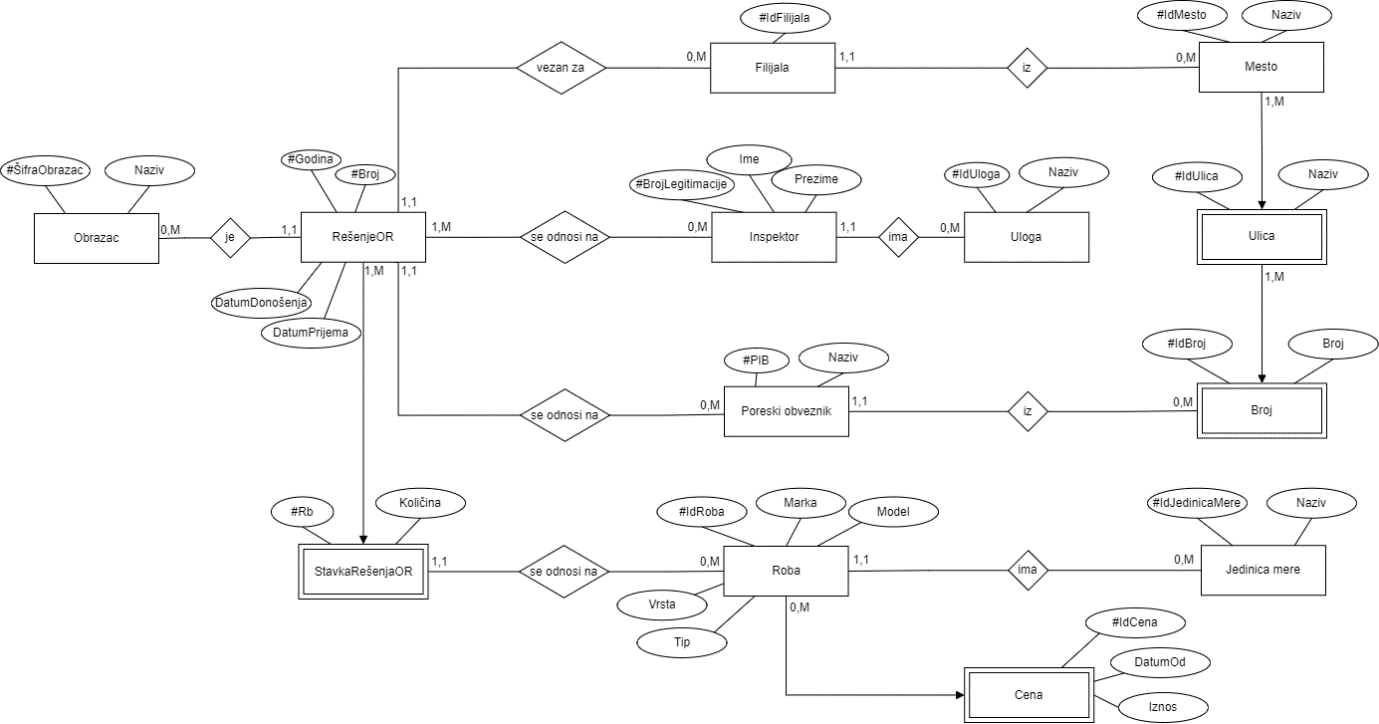
UKLJUCI\_TRIGER();

END;

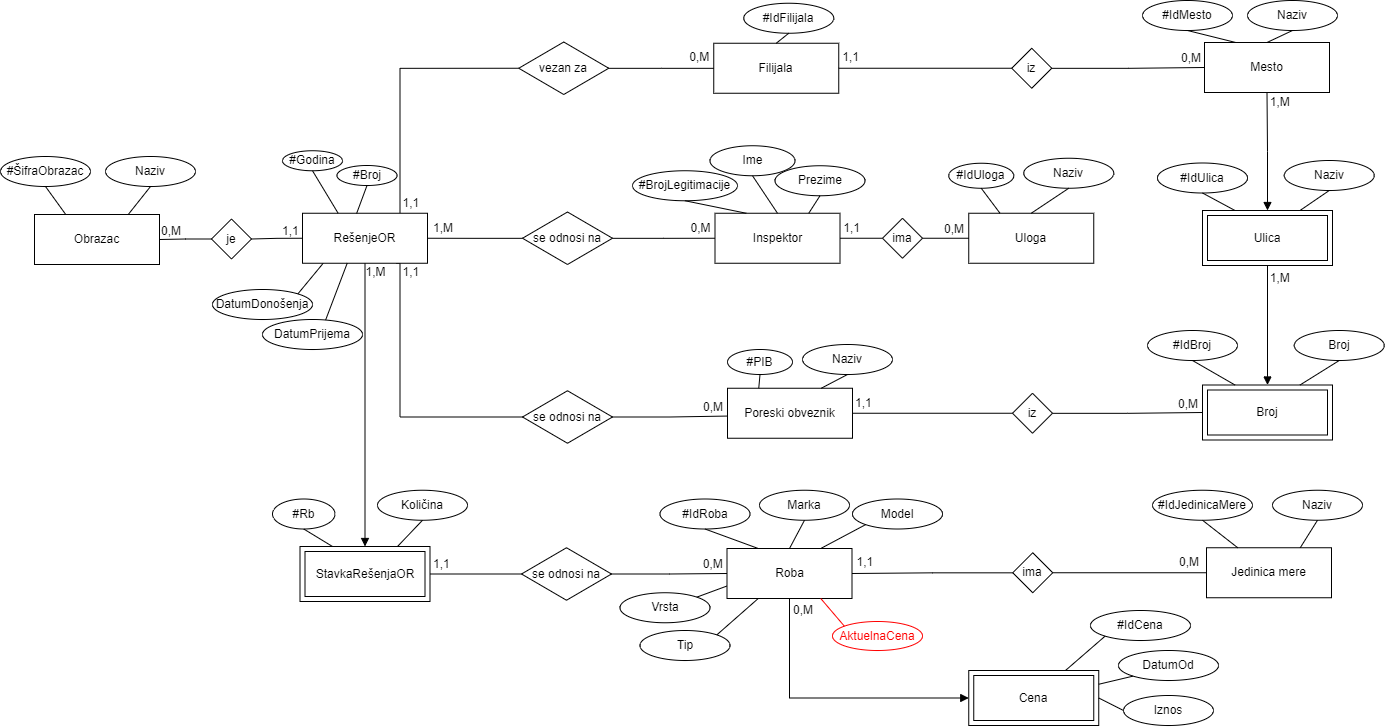
**Repeating Single Detail with Master**

U drugom primeru primenjena je Repeating Single Detail with Mastertehnika optimizacije tako što je tabeli Roba 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?** |
| **Roba** | insert |  | NE |  |
| update | AktuelnaCena | DA | Sprečava direktnu izmenu |
| delete |  | NE |  |
| **Cena** | insert |  | DA | Poziva proceduru koja ažurira vrednost kolone AktuelnaCena u tabeli Roba |
| update | Iznos | DA |
| delete | Iznos | DA |

**Triger koji id robe u promenljivu:**

CREATE OR REPLACE NONEDITIONABLE PACKAGE RobaPaket AS

IdRoba NUMBER(10);

END RobaPaket;

CREATE OR REPLACE TRIGGER ROBA\_CENA1

BEFORE INSERT OR UPDATE OR DELETE ON Cena

FOR EACH ROW

BEGIN

IF (INSERTING OR UPDATING)

THEN

BEGIN

RobaPaket.IdRoba := :NEW.IdRoba;

END;

ELSE

BEGIN

RobaPaket.IdRoba := :OLD.IdRoba;

END;

END IF;

END;

**Triger za zabranu direktne izmene atributa AktuelnaCena u tabeli Roba:**

CREATE OR REPLACE TRIGGER UPD\_AKTUELNA\_CENA\_ZABRANA

BEFORE UPDATE OF AktuelnaCena ON Roba

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(NUM => -20000,MSG => 'Ne mozete direktno menjati vrednost AktuelnaCena!');

END;

**Procedura koja ukljucuje triger za zabranu direktne izmene kolone AktuelnaCena u tabeli Roba:**

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;

**Procedura koja iskljucuje triger za zabranu direktne izmene kolone AktuelanCena u tabeli Roba:**

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;

**Procedura koja ažurira Aktuelnu cenu Robe:**

CREATE OR REPLACE PROCEDURE AKTUELNA\_CENA\_ROBE

(Roba IN NUMBER) AS

Aktuelna NUMBER := 0;

BEGIN

BEGIN

SELECT Iznos INTO Aktuelna FROM Cena

WHERE IdRoba = Roba AND DatumOd=(SELECT MAX(DatumOd) FROM Cena WHERE IdRoba = Roba AND DatumOd <= SYSDATE);

EXCEPTION WHEN no\_data\_found THEN Aktuelna := null;

END;

UPDATE Roba

SET AktuelnaCena = Aktuelna

WHERE IdRoba = Roba;

END;

**Triger za pozivanje procedure:**

CREATE OR REPLACE TRIGGER ROBA\_CENA2

AFTER INSERT OR UPDATE OR DELETE ON Cena

DECLARE

Roba NUMBER(10) := RobaPaket.IdRoba;

BEGIN

ISKLJUCI\_TRIGER\_CENA();

AKTUELNA\_CENA\_ROBE(Roba);

UKLJUCI\_TRIGER\_CENA();

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.