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

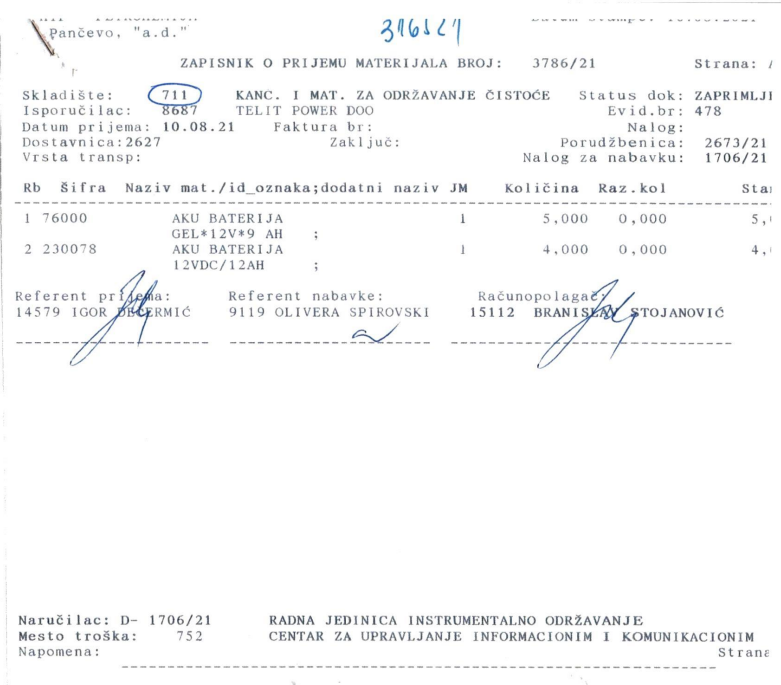
|  |  |  |  |
| --- | --- | --- | --- |
| **IME I PREZIME** | | **Ljubica Ilić** | |
| Broj indeksa | | **2023/3022** | |
| Studijski modul | | Informacioni sistemi | |
| Stečena diploma osnovnih akademskih studija | | Diplomirani inženjer organizacionih nauka | |
| **NAZIV PROJEKTA** | | Proces skladištenja ulaznih i izlaznih materijala unutar preduzeća „HIP-Petrohemija“ d.o.o Pančevo | |
| Napomena | |  | |
| *Verzija* | *Datum promene* | | *Opis promene* | |
| 1 | 16.11.2023. | | **PMOV i Relacioni model** | |
| 2 | 30.11.2023. | | **Realizacija proceduralne logike uz pomoć trigera** | |
|  |  | |  | |
|  |  | |  | |
|  |  | |  | |
|  |  | |  | |

# OPIS PROJEKTA

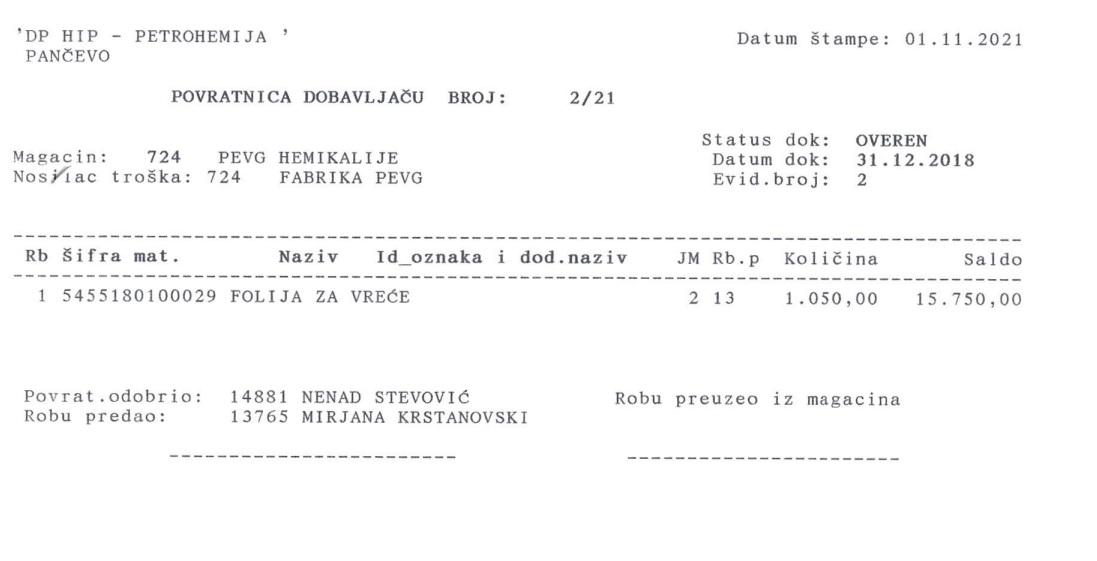
## Opis podsistema za koji se projektuje baza podataka

|  |
| --- |
| Podsistem za koji se projektuje baza podataka obuhvata proces prijema i skladištenja sirovina, hemikalija i materijala potrebnih za proces proizvodnje, kao i skladištenje gotovih proizvoda i njihova otprema kupcu. Magacini su podeljeni na ulazne i izlazne magacine.  Kada su u pitanju ulazni magacini, prijem ulaznih materijala se vrši na osnovu dostavnice dobijene od dobavljača zajedno sa pristiglom robom, formirane na osnovu poslate porudžbenice naše kompanije. Tom prilikom se izrađuje zapisnik o prijemu materijala, čime se jasno definiše magacin u koji se raspoređuju dostavljeni materijali. Ukoliko je došlo do oštećenja prilikom transporta, moguće je kreirati povratnicu dobavljaču za tu vrstu i količinu proizvoda koja je oštećena. Izlaz robe iz ulaznih magacina vrši se na osnovu trebovanja iz organizacionih jedinica kompanije. U slučaju da nije sva istrebovana roba iskorišćena od strane određene organizacione jedinice, nju je moguće vratiti u magacin uz kreiranje dokumenta pod nazivom povratnica materijala. Ukoliko je potrebno, moguće je i premeštanje materijala iz jednog u drugi magacin koje prati zapisnik o primopredaji.  Kada su u pitanju izlazni magacini, prijem gotove robe iz proizvodnje radi se dnevno, po smenama, na osnovu zapisnika o ulazu gotove robe u magacin. Otprema robe iz magacina se obavlja preko otpremnice, koja se kreira na osnovu naloga za otpremu robe dobijene od strane službe prodaje. |

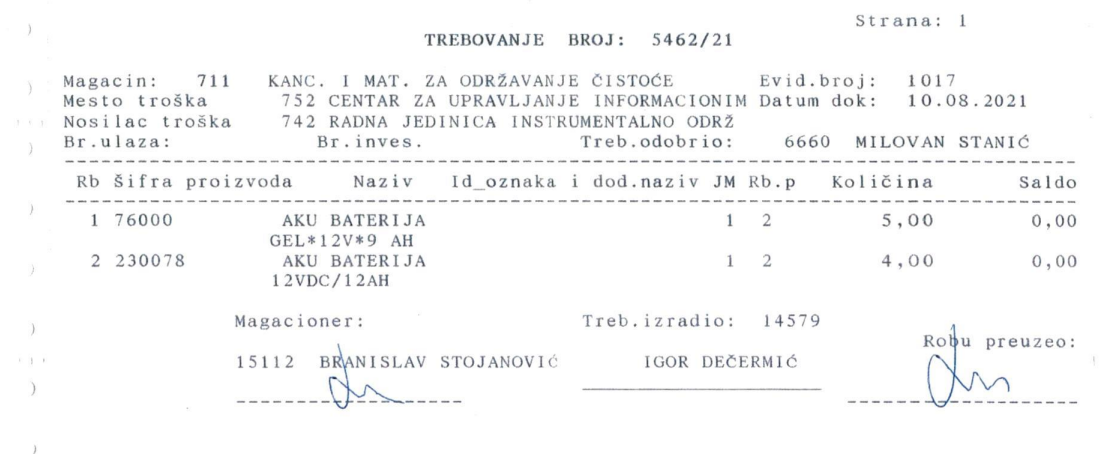
## Poslovna dokumenta koja se koriste u procesu



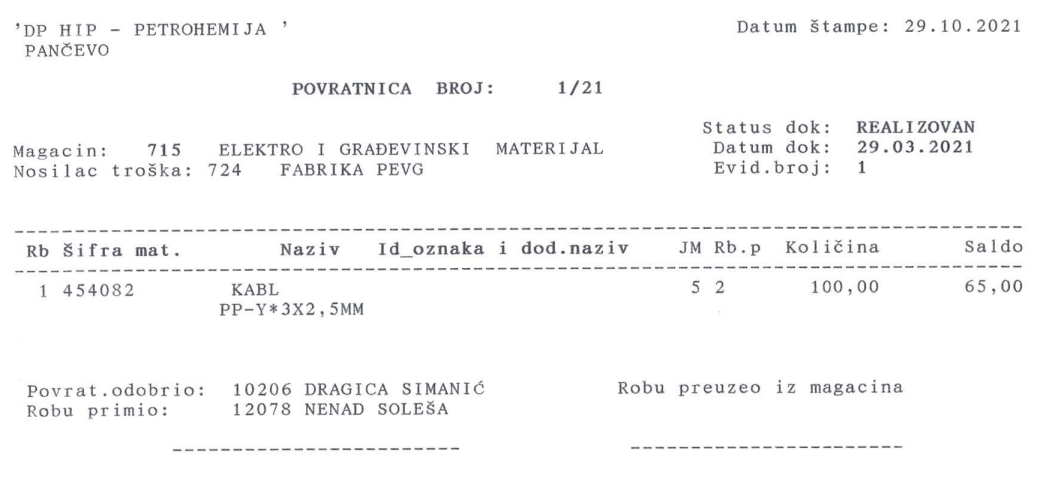
Slika 1: Zapisnik o prijemu materijala



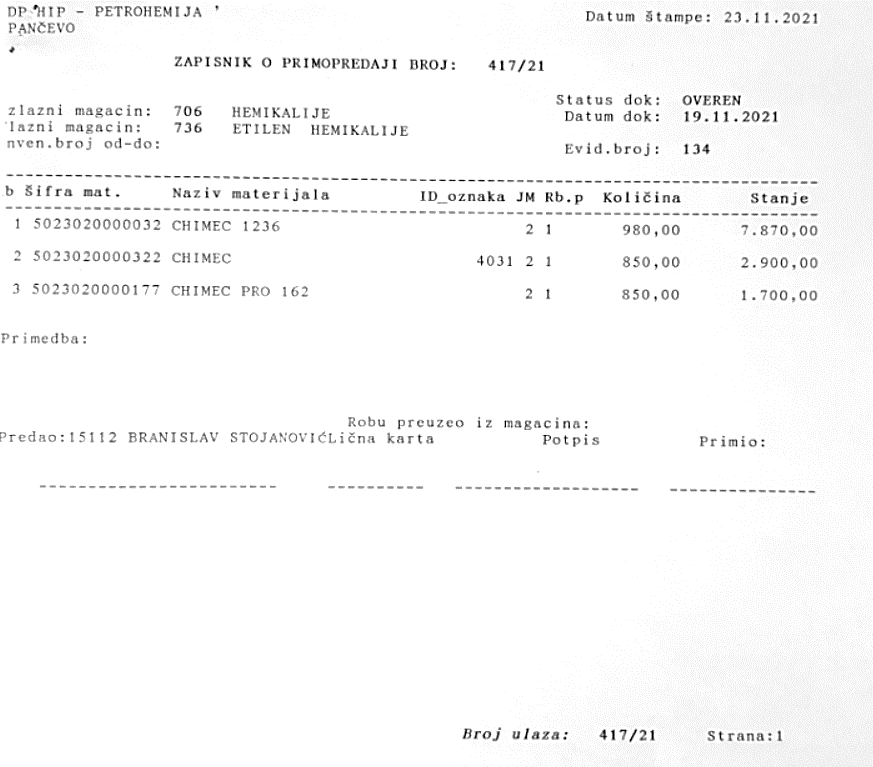
Slika 2: Povratnica dobavljaču



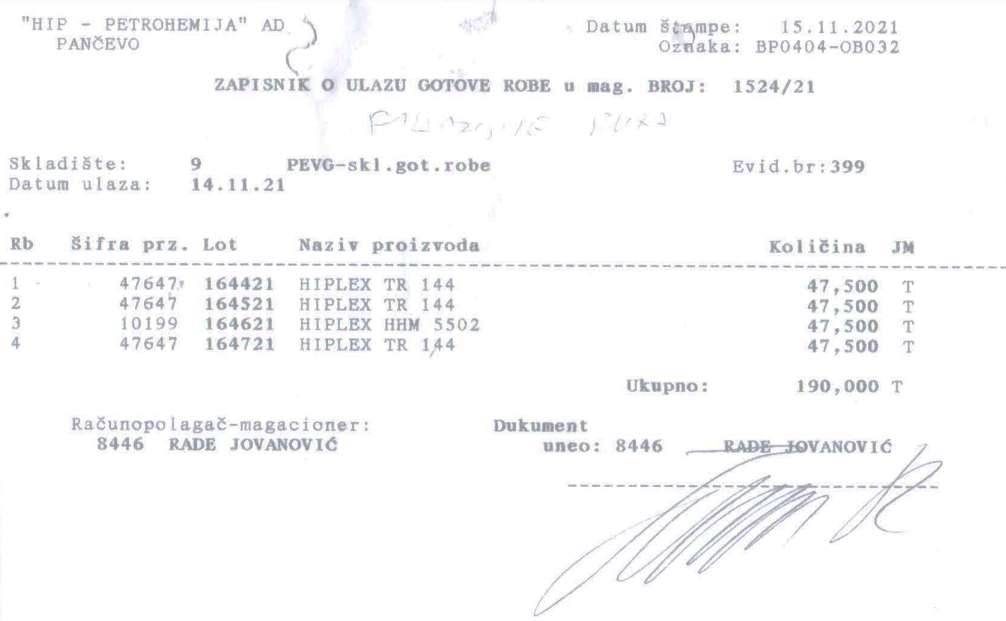
Slika 3: Trebovanje



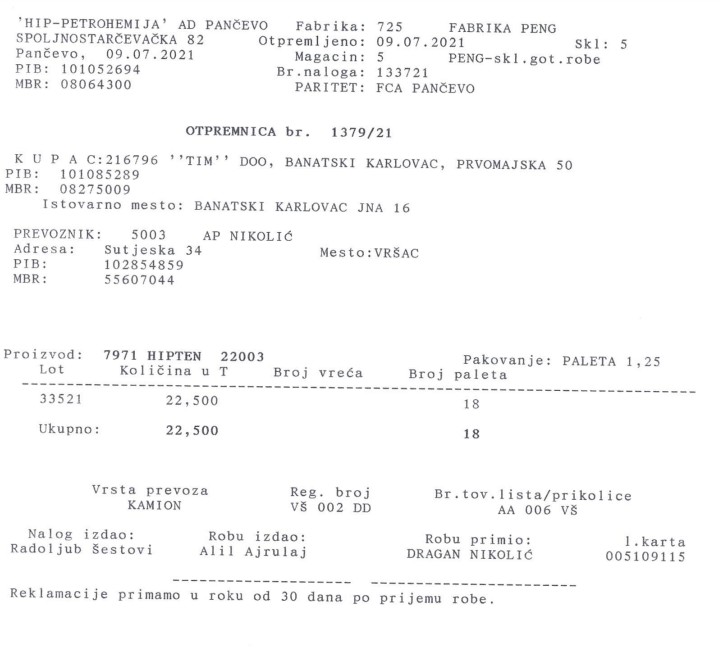
Slika 4: Povratnica materijala



Slika 5: Zapisnik o primopredaji



Slika 6: Zapisnik o ulazu gotove robe u magacin



Slika 7: Otpremnica kupcu

# PROJEKTOVANJE BAZE PODATAKA

## Model podatka



Slika 8: PMOV - Zapisnik o prijemu materijala



Slika 9: PMOV - Povratnica dobavljaču



Slika 10: PMOV - Trebovanje



Slika 11: PMOV - Povratnica materijala



Slika 12: PMOV - Zapisnik o primopredaji



Slika 13: PMOV - Zapisnik o ulazu gotove robe u magacin



Slika 14: PMOV - Otpremnica kupcu

## Relacioni model

Mesto (sifMesta, naziv, postanskiBroj)

PoslovniPartner (sifPartnera, nazivPartnera, maticniBroj, PIB, adresa, tekuciRacun, *sifMesta*)

Radnik (sifRadnika, ime, prezime, JMBG, telefon, adresa)

Porudzbenica (sifPorudzbenice, datumDokumenta, rokPlacanja, napomena, *sifUgovora, sifPonude, sifRadnikOdobrio, sifRadnikKontakt*)

NamenaMagacina (sifNamene, nazivNamene)

NacinIsporuke (sifNacinIsp, nazivNacinIsp, opisNacinIsp)

JedinicaMere (sifJM, nazivJM, oznakaJM)

Proizvod (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

Magacin (sifMagacina, nazivMagacina, *sifRadnika, sifNamene*)

IstorijaStanjaZaliha (sifProizvoda, datum, kolicinaNaStanju, *sifMagacina*)

ZapisnikOPrijemuMaterijala (sifZOPM, brojDostavnice, evidencioniBroj, datumDokumenta, *sifPorudzbenice, sifPartnera, sifRadnika, sifMagacina, sifNacinIsp*)

StavkaZapisnikaOPrijemuMaterijala (sifZOPM, redniBroj, kolicina, *sifProizvoda*)

PovratnicaDobavljacu (sifPovratniceDob, datumDokumenta, *sifZOPM, sifRadnikaOdobrio, sifRadnikaPredao, sifMagacina*)

StavkaPovratniceDobavljacu (sifPovratniceDob, redniBroj, kolicina, *sifProizvoda*)

OrganizacionaJedinica (sifOrgJed, nazivOrgJed)

Trebovanje (sifTrebovanja, evidencioniBroj, datumDokumenta, *sifOrgJedNosilacTroska, sifOrgJedMestoTroska, sifRadnikaIzradio, sifRadnikaOdobrio, sifMagacina*)

StavkaTrebovanja (sifTrebovanja, redniBroj, kolicina, *sifProizvoda*)

PovratnicaMaterijala (sifPovratniceMat, evidencioniBroj, datumDokumenta, *sifOrgJed, sifRadnikaOdobrio, sifRadnikaPrimio, sifMagacina*)

StavkaPovratniceMaterijala (sifPovratniceMat, redniBroj, kolicina, *sifProizvoda*)

ZapisnikOPrimopredaji (sifZOPP, evidencioniBroj, datumRazmene, primedba, *sifOrgJed, sifRadnikaPredao, sifRadnikaPrimio, sifMagacinaIz, sifMagacinaU*)

StavkaZapisnikaOPrimopredaji (sifZOPP, redniBroj, kolicina, *sifProizvoda*)

ZapisnikOUlazuGotoveRobe (sifZGR, evidencioniBroj, datumDokumenta, oznaka, *sifOrgJed, sifRadnika, sifMagacina*)

StavkaZapisnikaOUlazuGotoveRobe (sifZGR, redniBroj, brojLota, kolicina, *sifProizvoda*)

NalogZaOtpremu (sifNalogaZaOtp, datumDokumenta, *sifPrijVozila, sifPrevoza, sifRadnikaIzdao, sifMagacina*)

Paritet (sifPariteta, oznaka, opis)

Pakovanje (sifPakovanja, naziv)

OtpremnicaKupcu (sifOtpremnice, istovarnoMesto, datumDokumenta, napomena*, sifPartneraPrevozi, sifPartneraKupuje, sifNalogaZaOtp, sifPariteta, sifRadnika, sifMagacina*)

StavkaOtpremniceKupcu (sifOtpremnice, redniBroj, kolicina, brKomPakovanja, brojLota, *sifPakovanja, sifProizvoda*)

# DENORMALIZACIJA RELACIJA

## Denormalizacija relacija uz narušavanje 2NF

**Pre denormalizacije:**



Slika 15: Normalizovani konceptualni model

**Proizvod** (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

**IstorijaStanjaZaliha** (sifProizvoda, datum, kolicinaNaStanju, *sifMagacina*)

**Funkcionalne zavisnosti:**

sifProizvoda -> nazivProizvoda, dodatniNaziv, oznaka, *sifJM*

sifProizvoda, datum -> kolicinaNaStanju, *sifMagacina*

**Posle denormalizacije:**



Slika 16: Denormalizacija uz narušavanje 2NF (Pre-joining)

**Proizvod** (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

**IstorijaStanjaZaliha** (sifProizvoda, datum, nazivProizvoda, kolicinaNaStanju, *sifMagacina*)

**Funkcionalne zavisnosti:**

sifProizvoda, datum -> nazivProizvoda, kolicinaNaStanju, *sifMagacina*

sifProizvoda -> nazivProizvoda, dodatniNaziv, oznaka, *sifJM*

**-> Narušena druga normalna forma!**

**Tabela specifikacije trigera:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba uraditi |
| Proizvod | Insert |  | Ne |  |
| Update | nazivProizvoda | Da | Prilikom izmene vrednosti kolone nazivProizvoda u tabeli Proizvod, pokreće se triger koji tu izmenjenu vrednost ažurira u koloni nazivProizvoda u tabeli IstorijaStanjaZaliha u odgovarajućim redovima. |
| Delete |  | Ne |  |
| Istorija stanja zaliha | Insert |  | Da | Triger ažurira vrednost kolone nazivProizvoda na osnovu unete vrednosti kolone sifProizvoda. |
| Update | sifProizvoda | Da | Zabraniti direktno ažuriranje ove kolone. |
| nazivProizvoda | Da | Zabraniti direktno ažuriranje ove kolone. |
| Delete |  | Ne |  |

## Denormalizacija relacija uz narušavanje 3NF

**Pre denormalizacije:**



Slika 17: Normalizovani konceptualni model

**StavkaZapisnikaOPrijemuMaterijala** (sifZOPM, redniBroj, kolicina, *sifProizvoda*)

**Proizvod** (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

**JedinicaMere** (sifJM, nazivJM, oznakaJM)

**Funkcionalne zavisnosti:**

sifZOPM, redniBroj -> kolicina, *sifProizvoda*

sifProizvoda -> nazivProizvoda, dodatniNaziv, oznaka, *sifJM*

sifJM -> nazivJM, oznakaJM

**Posle denormalizacije:**



Slika 18: Denormalizacija uz narušavanje 3NF (Short – circuit keys)

**StavkaZapisnikaOPrijemuMaterijala** (sifZOPM, redniBroj, kolicina, *sifProizvoda, sifJM*)

**Proizvod** (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

**JedinicaMere** (sifJM, nazivJM, oznakaJM)

**Funkcionalne zavisnosti:**

sifZOPM, redniBroj -> kolicina, *sifProizvoda*, *sifJM*

sifProizvoda -> nazivProizvoda, dodatniNaziv, oznaka, *sifJM*

**-> Narušena treća normalna forma!**

**Tabela specifikacije trigera:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba uraditi |
| Proizvod | Insert |  | Ne |  |
| Update | sifJM | Da | Ukoliko postoji StavkaZapisnikaOPrijemuMaterijala za dati Proizvod, ažurira sifJM u tabeli StavkaZapisnikaOPrijemuMaterijala. |
| Delete |  | Ne |  |
| Stavka zapisnika o prijemu materijala | Insert |  | Da | U kolonu sifJM upisuje odgovarajuću vrednost za unetu vrednost sifProizvoda. |
| Update | sifProizvoda | Da | U kolonu sifJM upisuje odgovarajuću vrednost na osnovu nove vrednosti za sifProizvoda. |
| sifJM | Da | Sprečava direktnu izmenu. |
| Delete |  | Ne |  |

# KORISNIČKI DEFINISANI TIPOVI I TRIGERI

## Definicija korisničkom tipa i njegovo korišćenje

**Distinkt tip**

Za kolonu JMBG iz tabele: Radnik (sifRadnika, ime, prezime, JMBG, telefon, adresa)

CREATE OR REPLACE TYPE obj\_jmbg AS OBJECT

(vrednost NUMBER(13),

MEMBER FUNCTION get\_vrednost RETURN NUMBER,

MEMBER FUNCTION get\_pol RETURN VARCHAR2

)

INSTANTIABLE FINAL;

CREATE OR REPLACE TYPE BODY obj\_jmbg AS

MEMBER FUNCTION get\_vrednost RETURN number IS

BEGIN

RETURN SELF.vrednost;

END;

MEMBER FUNCTION get\_pol RETURN varchar2 IS

BEGIN

IF SUBSTR(TO\_CHAR(SELF.vrednost), 10, 3 ) >= 500

THEN RETURN 'Žensko';

ELSE

RETURN 'Muško';

END IF;

END;

END;

CREATE TABLE radnik

(

sifRadnika number NOT NULL,

ime varchar2(15),

prezime varchar2(15),

JMBG obj\_jmbg,

telefon number(15),

adresa varchar2(20),

CONSTRAINT sifra\_pk PRIMARY KEY(sifRadnika)

);

INSERT INTO radnik VALUES (1, 'Ljubica', 'Ilic', obj\_jmbg(2704999865075), 0665105288, 'Proleterska 14' );

INSERT INTO radnik VALUES (2, 'Petar', 'Petrovic', obj\_jmbg(2806996844039), 0637541297, 'Prvi maj 3' );

SELECT r.ime "Ime",

r.jmbg.get\_vrednost() "JMBG",

r.jmbg.get\_pol() "Pol"

from radnik r;

**Struktuirani tip**

Za kolonu naziv iz tabele: Pakovanje (sifPakovanja, naziv)

CREATE OR REPLACE TYPE obj\_naziv\_pak AS OBJECT(

tip VARCHAR2(10),

tezina NUMBER(5,3),

MEMBER FUNCTION get\_tip RETURN VARCHAR2,

MEMBER FUNCTION get\_tezina\_u\_t RETURN NUMBER,

MEMBER FUNCTION get\_tezina\_u\_kg RETURN NUMBER)

INSTANTIABLE NOT FINAL;

CREATE OR REPLACE TYPE BODY obj\_naziv\_pak AS

MEMBER FUNCTION get\_tip RETURN varchar2 IS

BEGIN

RETURN SELF.tip;

END;

MEMBER FUNCTION get\_tezina\_u\_t RETURN number IS

BEGIN

RETURN SELF.tezina;

END;

MEMBER FUNCTION get\_tezina\_u\_kg RETURN number IS

BEGIN

RETURN SELF.tezina \* 1000;

END;

END;

CREATE TABLE pakovanje (

sifPakovanja number NOT NULL,

naziv obj\_naziv\_pak,

CONSTRAINT pakovanje\_pk PRIMARY KEY (sifPakovanja)

);

INSERT INTO pakovanje VALUES (1, obj\_naziv\_pak('Paleta', 1.25));

INSERT INTO pakovanje VALUES (2, obj\_naziv\_pak('Vreca', 0.09));

INSERT INTO pakovanje VALUES (3, obj\_naziv\_pak('Vreca', 1.90));

INSERT INTO pakovanje VALUES (4, obj\_naziv\_pak('Vreca', 0.05));

UPDATE PAKOVANJE p

SET p.naziv.tip = 'Paleta'

WHERE p.naziv.get\_tezina\_u\_kg() > 1000;

SELECT p.naziv.get\_tip() "Tip",

p.naziv.get\_tezina\_u\_t() "Tezina t",

p.naziv.get\_tezina\_u\_kg() "Tezina kg"

FROM pakovanje p;

## Realizacija proceduralne logike uz pomoć trigera

**Trigeri 2NF:**

Proizvod (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

IstorijaStanjaZaliha (sifProizvoda, datum, nazivProizvoda, kolicinaNaStanju, *sifMagacina*)

1. Prilikom izmene vrednosti kolone nazivProizvoda u tabeli Proizvod, pokreće se triger koji tu izmenjenu vrednost ažurira u koloni nazivProizvoda u tabeli IstorijaStanjaZaliha u odgovarajućim redovima

CREATE OR REPLACE TRIGGER "TRG\_UPDATE\_PROIZVOD"

AFTER UPDATE OF nazivProizvoda

ON Proizvod

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER TRG\_UPDATE\_NAZIVPROIZVODA DISABLE';

UPDATE IstorijaStanjaZaliha

SET nazivProizvoda=:NEW.nazivProizvoda

WHERE sifProizvoda =:NEW.sifProizvoda;

COMMIT;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER TRG\_UPDATE\_NAZIVPROIZVODA ENABLE';

END;

END;

2. Prilikom unosa IstorijeStanjaZaliha, triger ažurira vrednost kolone nazivProizvoda na osnovu unete vrednosti kolone sifProizvoda.

CREATE OR REPLACE TRIGGER "TRG\_INSERT\_ISTORIJA"

BEFORE INSERT

ON IstorijaStanjaZaliha

FOR EACH ROW

DECLARE

v\_naziv VARCHAR2(20);

BEGIN

SELECT nazivProizvoda INTO v\_naziv

FROM Proizvod

WHERE sifProizvoda =:NEW.sifProizvoda;

:NEW.nazivProizvoda := v\_naziv;

END;

3. Zabraniti direktno ažuriranje kolone sifProizvoda u okviru IstorijaStanjaZaliha.

CREATE OR REPLACE TRIGGER "TRG\_UPDATE\_SIFPROIZVODA"

AFTER UPDATE OF sifProizvoda

ON IstorijaStanjaZaliha

BEGIN

RAISE\_APPLICATION\_ERROR (

NUM => -20000,

MSG => 'NE SMETE DA MENJATE POLJE SIFPROIZVODA IZ TABELE ISTORIJASTANJAZALIHA, JER JE ONO DEO PRIMARNOG KLJUCA');

END;

4. Zabraniti direktno ažuriranje kolone nazivProizvoda u okviru IstorijaStanjaZaliha.

CREATE OR REPLACE TRIGGER "TRG\_UPDATE\_NAZIVPROIZVODA"

BEFORE UPDATE OF nazivProizvoda

ON IstorijaStanjaZaliha

BEGIN

RAISE\_APPLICATION\_ERROR (

NUM => -20000,

MSG => 'NE SMETE DA MENJATE POLJE NAZIVPROIZVODA IZ TABELE ISTORIJASTANJAZALIHA');

END;

**Trigeri 3NF:**

Proizvod (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, *sifJM*)

StavkaZapisnikaOPrijemuMaterijala (sifZOPM, redniBroj, kolicina, *sifProizvoda, sifJM*)

1. Ukoliko postoji StavkaZapisnikaOPrijemuMaterijala za dati Proizvod, prilikom ažuriranje sifJM u okviru Proizvoda, ažurirati sifJM u tabeli StavkaZapisnikaOPrijemuMaterijala.

create or replace TRIGGER "TRG\_UPDATE\_PROIZVOD\_JM"

AFTER UPDATE OF sifJM

ON Proizvod

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER TRG\_UPDATE\_SIFJM\_STAVKAZOPM DISABLE';

UPDATE StavkaZapisnikaOPrijemuMaterijala

SET sifJM = :NEW.sifJM

WHERE sifProizvoda=:NEW.sifProizvoda;

COMMIT;

BEGIN

EXECUTE IMMEDIATE 'ALTER TRIGGER TRG\_UPDATE\_SIFJM\_STAVKAZOPM ENABLE';

END;

END;

2. Prilikom unosa StavkeZapisnikaOPrijemuMaterija, u kolonu sifJM upisuje se odgovarajuća vrednost za unetu vrednost sifProizvoda.

CREATE OR REPLACE TRIGGER "TRG\_INSERT\_STAVKAZOPM"

BEFORE INSERT

ON StavkaZapisnikaOPrijemuMaterijala

FOR EACH ROW

DECLARE

v\_sifra NUMBER;

BEGIN

SELECT sifJM INTO v\_sifra

FROM Proizvod

WHERE sifProizvoda=:NEW.sifProizvoda;

:NEW.sifJM := v\_sifra;

END;

3. Triger prilikom izmene vrednosti sifProizvoda u okviru StavkeZapisnikaOPrijemuMaterijala, u kolonu sifJM upisuje odgovarajuću vrednost na osnovu nove vrednosti za sifProizvoda.

CREATE OR REPLACE TRIGGER "TRG\_UPDATE\_SIFPROIZVODA\_STAVKAZOPM"

BEFORE UPDATE OF sifProizvoda

ON StavkaZapisnikaOPrijemuMaterijala

FOR EACH ROW

DECLARE

v\_sifra NUMBER;

BEGIN

SELECT sifJM INTO v\_sifra

FROM Proizvod

WHERE sifProizvoda=:NEW.sifProizvoda;

:NEW.sifJM:=v\_sifra;

END;

4. Zabraniti direktno ažuriranje kolone sifJM u okviru StavkeZapisnikaOPrijemuMaterijala.

CREATE OR REPLACE TRIGGER "TRG\_UPDATE\_SIFJM\_STAVKAZOPM"

AFTER UPDATE OF sifJM

ON StavkaZapisnikaOPrijemuMaterijala

BEGIN

RAISE\_APPLICATION\_ERROR (

NUM => -20000,

MSG => 'NE SMETE DA MENJATE POLJE SIFJM IZ TABELE STAVKAZAPISNIKAOPRIJEMUMATERIJALA');

END;

# OPTIMIZACIJA BAZE PODATAKA

## Definisanje i analiza indeksa

**Skeniranje indeksa po opsegu (Range)**

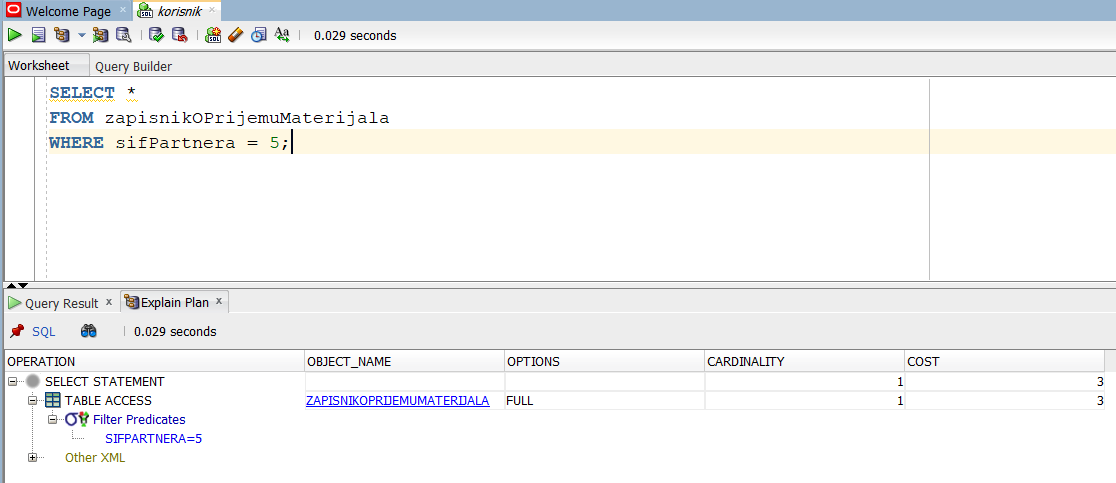
Kako se često pretražuju zapisniciOPrijemuMaterijala za nekog konkretnog poslovnog partnera , kandidat za indeks je kolona sifPartnera koja predstavlja spoljni ključ u tabeli zapisnikOPrijemuMaterijala. Kreiranjem indeksa se ubrzava proces spajanja ove dve tabele.

Izvršavanje sledećeg upita pre kreiranja indeksa traje 0.029s. Ukupni troškovi su 6.

SELECT \*

FROM zapisnikOPrijemuMaterijala

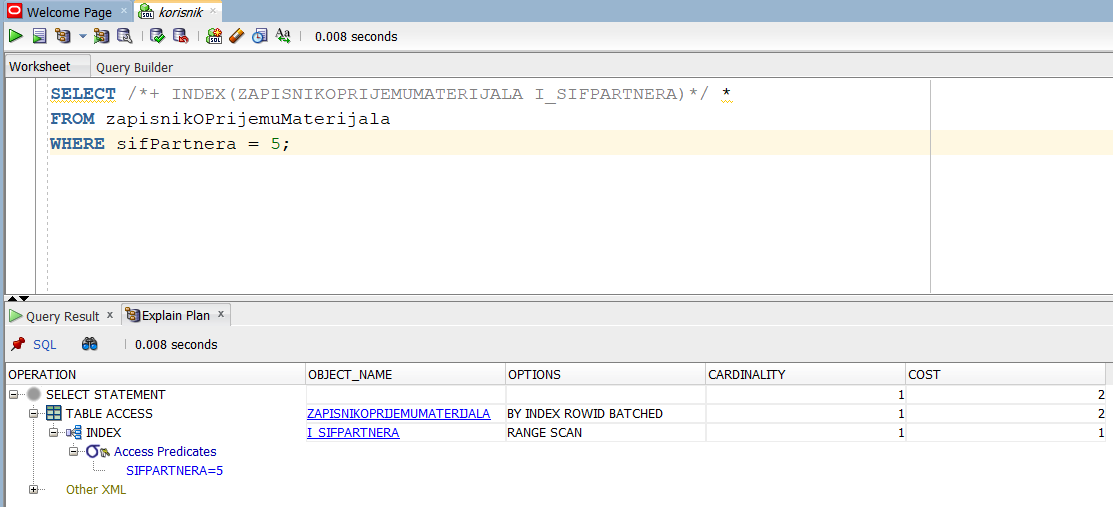
WHERE sifPartnera = 5;



Kreiramo indeks I\_SIFPARTNERA:

create index I\_SIFPARTNERA on ZapisnikOPrijemuMaterijala (sifPartnera);

Nakon kreiranja indeksa vreme izvršavanja je spalo na 0.008s. Ukupni troškovi su spali na 5.



**Skeniranje indeksa zasnovano na funkcijama** **(Function-based)**

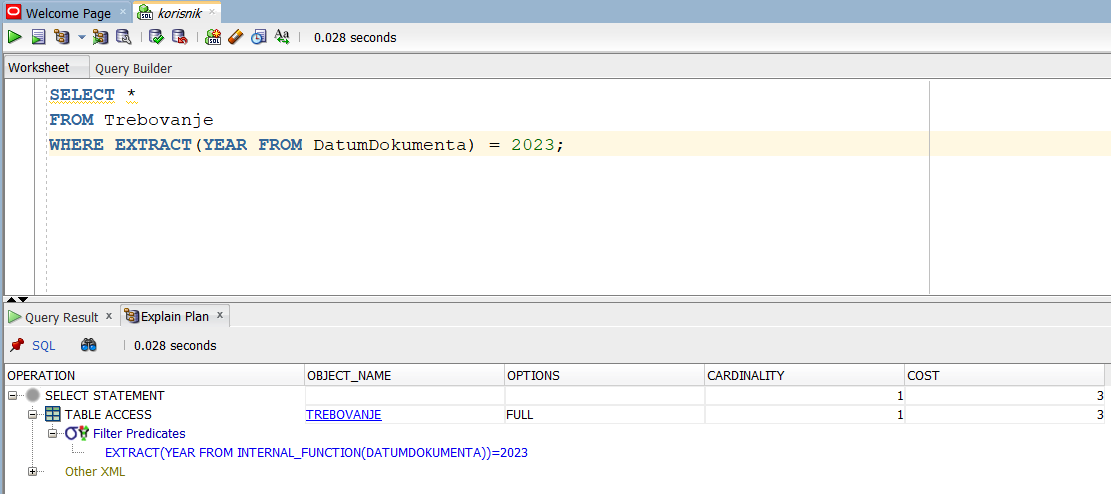
Zbog potrebe prelistavanja trebovanja po godinama, kreiraćemo index zasnovan na funkciji extract(), kojom izvlačimo godine iz datumaDokumenta. Funkcija extract() unapred izračunava vrednosti godina i samim tim ubrzava proces pretraživanja po ovom kriterijumu.

Izvršavanje sledećeg upita pre kreiranja indeksa traje 0.028s. Ukupni troškovi su 6.

SELECT \*

FROM Trebovanje

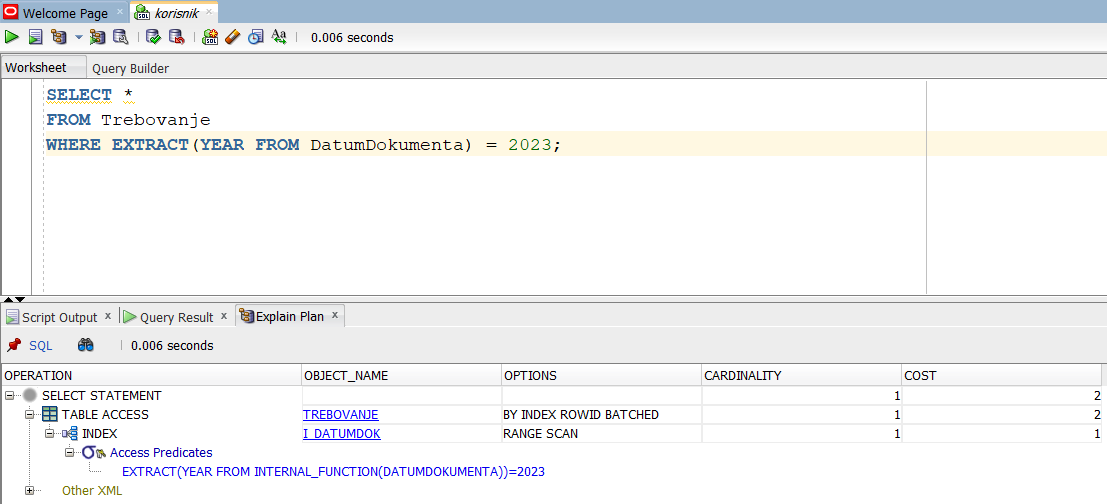
WHERE EXTRACT(YEAR FROM DatumDokumenta) = 2023;

****

Kreiramo indeks I\_ DATUMDOK:

create index I\_DATUMDOK on Trebovanje (EXTRACT(YEAR from datumDokumenta));

Nakon kreiranja indeksa vreme izvršavanja je spalo na 0.006s. Ukupni troškovi su spali na 5.



## Horizontalno particionisanje

ZapisnikOUlazuGotoveRobe (sifZGR, evidencioniBroj, datumDokumenta, oznaka, *sifOrgJed, sifRadnika, sifMagacina*)

Želimo da Zapisnike o ulazu gotove robe skladištimo po particijama, tako što ćemo one zapisnike koji su stariji od 01.01.2022. godine da smestimo u particiju P1\_STARIJA\_DOK, zapisnike koji su kreirani 2022. godine ćemo da smestimo u particiju P2\_PRETHODNAGOD\_DOK, i zapisnike iz tekuće, 2023. godine, zajedno sa buduće kreiranim dokumentima u particiju P3\_TEKUCAGOD\_DOK.

CREATE TABLE ZapisnikOUlazuGotoveRobe

(

sifZGR number NOT NULL,

evidencioniBroj number,

datumDokumenta date,

oznaka varchar2(20),

sifOrgJed number NOT NULL,

sifRadnika number NOT NULL,

sifMagacina number NOT NULL,

CONSTRAINT zogr\_pk PRIMARY KEY(sifZGR),

CONSTRAINT zogr\_fk1 FOREIGN KEY(sifOrgJed) REFERENCES organizacionaJedinica(sifOrgJed),

CONSTRAINT zogr\_fk2 FOREIGN KEY(sifRadnika) REFERENCES radnik(sifRadnika),

CONSTRAINT zogr\_fk3 FOREIGN KEY(sifMagacina) REFERENCES magacin(sifMagacina)

)

PARTITION BY RANGE(datumDokumenta)

(

PARTITION P1\_STARIJA\_DOK VALUES LESS

THAN(TO\_DATE('01.01.2022.', 'DD.MM.YYYY.')),

PARTITION P2\_PRETHODNAGOD\_DOK VALUES LESS

THAN(TO\_DATE('01.01.2023.', 'DD.MM.YYYY.')),

PARTITION P3\_TEKUCAGOD\_DOK VALUES LESS THAN (MAXVALUE)

);

Za proveru da li su particije tabele ZapisnikOUlazuGotoveRobe kreirane koristimo sledeći upit:

SELECT \*

FROM USER\_TAB\_PARTITIONS

WHERE TABLE\_NAME = 'ZAPISNIKOULAZUGOTOVEROBE';

## Vertikalno particionisanje

Pre vertikalnog particionisanja:

PoslovniPartner (sifPartnera, nazivPartnera, maticniBroj, PIB, adresa, tekuciRacun, *sifMesta*)

Nakon vertikalnog particionisanja:

PoslovniPartner (sifPartnera, nazivPartnera)

PoslovniPartnerDetalji (sifPartnera, maticniBroj, PIB, adresa, tekuciRacun, *sifMesta*)

View(pogled) za objedinjavanje podataka iz tabela PoslovniPartner i PoslovniPartnerDetalji:

CREATE OR REPLACE VIEW PARTNER\_VIEW

AS

SELECT p.sifPartnera, p.nazivPartnera,

pd.maticniBroj, pd.PIB,

pd.adresa, pd.tekuciRacun, pd.sifMesta

FROM PoslovniPartner p, PoslovniPartnerDetalji pd

WHERE p.sifPartnera=pd.sifPartnera;

Triger za insert:

CREATE OR REPLACE TRIGGER TRG\_PARTNERVIEW\_INSERT

INSTEAD OF INSERT ON POSLOVNIPARTNER\_VIEW

FOR EACH ROW

BEGIN

INSERT INTO PoslovniPartner (sifPartnera, nazivPartnera)

VALUES (:NEW.sifPartnera, :NEW.nazivPartnera);

INSERT INTO PoslovniPartnerDetalji (sifPartnera, maticniBroj,

PIB, adresa, tekuciRacun, sifMesta) VALUES (:NEW.sifPartnera, :NEW.maticniBroj,

:NEW.PIB, :NEW.adresa, :NEW.tekuciRacun, :NEW.sifMesta);

END;

Triger za update:

CREATE OR REPLACE TRIGGER TRG\_PARTNERVIEW\_UPDATE

INSTEAD OF UPDATE ON POSLOVNIPARTNER\_VIEW

FOR EACH ROW

BEGIN

UPDATE PoslovniPartner

SET sifPartnera = :NEW.sifPartnera, nazivPartnera = :NEW.nazivPartnera

WHERE sifPartnera = :OLD.sifPartnera;

UPDATE PoslovniPartnerDetalji

SET sifPartnera = :NEW.sifPartnera, maticniBroj = :NEW.maticniBroj,

PIB = :NEW.PIB, adresa = :NEW.adresa, tekuciRacun = :NEW.tekuciRacun, sifMesta = :NEW.sifMesta

WHERE sifPartnera = :OLD.sifPartnera;

END;

Triger za delete:

CREATE OR REPLACE TRIGGER TRG\_PARTNERVIEW\_DELETE

INSTEAD OF DELETE ON POSLOVNIPARTNER\_VIEW

FOR EACH ROW

BEGIN

DELETE FROM PoslovniPartner

WHERE sifPartnera = :OLD.sifPartnera;

DELETE FROM PoslovniPartnerDetalji

WHERE sifPartnera = :OLD.sifPartnera;

END;

## Primena drugih optimizacionih tehnika

**Storing Derivable Values tehnika optimizacije**



JedinicaMere (sifJM, nazivJM, oznakaJM)

**Proizvod** (sifProizvoda, nazivProizvoda, dodatniNaziv, oznaka, **dostupnaKolicina**, *sifJM*)

Magacin (sifMagacina, nazivMagacina, *sifRadnika, sifNamene*)

**IstorijaStanjaZaliha** (sifProizvoda, datum, kolicinaNaStanju, *sifMagacina*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba uraditi |
| Proizvod | Insert |  | Ne |  |
| Update | dostupnaKolicina | Da | Zabrana direktnog ažuriranja polja dostupnaKolicina u tabeli Proizvod |
| Delete |  | Ne |  |
| Istorija stanja zaliha | Insert |  | Da | Prilikom unosa nove IstorijeStanjaZaliha, okida se triger koji poziva proceduru za ažuriranje polja dostupnaKolicina u tabeli Proizvod. |
| Update | kolicinaNaStanju | Da | Prilikom ažuriranja polja kolicinaNaStanju u okviru tabele IstorijaStanjaZaliha, okida se triger koji poziva proceduru za ažuriranje polja dostupnaKolicina u tabeli Proizvod. |
| Delete |  | Da | Priliko brisanja IstorijeStanjaZaliha, okida se triger koji poziva proceduru za ažuriranje polja dostupnaKolicina u tabeli Proizvod. |

ALTER TABLE PROIZVOD

ADD dostupnaKolicina NUMBER(10,3);

ALTER TABLE PROIZVOD

MODIFY (dostupnaKolicina DEFAULT 0);

Prvo ćemo preko 2 trigera da pokrijemo slučajeve za insert/update/delete tabele IstorijaStanjaZaliha.

Prvi triger određuje šifru proizvoda i postavlja je u promenljivu paketa isz:

create or replace PACKAGE isz AS

proiz NUMBER:=0;

END;

create or replace TRIGGER "DOSTUPNAKOLICINA1"

BEFORE INSERT OR UPDATE OR DELETE ON IstorijaStanjaZaliha

FOR EACH ROW

BEGIN

IF (INSERTING OR UPDATING)

THEN

BEGIN isz.proiz:=:NEW.sifProizvoda;END;

ELSE

BEGIN isz.proiz:=:OLD.sifProizvoda;END;

END IF;

END;

Drugi triger čita vrednost promenljive paketa, smešta je u novu varijablu i poziva proceduru kojoj prosleđuje vrednost te varijable:

create or replace TRIGGER "DOSTUPNAKOLICINA2"

AFTER INSERT OR UPDATE OR DELETE ON IstorijaStanjaZaliha

DECLARE

sifPro NUMBER:=isz.proiz;

BEGIN

DOSTUPNAKOLICINAPROIZVODA(sifPro);

isz.proiz:=0;

END;

Procedura koja se poziva:

create or replace PROCEDURE "DOSTUPNAKOLICINAPROIZVODA"

(sifPro IN NUMBER) AS

kol istorijastanjazaliha.kolicinanastanju%type:=0;

suma istorijastanjazaliha.kolicinanastanju%type:=0;

mag number:= 0;

BEGIN

FOR mag IN ( SELECT distinct sifMagacina from istorijastanjazaliha where sifProizvoda = sifPro)

LOOP

SELECT kolicinaNaStanju into kol

from istorijaStanjaZaliha

where sifProizvoda = sifPro and datum =

(select max(datum) from istorijaStanjaZaliha where sifProizvoda = sifPro AND sifMagacina = mag.sifmagacina group by sifProizvoda, sifMagacina);

suma:= suma + kol;

END LOOP;

UPDATE Proizvod

SET dostupnaKolicina = suma

WHERE sifProizvoda = sifPro;

END;

Triger za zabranu direktnog ažuriranja polja dostupnaKolicina u tabeli Proizvod:

create or replace TRIGGER "TRG\_UPDATE\_DOSTUPNAKOLICINA"

AFTER UPDATE OF dostupnaKolicina

ON Proizvod

BEGIN

IF isz.proiz = 0 THEN

RAISE\_APPLICATION\_ERROR (

NUM => -20000,

MSG => 'NE SMETE DA MENJATE POLJE DOSTUPNAKOLICINA IZ TABELE PROIZVOD, JER SE ONO SAMO OBRACUNAVA');

END IF;

END;

**Hard – Coded Values tehnika optimizacije**

Početni konceptualni model



NamenaMagacina (sifNamene, nazivNamene)

Magacin (sifMagacina, nazivMagacina, *sifRadnika, sifNamene*)

Konceptualni model posle optimizacije

Kako tabela Namena magacina ima konstantan broj redova, nju možemo izbaciti, a njene vrednosti (Ulazni magacin, Izlazni magacin) ubaciti u okviru CHECK ograničenja za atribut namena u okviru tabele Magacin.



Magacin (sifMagacina, nazivMagacina, **namena**, *sifRadnika*)

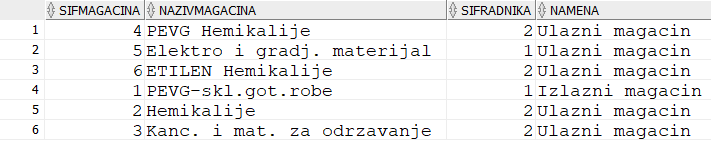
ALTER TABLE MAGACIN

DROP COLUMN sifNamene;

ALTER TABLE MAGACIN

ADD namena varchar2(40)

ADD CONSTRAINT check\_namena CHECK(NAMENA IN ('Ulazni magacin', 'Izlazni magacin'));



**Repeating Single Detail with Master tehnike optimizacije**

Početni konceptualni model



Magacin (sifMagacina, nazivMagacina, *sifRadnika, sifNamene*)

Trebovanje (sifTrebovanja, evidencioniBroj, datumDokumenta, *sifOrgJedNosilacTroska, sifOrgJedMestoTroska, sifRadnikaIzradio, sifRadnikaOdobrio, sifMagacina*)

Konceptualni model posle optimizacije

Redudantni atribut poslDatumTreb dodajemo u tabelu Magacin, da bismo mogli da pratimo kada se iz svakog magacina vršilo poslednje trebovanje.



Magacin (sifMagacina, nazivMagacina, **poslDatumTreb**, *sifRadnika, sifNamene*)

Trebovanje (sifTrebovanja, evidencioniBroj, datumDokumenta, *sifOrgJedNosilacTroska, sifOrgJedMestoTroska, sifRadnikaIzradio, sifRadnikaOdobrio, sifMagacina*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela | Tip trigera | Kolona | Potreban | Šta treba uraditi |
| Magacin | Insert |  | Ne |  |
| Update | poslDatumTreb | Da | Zabrana direktnog ažuriranja polja poslDatumTreb u tabeli Magacin. |
| Delete |  | Ne |  |
| Trebovanje | Insert |  | Da | Prilikom unosa novog Trebovanja, okida se triger koji poziva proceduru za ažuriranje polja poslDatumTreb u tabeli Magacin. |
| Update | datumDokumenta | Da | Prilikom ažuriranja polja datumDokumenta u okviru tabele Trebovanje, okida se triger koji poziva proceduru za ažuriranje polja poslDatumTreb u tabeli Magacin. |
| Delete |  | Da | Priliko brisanja Trebovanja, okida se triger koji poziva proceduru za ažuriranje polja poslDatumTreb u tabeli Magacin. |

ALTER TABLE MAGACIN

ADD poslDatumTreb DATE;

ALTER TABLE MAGACIN

MODIFY (poslDatumTreb DEFAULT null);

Prvo ćemo preko 2 trigera da pokrijemo slučajeve za insert/update/delete tabele trebovanje.

Prvi triger određuje šifru magacina i unosi je u promenljivu paketa:

create or replace PACKAGE treb AS

magacin NUMBER:=0;

END;

CREATE OR REPLACE TRIGGER "POSLDATUMTREB1"

BEFORE INSERT OR UPDATE OR DELETE ON Trebovanje

FOR EACH ROW

BEGIN

IF (INSERTING OR UPDATING)

THEN

BEGIN treb.magacin := :NEW.sifMagacina; END;

ELSE

BEGIN treb.magacin := :OLD.sifMagacina; END;

END IF;

END;

Drugi triger isčitava vrednost promenljive paketa, smešta je u novu varijablu i poziva proceduru kojoj prosleđuje tu varijablu:

create or replace TRIGGER "POSLDATUMTREB2"

AFTER INSERT OR UPDATE OR DELETE ON Trebovanje

DECLARE

sifMag NUMBER:=treb.magacin;

BEGIN

POSLEDNJIDATUMTREBOVANJA(sifMag);

treb.magacin := 0;

END;

Procedura koja se poziva:

CREATE OR REPLACE PROCEDURE "POSLEDNJIDATUMTREBOVANJA"(sifMag IN NUMBER)

AS

poslDatum magacin.posldatumtreb%type;

BEGIN

SELECT max(datumDokumenta) INTO poslDatum

FROM Trebovanje

WHERE sifMagacina=sifMag;

UPDATE Magacin

SET poslDatumTreb = poslDatum

WHERE SifMagacina = sifMag;

END;

Triger za zabranu direktnog ažuriranja polja poslDatumTreb u tabeli Magacin:

CREATE OR REPLACE TRIGGER "TRG\_UPDATE\_POSLDATUMTREB"

AFTER UPDATE OF poslDatumTreb

ON Magacin

BEGIN

IF treb.magacin = 0 THEN

RAISE\_APPLICATION\_ERROR (

NUM => -20000,

MSG => 'NE SMETE DA MENJATE POLJE POSLDATUMTREB IZ TABELE MAGACIN, JER SE ONO SAMO OBRACUNAVA');

END IF;

END;

# TEHNOLOGIJE ZA IMPLEMENTACIJU PROJEKTA

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

Sistem za upravljanje bazom podataka je softver pomoću kog se upravlja podacima i obezbeđuje pristup do njih. Za potrebe ovog projekta korišćena je **Oracle baza podataka**, koja predstavalja sistem za upravljanje releacionim bazama podataka (**SURBP**). U vlasništvu je kompanije Oracle Corporation, a nastala je još 1977. godine od strane [Larry Ellison](https://en.wikipedia.org/wiki/Larry_Ellison)-a i drugih inženjera. Kompanija Oracle je jedna od najstarijih kompanija za upravljanje bazama podataka na svetu. Redovno prati promene i prilagođava se potrebama kompanija tako što često objavljuje nove verzije i izdanja. U ovom radu korišćena je verzija **Oracle 21c Express Edition**. To je jedna od najnovijih verzija, a izdanje Express Edition je besplatno izdanje, veoma lako za instalaciju i upravljanje, ali sa ograničenim funkcionalostima. Može se koristiti na Windows i Linux operativnim sistemima. Jedna od največih prednosti korišćenje Oracle baze podataka je podrška proširenju PL/SQL-a za proceduralno programiranje.

Radi lakšeg razvoja i upravljanja Oracle bazom podataka korišćen je **SQL developer**. On predstavlja besplatno, integrisano razvojno okruženje (IDE) za rad sa SQL-om i PL-SQL-om u Oracle bazama podataka. Oracle SQL developer omogućava korisnicima i administratorima da kreiraju i modifikuju tabele baze podataka preko SQL upita koristeći CRUD SQL komande. CRUD je akronim za kreiranje, čitanje, ažuriranje i brisanje i odnosi se na četiri osnovne operacije koje programeri koriste kako bi manipulisali podacima u tabelama baze podataka. Pored tabela, SQL developer podržava i širok spektar drugih objekata (Pogleda, indeksa, paketa, procedura, trigera, tipova itd. ), koji su takođe bili korišćeni u ovom projektu.

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

Kao programsko ogruženje za razvoj korisničnog interfejsa korišćen je **Netbeans IDE**, a kod je napisan u programskom jeziku **Java**. Netbeans IDE je besplatno okruženje otvorenog koda koje omogućava brzo i lako razvijanje desktop, mobilnih i web aplikacija. Uglavnom se koristi za razvoj Java aplikacija, ali podržava i druge programske jezike (PHP, Fortran, Python, Rybi, C, C++, Javascript), kao i gotovo sve operativne sisteme (Windows, Linux, Mac OS, Solaris).

Netbeans su izvorno razvili češki studenti 1996. godine kao deo studentkog projekta (izvorno nazvan Xelfi). Xelfi je bio prvi Java IDE napisan u Javi. Roman Stanek osnovao je kompaniju za proizvodnju komercijalne verzije NetBeans IDE-a 1997. godine. Nedugo zatim, 1999. godine, NetBeans je kupila kompanija Sun Microsystems, i u njenom vlasništvu 2000. godine Netbeans postaje okruženje otvorenog koda (open-source). Sun Microsystems je 2010. godine kupila kompanija Oracle, koja je nastavila da razvija Netbeans narednih nekoliko godina, nakon čega je 2016. godine donirala izvorni kod Netbeans-a kompaniji Apache Software Foundation. U projektu je korišćeno izdanje **Netbeans IDE 8.2**, što je i poslednje izdanje kompanije Oracle. Buduća izdanja, počevši od izdanja 9.0,, dobila su novi naziv: Apache Netbeans.