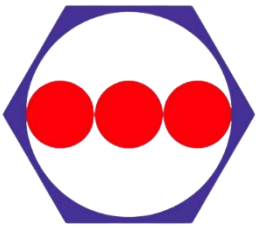


INFORMACIONI SISTEM ZA PODRŠKU UPRAVLJANJU PREDUZEĆEM ZA PROIZVODNJU METALNE GALANTERIJE

-SISTEMI BAZA PODATAKA-

Sanja Tica IT80/2019

Novi Sad, 2022.

SADRŽAJ

UVOD

Projekat koji je predstavljen u ovoj dokumentaciji podrazumjeva razvoj baze podataka za potrebe rada preduzeća koje se bavi proizvodnjom metalne galanterije, a koja treba da omogući automatizaciju procesa obrade podataka o resursima i procesima, koji su neizostavni dio procesa proizvodnje.

Ovakav projekat ima za cilj da omogući svojim korisnicima lakšu interakciju, brži, bolji i efikasniji protok i pristup svim informacijama koje kroz pomenuti sistem cirkulišu, kao i olakšani nadzor i kontrolu samog načina na koji se proizvodnja odvija.

Dalje, u tekstu, biće priložen ER model podataka,odnosno šema koja reprezentuje cijelokupan sistem, zatim ER model dijela sistema, koji reprezentuje proces kupovine već gotovog proizvoda, tabelarni prikaz obilježja i ograničenja koja je potrebno zadovoljiti u sistemu, te relacioni model koji predstavlja implementaciju prethodno pomenutog ER modela podataka.

ANALIZA PROGRAMSKOG DOMENA

Cilj rada jednog ovakvog sistema jeste optimizacija samog procesa proizvodnje, ali i poboljšanje interakcije kako među korisnicima sistema, tako i korisnika i sistema.

Proizvodnja je proces koji obuhvata veliki broj podprocesa, počevši od dobavljanja potrebnih sirovina i materijala, zatim angažovanja adekvatnog tima za izvršavanje proizvodnih operacija, pohađanja potrebnih obuka za rukovanje određenom opremom i sredstvima rada, te nadgledanja skladišta sirovina i gotovih proizvoda. Obzirom da se radi o prilično velikom broju podprocesa u okviru jednog procesa, potrebna je dobra sinhronizacija kako bi se čitav proces odvijao na adekvatan način. Samim tim, u ovakvom jednom sistemu neophodno je voditi evidenciju o svim prethodno pomenutim procesima, što znači da je u pitanju velika količina podataka (recimo količina materijala koja je utrošena u jednom mjesecu, količina prodatih proizvoda za neko vrijeme, spisak mašina i alata koji se koriste u

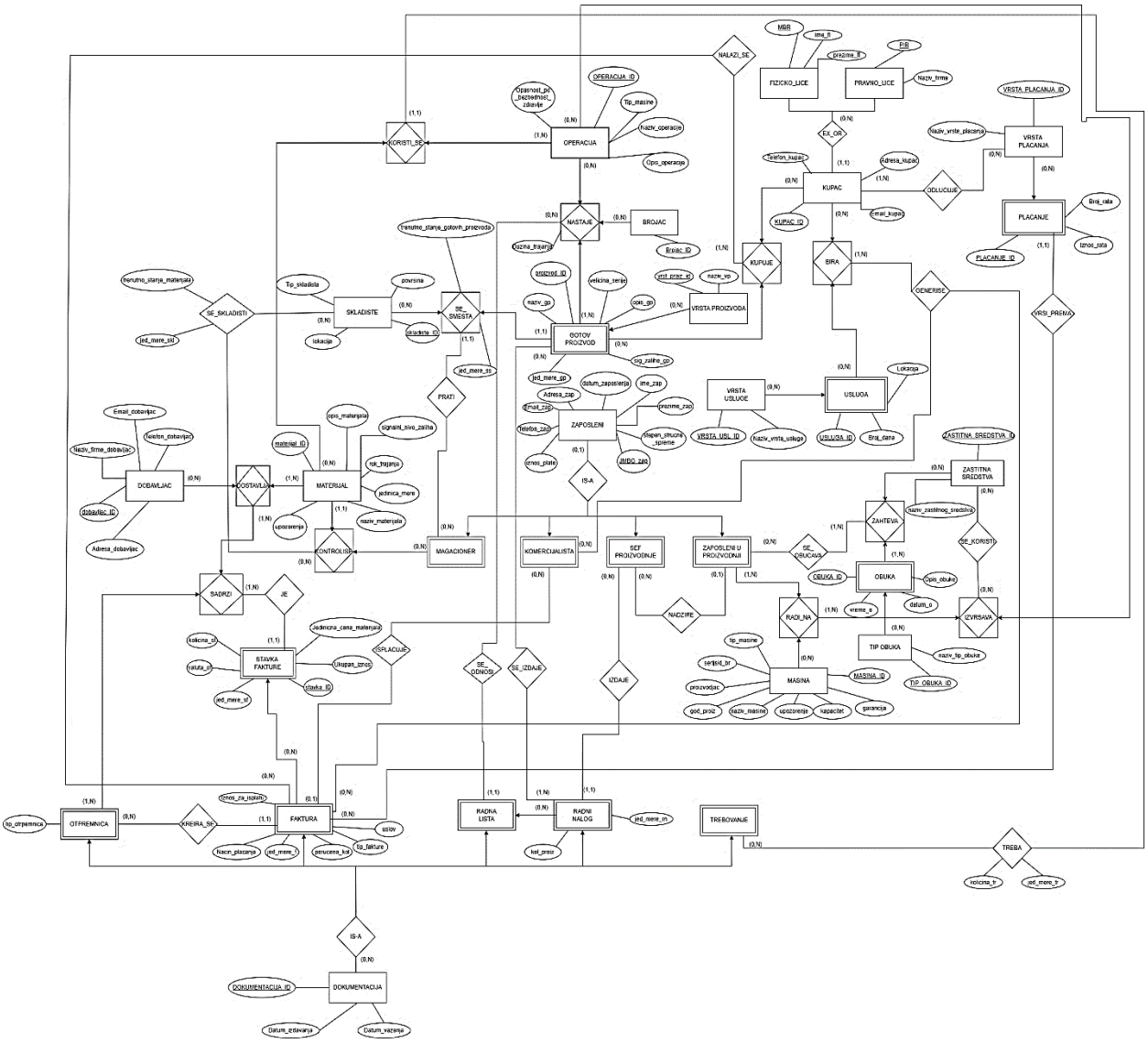
proizvodnom procesu, učesnici u procesima), a koje je potrebno na nekom odgovarajućem mjestu

skladištiti određeni vremenski period. Pri optimizaciji jednog ovakvog procesa u velikoj mjeri može pomoći postojanje baze podataka.

Baza podataka predstavlja skup međusobno povezanih podataka, koji su logički koherentni i neodvojivi od svoga značenja. Takođe, reprezentuje neki aspekat realnog svijeta (Miniworld ili University of Discourse – UID).

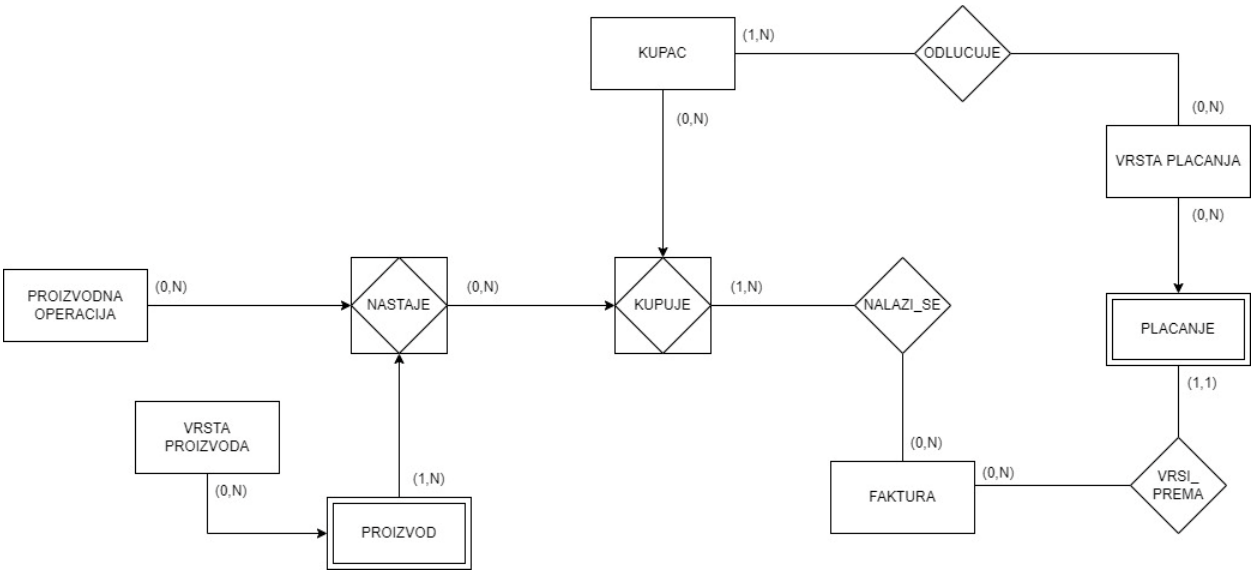
Uz jedan ovakav sistem, olakšan je nadzor i kontrola proizvodnog procesa, vođenje evidencije o svim podacima koji su neophodni za proces, pa samim tim i konačna realizacija. Takođe, baza podataka omogućava da svi podaci, od onog trenutka kada ih u ovaj sistem zabilježimo, budu na dohvat ruke kada god je to potrebno, umjesto da svaki put iznova podatke prikupljamo, organizujemo i skladištimo. Stoga su beneficije upotrebe baze podataka brojne, kako za interne, tako i za eksterne strane, odnosno sve zainteresovane grupe kojima je dati sistem od važnosti.

ER MODEL



Slika 1: ER model čitavog sistema

ER MODEL PODŠEME



Slika 2 : ER model procesa kupovine gotovog proizvoda

Proizvodi, koje je preduzeće proizvelo kroz niz određenih proizvodnih operacija, nakon završne obrade isporučuju se krajnjem korisniku. Proizvodi se mogu klasifikovati u više različitih kategorija, stoga kupac prilikom kupovine bira neki od proizvoda koji se nalaze u ponudi preduzeća. Takođe, moguće je da kupac izvrši kupovinu različitih proizvoda, a koji mogu biti i različite vrste. Da bi preduzeće neko pravno ili fizičko lice smatralo kupcem u sistemu, ne mora da kupi nijedan proizvod. Razlog za to jeste što kupac može, umjesto same kupovine proizvoda, da odabere neke od usluga koje preduzeće nudi, odnosno lice može odabrati ugradnju ili prevoz proizvoda, što se mora platiti, pa tako postaje kupac iako nije kupio neki konkretan proizvod. Te usluge kupac može da zatraži uz sam proizvod, ali kako usluga i proizvod nisu vezani jedno za drugo, fizičko ili pravno lice se jednako smatra kupcem i ukoliko se odluči samo za proizvod ili samo za uslugu.

Prilikom sprovođenja kupovine, preduzeće kupcu za kupljene proizvode izdaje fakturu, kao osnovni dokument na kom su obuhvaćeni kupljeni proizvodi i ukupna cijena tih proizvoda, a na osnovu koje kupac vrši plaćanje željenog proizvoda. Kupac ima i mogućnost izbora načina plaćanja: cjelokupan iznos odjednom

(gotovinom ili uplatom preko računa) ili podjelom na manje iznose, koje mora isplatiti u određenom vremenskom periodu (u ratama).

TABELARNI PRIKAZ OBILJEŽJA I OGRANIČENJA

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	KUPACID	Identifikacioni broj kupca	NUMERIC	8	⊥	d>0
2.	IME	Ime kupca	VARCHAR	15	⊥	Δ
3.	PREZIME	Prezime kupca	VARCHAR	20	⊥	Δ
4.	ADRESA	Adresa kupca	VARCHAR	35	⊤	Δ
5.	TELEFON	Telefon kupca	VARCHAR	15	⊤	Δ
6.	EMAIL	Email kupca	VARCHAR	15	⊥	Δ
7.	DATUM_RODJENJA	Datum rođenja kupca	DATE		⊤	Δ
Ključ	*K={KUPACID}*					

Tabela 1: Tabelarni prikaz obilježja TE KUPAC

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	VRSTAPROIZVODAID	Identifikacioni broj vrste proizvoda	NUMERIC	8	⊥	d>0
2.	NAZIV	Naziv vrste proizvoda	VARCHAR	25	⊥	Δ

Ključ	*K = {VRSTAPROIZVODAID}*

Tabela 2: Tabelarni prikaz obilježja TE VRSTA PROIZVODA

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	**PROIZVODID**	**Identifikacioni broj proizvoda**	**NUMERIC**	**8**	⊥	**d>0**
2.	**NAZIV**	**Naziv proivoda**	**VARCHAR**	**30**	⊥	**Δ**
3.	**OPIS**	**Opis proizvoda (namjena, funkcionalnosti)**	**VARCHAR**	**80**	⊤	**Δ**
4.	**VELICINA_SERIJE**	**Veličina jedne serije datog proizvoda (ako je serijska proizvodnja)**	**NUMERIC**	**6**	⊤	**Δ**
5.	**JEDINICA_MJERE**	**Jedinica mjere proizvoda**	**VARCHAR**	**10**	⊥	**d={kg, m^2, m, l, komad, blok, table}**
6.	**SIGNALNE_ZALIHE**	**Minimalna neophodna količina proizvoda na skladištu**	**NUMERIC**	**5**	⊥	**Δ**

7.	**KATEGORIJA**	**Kategorija proizvoda: N – Namještaj AO – Auto oprema MA – Mašine i alati GR – Građevina PZO – Podne i zidne obloge**	**VARCHAR**	**5**	⊥	**Δ**
8.	**CIJENA**	**Cijena proizvoda**	**NUMERIC**	**8**	⊥	**Δ**
Ključ	**K = {VRSTAPROIZVODAID + PROIZVODID}**					

Tabela 3: Tabelarni prikaz obilježja TE PROIZVOD

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	**FAKTURAID**	**Identifikacioni broj fakture**	**NUMERIC**	**8**	⊥	**d>0**
2.	**NACIN_PLACANJA**	**Isplata iznosa obračunatog na fakturi: 1 – Gotovinski 2 – U ratama 3 – Preko računa**	**CHAR**	**1**	⊥	**Δ**
3.	**JEDINICA_MJERE**	**Jedinica mjere fakture**	**VARCHAR**	**10**	⊥	**d={kg, m^2, m, l, komad, blok, table}**
4.	**VALUTA**	**Valuta u kojoj se izražava**	**VARCHAR**	**3**	⊥	**d={rsd,eur,usd,chf}**

		iznos za isplatu				
5.	**PORUCENA_KOLICINA**	**Poručena količina fakturisanog proizvoda**	**NUMERIC**	**10**	⊥	**Δ**
6.	**TIP_FAKTURE**	**Tip fakture: 0 – ulazna 1 – izlazna**	**NUMERIC**	**1**	⊥	**d={0,1}**
7.	**USLOV**	**Uslov fakturisanja**	**VARCHAR**	**20**	⊤	**Δ**
8.	**DATUM_IZDAVANJA**	**Datum izdavanja fakture**	**DATE**		⊥	**Δ**
9.	**PDV**	**Iznos PDV-a obračunat na osnovicu fakture**	**NUMERIC**	**3**	⊤	**Δ**
Ključ	***K = {FAKTURAID}***					

Tabela 4: Tabelarni prikaz obilježja TE FAKTURA

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	**OPERACIJAID**	**Identifikacioni broj operacije**	**NUMERIC**	**8**	⊥	**d>0**
2.	**NAZIV**	**Naziv proizvodne operacije**	**VARCHAR**	**30**	⊥	**Δ**

3.	**OPIS**	**Opis operacije**	**VARCHAR**	**100**	⊤	**Δ**
4.	**TIP_MASINE**	**Tip mašine koja se koristi za realizaciju operacije (ako se koristi)**	**VARCHAR**	**35**	⊤	**Δ**
5.	**RIZIK**	**Nivo rizika koji sa sobom nosi operacija (opasnost po bezbjednost i zdravlje)**	**VARCHAR**	**50**	⊤	**Δ**
6.	**VRIJEME_TRAJANJA**	**Vrijeme trajanja operacije izražena u satima ili minutama**	**VARCHAR**	**5**	⊥	**d={h,min}**
7.	**DUZINA_TRAJNJA**	**Duzina trajanja operacije**	**NUMERIC**	**7**	⊥	**Δ**

Ključ	***K={OPERACIJAID}***

Tabela 5: Tabelarni prikaz obilježja TE PROIZVODNA OPERACIJA

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	**VRSTAPLACANJAID**		**NUMERIC**	**8**	⊥	**d>0**
2.	**NAZIV**	**Naziv vrste plaćanja: 0 – Gotovina 1 – U ratama**	**CHAR**	**1**	⊥	**Δ**
Ključ	***K={VRSTAPLACANJAID}***					

Tabela 6: Tabelarni prikay obilje\ja TE VRSTA PLACANJA

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
1.	**PLACANJEID**	**Identifikacioni broj plaćanja**	**NUMERIC**	**8**	⊥	**d>0**
2.	**BROJ_RATA**	**Broj rata u kojima se isplaćuje zaduženje (ako je plaćanje u ratama)**	**NUMERIC**	**3**	⊤	**Δ**
3.	**IZNOS_RATE**	**Iznos jedne rate**	**NUMERIC**	**5**	⊤	**Δ**

Ključ	***K={PLACANJEID+VRSTAPLACANJAID}***

Tabela 7: Tabelarni prikaz obilježja TE PLACANJE

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
Ključ	***K={KUPACID+VRSTAPLACANJAID}***					

Tabela 8: Tabelarni prikaz obilježja TP ODLUCUJE

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
Ključ	***K={VRSTAPROIZVODAID+PROIZVODID+OPERACIJAID}***					

Tabela 9: Tabelarni prikaz obilježja TP NASTAJE

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
Ključ	***K={VRSTAPROIZVODAID+PROIZVODID+OPERACIJAID+KUPACID}***					

Tabela 10: Tabelarni prikaz obilježja TP KUPUJE

Redni broj	Atribut	Opis	Tip podatka	Dužina podatka	Null	Uslov
Ključ	***K={VRSTAPROIZVODAID+PROIZVODID+OPERACIJAID+KUPACID +FAKTURAID}***					

Tabela 11: Tabelarni prikaz obilježja TP NALAZI SE

RELACIONI MODEL

Danas se, najčešće, u komercijalnoj upotrebi mogu naći relacioni ili objektno – relacioni modeli podataka.

Relacioni model nastao je 70-ih godina prošlog vijeka, u cilju prevazilaženja nedostataka koje su imali modeli koji su se prethodno najviše koristili (mrežni, hijerarhijski...). Relacioni model se zasniva na teoriji skupova i predikatskom računu, i dominantno se koristi za modelovanje realnih problema i dinamično upravljanje podacima.Osnovne karakteristike:

- Minimalna redundantnost podataka
- Jednostavnost ažuriranja
- Izbjegavanje anomalija ažuriranja
- Redoslijed kolona i tabela ne utiče na sadržaj podataka
- Ne potoje dvije identične torke u jednoj tabeli
- Svaka torka se može jednoznačno identifikovati pomoću primarnog ključa

Model podatakapredstavlja matematičku apstrakciju koja se koristi za izgradnju šeme baze podataka, koja reprezentuje model baze podataka informacionog sistema, te model posmatranog dijela realnog sistema. Trojka:

(S, I, O)

S – ***Strukturalna komponenta***

- Služi za modelovanje logičke strukture obilježja kao statičke strukture

I – ***Integritetna komponenta***

- Služi za modelovanje ograničenja nad podacima

O– ***Operacijska komponenta***

- Služi za modelovanje dinamike izmjene stanja podataka i šeme BP

Osnovni koncepti *relacionog* modela podataka

- **Torka** predstavlja jedan realni entitet ili poveznik. Pomoću torke se svakom obilježju, iz nekog skupa obilježja, dodjeljuje konkretna vrijednost, definisana domenom[1] datog obilježja. Formalna reprezentacija:

  $t : A \rightarrow DOM$
  $(\forall Ai \in U)(t(Ai) \in dom(Ai))$
  Primjer:
  t1={(BrIndeksa, IT80/2019), (Ime, Sanja), (Prezime, Tica),(Godiste,2000)}

- **Šema relacije N**predstavlja uređenu dvojku (R,O). To je konačan skup atributa R, sa konačnim skupom ograničenja O nad tim atributima. Šemom relacije se tumače svojstva neke klase objekata ili veza između realnih entiteta.

- **Relacija** nad nekim skupom obilježja A predstavlja konačan skup torki i reprezentuje skup realnih entiteta ili poveznika. Relacija nad n skupova predstavlja podskup Dekartovog proizvoda[2] tih n skupova. Formalna reprezentacija:

  $$r(U) \subseteq \{t \mid t: A \rightarrow DOM\}, \mid r \mid \in \mathbb{N}0 \}$$

  Primjer:

BrIndeksa	Ime	Prezime	Godiste
IT80/2019	Sanja	Tica	2000

[1]Domen obilježja: skup svih mogućih vrijednosti koje neko obilježje može da uzme
[2]Dekartov proizvod skupova A i B : Skup uređenih parova (a,b) kod kojih je prvi element iz skupa A, a drugi iz skupa B

IT67/2018	Jovana	Jovanovic	1999

Tabela 12 : Primjer relacije

- **Ključ šeme relacije**

  Ključ šeme relacije prestavlja podskup skupa obilježja šeme relacije, kojim se jednoznačno identifikuje data šema relacije. Ključ šeme relacije zadovoljava sledeća 2 svojstva:

  1. *Jedinstvenost ključa*: ne mogu postojati 2 torke sa identičim skupom atributa koji predstavljaju ključ
  2. *Minimalnost ključa*: ne postoji pravi podskup skupa K (skupa atributa ključa) takav da i dalje važi svojstvo 1.

  Ekvivalentni ključevi: Relacija može da ima više ključeva, koji su međusobno jednaki, i svi predstavljaju „kandidata“ za primarni ključ, ali se za primarni ključ uvijek bira samo jedan od njih, kao predstavnik ekvivalentnih ključeva.

- **Šema relacione BP**

  Šema relacione BP je imenovani par (S,I), gdje S predstavlja konačan skup šema relacija, a I predstavlja skup ograničenja koja važe među njima. Šema relacione BP je definicija relacione baze podataka.

- **Relaciona baza podataka**

  Relaciona baza podataka je konačan skup relacija nad šemom relacione BP.

RELACIONI MODEL PODŠEME

KUPAC ({KUPACID, IME, PREZIME, ADRESA, TELEFON, EMAIL, DATUM_RODJENJA}, {KUPACID})

VRSTA_PROIZVODA({VRSTAPROIZVODAID, NAZIV}, {VRSTAPROIZVODAID})

VRSTA_PLACANJA({VRSTAPLACANJAID,NAZIV}, {VRSTAPLACANJAID})

PROIZVODNA_OPERACIJA ({OPERACIJAID, NAZIV, OPIS, TIP_MASINE, RIZIK, DUZINA_TRAJANJA}, {OPERACIJAID})

FAKTURA ({FAKTURAID, NACIN_PLACANJA, JEDINICA_MJERE, VALUTA, PORUCENA_KOLICINA, TIP_FAKTURE, USLOV, DATUM_IZDAVANJA, PDV}, {FAKTURAID})

ODLUCUJE ({KUPACID, VRSTAPLACANJAID}, {KUPACID+VRSTAPLACANJAID})

PLACANJE ({PLACANJEID, BROJ_RATA, IZNOS_RATE, VRSTA_PLACANJAID, FAKTURAID}, {PLACANJEID+VRSTAPLACANJAID})

PROIZVOD ({PROIZVODID, NAZIV, OPIS, VELICINA_SERIJE, JEDINICA_MJERE, SIGNALNE_ZALIHE, KATEGORIJA, VRSTAPROIZVODAID}, {PROIZVODID+VRSTAPROIZVODAID})

NASTAJE ({VRSTAPROIZVODAID, PROIZVODID, OPERACIJAID}, {VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID})

KUPUJE ({VRSTAPROIZVODAID, PROIZVODID, OPERACIJAID, KUPACID}, {VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID + KUPACID})

NALAZI_SE ({VRSTAPROIZVODAID, PROIZVODID, OPERACIJAID, KUPACID, FAKTURAID}, {VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID + KUPACID + FAKTURAID})

KUPAC [KUPACID] ⊆ ODLUCUJE [KUPACID]

ODLUCUJE [KUPACID] ⊆ KUPAC [KUPACID]

ODLUCUJE [VRSTAPLACANJAID] ⊆ VRSTA_PLACANJA [VRSTAPLACANJAID]

PLACANJE [VRSTAPLACANJAID] ⊆ VRSTA_PLACANJA [VRSTAPLACANJAID]

PLACANJE [FAKTURAID] ⊆ FAKTURA [FAKTURAID]

NULL(PLACANJE, FAKTURAID) = ⊥

NASTAJE [OPERACIJAID] ⊆ PROIZVODNA_OPERACIJA [OPERACIJAID]

NASTAJE [VRSTAPROIZVODAID + PROIZVODID] ⊆ PROIZVOD [VRSTAPROIZVODAID + PROIZVODID]

PROIZVOD [VRSTAPROIZVODAID] ⊆ VRSTA_PROIZVODA [VRSTAPROIZVODAID]

PROIZVOD [VRSTAPROIZVODAID + PROIZVODID] ⊆NASTAJE [VRSTAPROIZVODAID + PROIZVODID]

KUPUJE [KUPACID] ⊆ KUPAC [KUPACID]

KUPUJE [VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID] ⊆ NASTAJE [VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID]

KUPUJE [VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID +KUPACID] ⊆ NALAZI_SE [VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID +KUPACID]

NALAZI_SE [FAKTURAID] ⊆ FAKTURA [FAKTURAID]

NALAZI_SE [VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID +KUPACID] ⊆ KUPUJE [VRSTAPROIZVODAID + PROIZVODID + OPERACIJAID +KUPACID]

DDL

Data Definition Language – jezik za definiciju podataka. Koristi se za specifikaciju konceptualne i interne šeme (fizičke strukture podataka). Definiše sintaksu za kreiranje objekata na kojima se zasniva baza podataka kao što su tabele, šeme, korisnici, indeksi…

Komande ovog jezika koriste se za promjenu strukture baze podataka, kao što je kreiranje novih tabela ili objekata sa svim njihovim atributima (tip podatka, naziv tabele…). Najčešće naredbe koje se koriste : CREATE, ALTER, DROP, TRUNCATE, DELETE.

Primjeri:

- **Kreiranje tabela:**

```
CREATE TABLE KUPAC (
        KUPACID NUMERIC (8) NOT NULL,
        IME VARCHAR (15) NOT NULL,
        PREZIME VARCHAR (20) NOT NULL,
        ADRESA VARCHAR (35),
        EMAIL VARCHAR (30) NOT NULL,
        DATUM_RODJENJA DATE
)
```

- NUMERIC (8), VARCHAR (25) – tip i dužina podatka
- (NOT) NULL – dozvola/zabrana dodjele NULL vrijednosti obilježja

- **Modifikacija tabela:**

-Brisanje obilježja

```
ALTER TABLE KUPAC
DROP COLUMN DATUM_RODJENJA
```

-Modifikacija obilježja tabele

```
ALTER TABLE KUPAC
ALTER COLUMN ADRESA VARCHAR (30) NOT NULL
```

DML

Data Manipulation Language – jezik za manipulaciju nad podacima. Koristi se za dodavanje, brisanje I izmjenu podataka. Za razliku od DDL-a ne dozvoljava promjenu strukture baze podataka, već samo manipulacije nad konkretnim podacima. Za izvršavanje naredbi DML jezika, obično je potrebna dozvola. Neke klase aplikacija mogu biti ograničene na read-only operacije, što znači da ne postoji dozvola za bilo kakvu promjenu podataka, već samo iščitavanje, čime se mogu izbjeći zlonamjerni napadi na sadržaj baze podataka, ili bilo kakav oblik štete koju napadač može prouzrokovati ako nađe ranjivosti u aplikaciji. Najčešće korišćene komande DML jezika : INSERT, UPDATE, DELETE.

Primjeri:

- **Popunavanje tabela**

  INSERT INTO KUPAC (KUPACID, IME, PREZIME, ADRESA, EMAIL, DATUM_RODJENJA)
  VALUES (1, ‘SANJA’, ‘TICA’, ‘BULEVAR DESPOTA STEFANA 7A’, ‘TICASANJA@GMAIL.COM’, ‘2000-08-02’)

- **Brisanje torke iz tabele**

  DELETE
  FROM KUPAC
  WHERE KUPACID = 1

- **Modifikacija obilježja u tabeli**

  UPDATE KUPAC
  SET ADRESA = ‘ PETRA DRAPŠINA 14’
  WHERE KUPACID = 1

SQL

Structured Query Language – jezik za postavljanje upita nad bazom podataka. SQL je deklarativan, visoko standardizovani upitni jezik, koji služi za iščitavanje podataka iz baze. Uniforman je, jer se kao rezultat izražavanja upita dobija tabela popunjena podacima koji zadovoljavaju uslov specificiran u okviru upita.

SQL je standardni jezik za relacione baze podataka. Većina relacionih sistema za upravljanje bazama podataka kao što su MySQL, Oracle, Postgres, MS Access i SQL Server koristi SQL za svoj podrazumjevani jezik za rad sa bazom podataka.

Primjeri:

- **Iščitavanje podataka iz tabela naredbom SELECT**

```
SELECT KUPACID, IME, PREZIME, DATUM_RODJENJA
FROM KUPAC
ORDER BY IME ASC, DATUM RODJENJA DESC
```

```
SELECT COUNT(FAKTURAID), DATUM_IZDAVANJA, IZNOS
FROM FAKTURA F INNER JOIN PLACANJE P ON F.FAKTURAID = P.FAKTURAID
WHERE DATUM_IZDAVANJA = '2022-03-25'
GROUP BY DATUM_IZDAVANJA, IZNOS
ORDER BY IZNOS DESC
```

OBJEKTI

Prilikom rada na projektu korišćeni su brojni objekti koji se inače koriste za kreiranje i rad sa bazom podataka.

Kao okruženje za kreiranje baze i manipulaciju nad podacima korišćen je Microsoft SQL Server Management Studio.

Prvi korak pri kreiranju same baze podataka i tabela koje je čine jeste kreiranje šeme baze podataka u koju će biti smještene te tabele.

- **SQL SCHEMA**služi za logićko grupisanje objekata baze podataka. Ovo je koristan mehanizam za razdvajanje objekata baze podataka, za recimo različite aplikacije, prava pristupa, upravljanje administracijom baze podataka. Ne postoji maksimalan broj objekata koji mogu pripadati jednoj šemi. Pogodnosti korišćenja SQL šema su brojne :
  - Prenos vlasništva nad SQL šemom među korisnicima
  - Dijeljenje šeme među korisnicima
  - Mogućnost prebacivanja objekata između šema
  - Veća kontrola nad pristupom i bezbjednošću baze podataka

  Sintaksa za kreiranje šeme :

```
IF SCHEMA_ID (‘MetalnaGalanterija’) IS NOT NULL
DROP SCHEMA MetalnaGalanterija
GO        -- provjera da li  u bazi već postoji ovakva šema

CREATE SCHEMA MetalnaGalanterija
GO        -- kreiranje šeme
```

- **SQL TABELE** odgovaraju relacijama, kao osnovnom konceptu relacionog modela podataka, pri prelasku na konstrukciju baze podataka. Tabela je skup podataka ,koji su organizovani po modelu vertikalnih kolona i horizontalnih redova. Kolone predstavljaju atribute relacije, a redovi su konkretne vrijednosti tih atributa, odnosno torke. Prva kolona je najčešće primarni ključ, kojim se jednoznačno identifikuje svaka torka u okviru date tabele. Pored toga, svaki atribut ima svoj tip (i dužinu) podataka, te podatak o tome

da li je moguće dodijeliti null vrijednosti obilježju, ili je ono ipak obavezno. U tabeli se ne mogu pojaviti 2 torke sa identičnim vrijednostima svakog obilježja. Tabela je zapravo osnovni gradivni blok relacionih baza podataka, i način na koji se podaci u okviru njih organizuju.

Sintaksa za kreiranje SQL tabela

```
IF OBJECT_ID ('MetalnaGalanterija.KUPAC','U') IS NOT NULL
DROP TABLE MetalnaGalanterija.KUPAC
GO          -- provjera da li u bazi već postoji ovakva tabela

CREATE TABLE MetalnaGalanterija.KUPAC (
        KUPACID NUMERIC (8) NOT NULL,
        IME VARCHAR (15) NOT NULL,
        PREZIME VARCHAR (20) NOT NULL,
        ADRESA VARCHAR (35),
        EMAIL VARCHAR (30) NOT NULL,
        DATUM_RODJENJA DATE
        CONSTRAINT PK_KUPAC PRIMARY KEY (KUPACID)
)           -- kreiranje tabele i definicija primarnog ključa na nivou tabele
```

- **SQL SEKVENCE**predstavljaju automatski način za dodjelu vrijednosti primarnih ključeva u tabelama, koja generiše niz numeričkih vrijednosti u skladu sa specifikacijom prema kojoj je kreirana. Obzirom da svaka vrijednost primarnog ključa u okviru tabele mora biti jedinstvena, sekvence predstavljaju dobar način da se to postigne, obzirom da SUBP samostalno dodjeljuje ove vrijednosti, vodeći računa o tome da ne dođe do konflikta, odnosno da se nikada ne pojave 2 torke sa identičnim ključem. U okviru sekvence sami možemo odrediti od kog broja želimo da sekvenca krene, koliki će biti "korak" sekvence, odnosno za koliko će naredna vrijednost biti

veća (ili manja), te da li će sekvenca biti ciklična ili ne. Razlika između ciklične i
i neciklične jeste u tome do kog broja maksimalno može ići sekvenca. Ako je cycle, to znači da čak i kada dosegne svoj maksimum, sekvenca će krenuti ispočetka, od najmanje definisane vrijednosti i tako u krug. Ako je sekvenca no cycle, jednom kada dosegne maksimalnu vrijednost, neće se opet vraćati na početak, do minimalne.

Sintaksa za kreiranje sekvenci

```
IF EXISTS (SELECT * FROM SYS.SEQUENCES WHERE NAME= 'MetalnaGalanterija.kupacSekv'
DROP SEQUENCE MetalnaGalanterija.kupacSekv
GO          -- provjera da li ova sekvenca već postoji u bazi

CREATE SEQUENCE MetalnaGalanterija.kupacSekv
MINVALUE 1
START WITH 1
CYCLE       -- kreiranje sekvence, koja u ovom slućaju kreće od 1, ima "korak" 1 i ciklična je
```

➢ SQL FUNKCIJE

SQL korisnički definisane funkcije služe da bi se pokrili scenariji koje nije jednostavno (ili uopšte moguće) pokriti upotrebom SQL upita, ili nekog drugog mehanizma u SQL-u. Pomoću njih imamo mogućnost dobijanja informacija za različite proslijeđene vrijednosti istovremeno, gdje ne moramo da, recimo, zadamo šifru jednog konkretnog proizvoda i da vratimo vrstu samo tog proizvoda, već je to moguće uraditi za svaki proizvod u okviru tabele.

Mogu da budu skalarne funkcije, koje mogu prihvatiti više ulaznih parametara, ali vraćaju samo jednu izlaznu vrijednost, zatim funkcije koje svoj rezultat vraćaju u formi tabele, gdje postoje 2 različite vrste funkcija:

1. Inline table-valued funkcije
2. Multi statement table-valued funkcije

Razlike između ove 2 vrste funkcija je u tabelama koje vraćaju. U slučaju inine table-valued funkcija tabela nastaje najčešće kao rezultat upita, i nije potrebno definisati njenu strukturu, dok kod multi statement table-valued funkcija, isto kao i kod klasičnog kreiranja tabela, definišemo strukturu, odnosno naziv tabele, obilježja i njihove tipove podataka, i neophodno je eksplicitno napuniti tabelu podacima.

Funkcije se mogu pozivati samostalno, u okviru sopstvenog select-a, kao dio nekog drugog upita, u uslovu kod where ili having klauzula, u okviru order by ili group by klauzula, values klauzule u insert naredbi, ili set klauzule u update-u.

Ne podržavaju boolean, record ili table tipove podataka, i ne mogu sadržati dml komande u okviru samog tijela funkcije.

Sintaksa za kreiranje funkcija:

```
if object_id('MetalnaGalanterija.VratiVrstu','F') is not null
drop function MetalnaGalanterija.VratiVrstu;
go--provjera da li funkcija možda već postoji u bazi

create function MetalnaGalanterija.VratiVrstu
(
	@ProizvodID as numeric   –specifikacija ulaznih parametara
)
returns varchar(25)   --specifikacija tipa izlaznog parametra
as
begin
declare @vrstaProizvoda as varchar(25)--ovde deklarisemo promjenljivu u koju smjestamo rezultat upita, koji bi trebao da nam vrati vrstu proizvoda kojoj proizvod sa proslijeđenim id-em pripada:
= (select naziv from MetalnaGalanterija.VRSTA_PROIZVODA vp inner
join MetalnaGalanterija.PROIZVOD p on
vp.VrstaProizvodaID=p.VrstaProizvodaID
where p.ProizvodID=@ProizvodID
```

```
return @vrstaProizvoda;    --rezultat funkcije ; izlazni parametar
end
go
```

➢ SQL PROCEDURE

Procedure imaju ulogu sličnu funkcijama, međutim, među njima i dalje postoje neke suštinske razlike. Procedure, za razliku od funkcija ne mogu da se pozivaju u okvru nekog drugog upita, ili neke klauzule, već se izvršavaju same za sebe.

Procedura može imati svoje ulazne parametre, koji mogu biti ili opcioni, ili obavezni. Ukoliko je postavljena neka podrazumjevana vrijednost parametra, taj parametar se smatra opcionim, međutim ukoliko je za parametar definisan samo tip podatka, onda se on smatra obaveznim, te prilikom samog pozivanja procedure neophodno je proslijediti vrijednost koja će odgovarati ovakvom parametru. Izlazni parametri procedure su takođe opcioni, i oni se označavaju sa OUT ili OUTPUT da bi se naglasilo da su ovo parametri očekivani na izlazu. Sa parametrima u okviru procedura se postupa slično kao i sa varijablama. Za preuzimanje izlaznih vrijednosti, prilikom samog poziva procedure potrebno je deklarisati promjenljivu u koju ćemo smjestiti sadržaj izlaznog parametra, te taj sadržaj prikazati.

Tijelo procedure se definiše između BEGIN END bloka.

Sintaksa za kreiranje procedura:

```
if object_id('MetalnaGalanterija.IznosFakture','P')is not null
drop proc MetalnaGalanterija.IznosFakture;
go      --provjera da li procedura postoji u bazi

create proc MetalnaGalanterija.IznosFakture
@fakturaid as numeric--obavezni ulazni parametar procedure
as
begin
```

```
declare @kolicina as numeric = (select porucena_kolicina from
MetalnaGalanterija.FAKTURA where FakturaID=@fakturaid)

declare @cijena as numeric = (select cijena from
MetalnaGalanterija.PROIZVOD p inner join
MetalnaGalanterija.NALAZI_SE n on p.ProizvodID=n.ProizvodID where
n.FakturaID = @fakturaid)
declare @iznos = @cijena * @kolicina
print @iznos;     -- procedura vraca @iznos
end
```

➢ SQL TRIGERI

Trigeri predstavljaju posebne proceduralne mehanizme, koji se koriste za implementaciju specifičnih pravila poslovanja i složenih ograničenja. Oni se ne mogu eksplicitno pozvati, već se izvršavaju automatski. Nazivaju se i okidači, i aktivira ih neka DML naredba, INSERT, UPDATE ili DELETE.
Postoje 2 vrste okidača:
AFTER i INSTEAD OF trigeri.
AFTER trigeri izvršavaju se nakon određene DML naredbe za koju su definisani, a nad tabelom nad kojom je triger definisan.
INSTEAD OF trigeri izvršavaju se umjesto određene DML naredbe za koju su definisani. Za razliku od after trigera, gdje je potrebno da se neka naredba izvrši da bi triger bio pokrenut, u slučaju INSTEAD OF čim se naredba pozove, triger će se izvršiti, umjesto same naredbe.
Triger ima svoju oblast definisanosti, odnosno tabelu nad kojom se aktivira, zatim, specifikaciju operacije koja pokreće aktivaciju trigera, te aktivnost (proceduru), koja bi se trebala izvršiti kao posljedica pokretanja trigera.

Sintaksa za kreiranje trigera:

```
if object_id('MetalnaGalanterija.triger1','TR')is not null
drop trigger MetalnaGalanterija.triger1;
go

create trigger MetalnaGalanterija.triger1
on MetalnaGalanterija.KUPAC    --oblast definisanosti
```

```
after INSERT, UPDATE    --specifikacija operacija koje ga pokreću
as
begin
        if @@rowcount = 0
        set nocount on;

        select * from inserted;      -- tabele koje postoje u trenutku aktivacije i
        select * from deleted;       -- izvršenja trigera

print 'After insert/update trigger';

go

Poziv:
Insert into kupac
Values
 (52,'Petar','Jovanovic','Vojvodjanska
 1','0648956231','petarjov@gmail.com','1989-07-12');
```

➢ SQL KURSORI

Kursori su posebni objekti koji se koristite najčešće unutar uskladištenih procedura, u situacijama kada je, recimo, za realizaciju procedure potrebno preuzeti više od jedne vrijednosti koju neki upit vraća. Tada se koristi kursor, za prolazak kroz taj niz vrijednosti, i odgovarajuće manipulacije nad preuzetim vrijednostima.

Sintaksa za kreiranje kursora

```
declare @fakturaid as numeric
declare racun_cursor cursor for
    select FakturaID from MetalnaGalanterija.NALAZI_SE
                        --deklaracija kursora, nad kojom tabelom i za koja
                        obilježja će se koristiti
open racun_cursor--otvaranje kursora
fetch next from racun_cursor into @fakturaid
```

```
                        --preuzimanje sledeće vrijednosti za FakturaID
while @@fetch_status = 0
                         --ako je status 0, fetch je uspješno izvšen
begin
   exec proc MetalnaGalanterija.IznosFakture @fakturaid;
fetch next from racun_cursor into @fakturaid
end
close racun_cursor
deallocate racun_cursor
                        --obavezno zatvaranje kada završimo sa upotrebom
                        kursora, te njegova dealokacija
```

UPITI

Prvi upit

1. Za svaku fakturu koja je evidentirana u sistemu iščitati uslov fakturisanja, ako postoji, a ako ne postojiispisati poruku 'Nije naveden uslov fakturisanja', ispisati način na koji se isplaćuje faktura, ako je u ratama ispisati i iznos i broj rata, a ako nije ispisati 0, u formatu:
   1. Uslov fakturisanja
   2. Nacin isplate
   3. Broj i iznos rata

```
selectisnull(uslov,'Nije naveden uslov fakturisanja')as "Uslov fakturisanja",
 nacin_placanja as "Nacin isplate",
iif(convert(varchar,pl.VrstaPlacanjaID)='2',convert(varchar,pl.broj_rata)+',
'+convert(varchar,pl.iznos_rate),'Placanje se vrsi gotovinski')as "Broj i iznos rata"
from MetalnaGalanterija.FAKTURA f innerjoin MetalnaGalanterija.PLACANJE pl on f.FakturaID=pl.FakturaID
```

Rezultat:

	Uslov fakturisanja	Nacin isplate	Broj i iznos rata
1	Nije naveden uslov fakturisanja	1	7, 200
2	Nije naveden uslov fakturisanja	1	4, 3000
3	Nije naveden uslov fakturisanja	1	5, 1000
4	Nije naveden uslov fakturisanja	2	6, 12000
5	Minimalna porucena kolicina 124l	3	Placanje se vrsi gotovinski
6	Nije naveden uslov fakturisanja	2	7, 950
7	Nije naveden uslov fakturisanja	3	4, 3000
8	Nije naveden uslov fakturisanja	2	Placanje se vrsi gotovinski
9	Eelktronska faktura	2	Placanje se vrsi gotovinski
10	Fakturu generise komercijalista	3	Placanje se vrsi gotovinski
11	Nije naveden uslov fakturisanja	1	9, 2698
12	Nije naveden uslov fakturisanja	1	10, 1200

Slika 3: Rezultat izvršavanja upita 1

Rezultat upita u ovom slučaju je 12 torki, iz razloga što postoji evidentirano 12 torki u čitavoj tabeli FAKTURA, i neke od njih imaju uslov, a neke ne. Za one koje nemaju definisan uslov, ispisace se string „Nije naveden uslov fakturisanja“, a za one koje imaju bice ispisan konkretan uslov, takođe u vidu stringa. Za

realizaciju je korišćena funkcija isnull, koja u ovom upitu provjerava da li je obilježje uslov null, ako jeste, ispisuje odgovarajuću poruku, a ako nije ispisuje uslov. Da bi svi traženi podaci bili adekvatno prikazani, neophodan je join tabela. Join se koristi za spajanje tabela po određenom obilježju (najčešće primarni ili strani ključ), a kako bismo mogli u okviru istog upita da pristupimo podacima različitih tabela. U ovom slučaju spajane su tabele FAKTURA i PLACANJE, po obilježju FakturaID, koje se nalazi kao strani ključ u tabeli PLACANJE.

Drugi upit

2. Za sve kupce čije ime počinje na slovo M ili su rođeni 2000. godine ispisati podatke o imenu, prezimenu, datumu rođenja i podatak o tome koliko su puta kupovali proizvode koji su rezultat rada metalne galanterije sortirano po imenu u formatu :

1. ime, prezime as Kupac
2. datum_rodjenja as Godine
3. Broj kupovina

```
select ime +' '+ prezime as "Kupac", datum_rodjenja as "Godine",count(ns.KupacID)as "Broj kupovina"
from MetalnaGalanterija.KUPAC k innerjoin MetalnaGalanterija.KUPUJE ku on k.KupacID=ku.KupacID innerjoin
MetalnaGalanterija.NALAZI_SE ns on ku.KupacID=ns.KupacID
where ime like'M%'ordatepart(year,datum_rodjenja)= 2000
groupby ime, prezime, datum_rodjenja
orderby ime;
```

Rezultat:

	Kupac	Godine	Broj kupovina
1	Marijana Simic	2000-11-06	1
2	Marko Markovic	2000-04-21	4
3	Sanja Tica	2000-08-02	4

Slika 4: Rezultat izvršavanja upita 2

Rezultat upita su u ovom slučaju 3 torke koje zadovoljavaju postavljene uslove, a to je da ime mora počinjati slovom M (provjera zadovoljenosti ovog uslova vrši se pomoću operatora %, gdje možemo proslijediti ili recimo prvi, ili poslednji

karakter kojim počinje tražena vrijednost, ili neke karaktere u sredini) ili da kupac mora kupiti više od jednog proizvoda koje je preduzeće proizvelo, a podatak o broju kupovina možemo dobiti tako što spojimo tabele KUPAC i KUPUJE, da bismo znali koje proizvode je neki kupac kupio, a zatim spajamo i tabelu Nalazi_se da bismo došli do fakture, odnosno evidencije o tome čiji ID (kog kupca) se nalazi na kojoj fakturi, odakle pomoću funkcije count dolazimo do broja kupovina koje je kupac obavio.

Treći upit

3.Za proizvodne operacije koje traju manje od svih proizvodnih operacija čije vrijeme trajanja je u minutama,ispisati naziv, rizik, i tip mašine koja se koristi za realizaciju ove operacije, te dužinu trajanja u minutama.

```
select o.naziv, o.rizik, o.tip_masine, duzina_trajanja
from MetalnaGalanterija.PROIZVODNA_OPERACIJA o innerjoin MetalnaGalanterija.NASTAJE n on
o.OperacijaID=n.OperacijaID
where o.duzina_trajanja <(selectavg(duzina_trajanja)as duzinaTrajanja from
MetalnaGalanterija.PROIZVODNA_OPERACIJA where vrijeme_trajanja='min')and vrijeme_trajanja ='min'
```

Rezultat:

	naziv	rizik	tip_masine	duzina_trajanja
1	Rezanje	Moze izazvati posjekotine pri neadekvatnom koris...	Rezaci	25
2	Laserska obrada	Moguce opekotine	Laser	20
3	Rezanje	Moze izazvati posjekotine pri neadekvatnom koris...	Rezaci	25
4	Laserska obrada	Moguce opekotine	Laser	20
5	Rezanje	Moze izazvati posjekotine pri neadekvatnom koris...	Rezaci	25

Slika 5: Rezultat izvršavanja upita 3

Kao rezultat upita izlistaće se 5 torki koje zadovoljavaju dati uslov. Upit je realizovan preko podupita u where klauzuli, gdje je pronađena operacija koja traje najkraće, od svih koje su izražene u minutama, i na osnovu te informacije postavljen je uslov da se iščitaju sve operacije koje traju manje od te dobijene vrijednosti. Radi prikaza svih obilježja, takođe je bio potreban join određenih tabela.

Četvrti upit

4. Za onu vrstu proizvoda koja je najviše kupovana, ispisati podatak o tome kolika je ukupna proizvedena količina na pocetku, te koliko je od te ukupno proizvedene količine proizvoda prodato.

```
select vp.naziv as Vrsta,sum(porucena_kolicina)as "Ukupno prodato",sum(velicina_serije)as "Ukupno
proizvedeno"
from MetalnaGalanterija.VRSTA_PROIZVODA vp innerjoin MetalnaGalanterija.NALAZI_SE ns on
vp.VrstaProizvodaID=ns.VrstaProizvodaID  innerjoin
MetalnaGalanterija.FAKTURA f on ns.FakturaID=f.FakturaID innerjoin MetalnaGalanterija.PROIZVOD p on
ns.ProizvodID=p.ProizvodID
innerjoin(selecttop 1 sum(porucena_kolicina)as maksProdato,ns.VrstaProizvodaID,vp.Naziv
from MetalnaGalanterija.VRSTA_PROIZVODA vp
innerjoin MetalnaGalanterija.NALAZI_SE ns on vp.VrstaProizvodaID=ns.VrstaProizvodaID
innerjoin MetalnaGalanterija.FAKTURA f on ns.FakturaID=f.FakturaID
groupby ns.VrstaProizvodaID,vp.Naziv) novaTab on ns.VrstaProizvodaID=novaTab.VrstaProizvodaID
groupby vp.Naziv,maksProdato
havingsum(porucena_kolicina)=maksProdato
```

Rezultat:

	Vrsta	Ukupno prodato	Ukupno proizvedeno
1	Alati i oprema	1716	2866

Slika 6: Rezultat izvršavanja upita 4

U rezultatu ovog upita dobijaju se samo 1 torka, što je i bio prvobitni cilj, iz razloga što od svih 6 vrsta proizvoda, jedna je najprodavanija, što je u ovom slučaju vrsta Alati i oprema. Da bismo došli do podataka o tome kolika je maksimalna prodata (poručena) količina proizvoda, iz tabele VRSTA_PROIZVODA morali smo preko niza odgovarajućih tabela doći do tabele FAKTURA gdje se nalazi podatak o tome koliko je poručenih proizvoda ove vrste evidentirano na svim fakturama. Kako bismo pronašli onu vrstu koja je najprodavanija, iskorišćena je funkcija top 1, u ovom slučaju od sume svih poručenih proizvoda, za šta nam je bio potreban podupit u from klauzuli, kao jedna dodatna tabela, sa kojom spajamo sve što smo prethodno spojili. Podatak o ukupnoj proizvedenoj količini pronašli smo u tabeli PROIZVOD, preko VrstaProizvodaID.

Peti upit

5. Za svaku od kategorija proizvoda, ispisati podatke o tomekoliko proizvoda svake kategorije je prodato, i koji su to proizvodi, kod onih kategorija kod kojih je ukupna količina prodatih proizvoda jednaka ili maksimalnoj prodatoj količini nekog proizvoda ili minimalnoj prodatoj, a čiji proizvodi imaju cijenu manju od prosječne cijene svih proizvoda.

```
select vp.naziv as "Kategorija proizvoda", p.naziv as "Proizvod",count(n.ProizvodID)as "Broj prodatih
proizvoda",porucena_kolicina as "Porucena kolicina"
from MetalnaGalanterija.VRSTA_PROIZVODA vp leftjoin MetalnaGalanterija.PROIZVOD p
on vp.VrstaProizvodaID=p.VrstaProizvodaID innerjoin MetalnaGalanterija.NALAZI_SE n on
n.ProizvodID=p.ProizvodID innerjoin
 MetalnaGalanterija.FAKTURA f on n.FakturaID=f.FakturaID innerjoin
(select n.ProizvodID,max(porucena_kolicina)as maksKolicina from MetalnaGalanterija.PROIZVOD p innerjoin
MetalnaGalanterija.NALAZI_SE n on
 p.ProizvodID = n.ProizvodID innerjoin MetalnaGalanterija.FAKTURA f on n.FakturaID=f.FakturaID groupby
n.ProizvodID)maksProdavani on n.ProizvodID=maksProdavani.ProizvodID
innerjoin(select n.ProizvodID,min(porucena_kolicina)as minKolicina from MetalnaGalanterija.PROIZVOD p
innerjoin MetalnaGalanterija.NALAZI_SE n on
 p.ProizvodID = n.ProizvodID innerjoin MetalnaGalanterija.FAKTURA f on n.FakturaID=f.FakturaID groupby
n.ProizvodID)minProdavani on n.ProizvodID=minProdavani.ProizvodID
where p.cijena <(selectavg(cijena)from MetalnaGalanterija.PROIZVOD)
groupby vp.Naziv,p.naziv,minKolicina,maksKolicina,porucena_kolicina
having f.porucena_kolicina = minKolicina or f.porucena_kolicina= maksKolicina
```

Rezultat:

	Kategorija proizvoda	Proizvod	Broj prodatih proizvoda	Porucena kolicina
1	Alati i oprema	cekic	1	150
2	Alati i oprema	Sraf	1	230
3	Elektrode	Anoda	1	396
4	Elektrode	Daska	1	280
5	Elektrode	Katoda	1	412
6	Elektrode	Makaze	1	150
7	Vijcana roba i ekseri	Celicni ekser	1	485
8	Vijcana roba i ekseri	Produzni kabl	1	145
9	Zastitna oprema	Kaciga	1	1000

Slika 7: Rezultat izvršavanja upita 5

Rezultat izvršavanja upita je 9 torki koje zadovoljavaju uslov. Minimalna i maksimalna količina poručenih proizvoda su pronađeni preko podupita u from-u, ali da bismo došli do tog podatka potrebno je uraditi join nekoliko tabela, počevši od PROIZVOD, jer su potrebni i ostali podaci i iz ove tabele, ne samo ID, a da bismo došli do podatka o tome koja je prosječna poručena količina, potrebna je

tabela FAKTURA i tabela NALAZI_SE, a da bismo provjerili zadovoljenost uslova da je cijena manja od prosječne potreban je podupit u where klauzuli.

FUNKCIJE

Prva funkcija

1.Za svaki proizvod iz baze, prikazivati podatke o tome koji kupac je kupio proizvod, njegov naziv i vrstu, kolika mu je cijena, i na osnovu podatka o pdv-u, koliko kupac mora dodatno platiti na osnovnu cijenu proizvoda, te prikazati konačno cijena+pdv, u formi tabele

```
ifobject_id('MetalnaGalanterija.PrikaziKupovinuProiz','FN')isnotnull
dropfunction MetalnaGalanterija.PrikaziKupovinuProiz;
go

createfunction MetalnaGalanterija.PrikaziKupovinuProiz
(
        @proizvodid asnumeric (8)
)
returns @tabela table
(
        Kupac varchar(40),
        Proizvod varchar(30),
        Cijena numeric(8),
        Vrsta varchar(25),
        PDV numeric(3),
        Ukupno numeric(5,2)
)
as
begin
declare @pdv asnumeric(3,2)=(select pdv from MetalnaGalanterija.FAKTURA f innerjoin
MetalnaGalanterija.NALAZI_SE n on f.FakturaID=n.FakturaID where n.ProizvodID=@proizvodid)
declare @cijenaP asnumeric(8)=(select cijena from MetalnaGalanterija.PROIZVOD p where
ProizvodID=@proizvodid)
declare @cijenaSaPdv asnumeric(8)=(@pdv/100)*@cijenaP
declare @ukupno asnumeric (5,2)= @cijenaP + @cijenaSaPdv
        insert @tabela
        select ime +' '+prezime as Kupac,p.naziv as Proizvod, cijena as Cijena, v.naziv as Vrsta,
@cijenaSaPdv as PDV,@ukupno as Ukupno
        from MetalnaGalanterija.VRSTA_PROIZVODA V innerjoin MetalnaGalanterija.PROIZVOD p on
v.VrstaProizvodaID=p.VrstaProizvodaID
        innerjoin MetalnaGalanterija.KUPUJE k on p.ProizvodID=k.ProizvodID innerjoin
MetalnaGalanterija.KUPAC ku on k.KupacID=ku.KupacID
        where k.ProizvodID=@proizvodid
        return
end;
go

select Kupac,Proizvod,Cijena,Vrsta ,PDV,Ukupno
from MetalnaGalanterija.PrikaziKupovinuProiz(2);
```

Funkcija 1: MetalnaGalanterija.PrikaziKupovinuProiz

U ovoj funkciji cilj je prikazati podatke o nekom proizvodu, i to podatak o tome koji kupac je kupio proizvod (ime i prezime), zatim naziv i vrsta proizvoda, koliki su cijena i pdv na dati proizvod, i na kraju koliko je kupac dužan platiti za dati proizvod.

Korišćena je funkcija koja kao rezultat vraća podatke u formi tabele, te je bilo potrebno definisati strukturu tabele, odnosno koja obilježja će ona imati, i kog tipa podatka će ta obilježja biti, a u ovom slučaju tabela ima 6 obilježja koja bi trebala biti vraćena. Ukupna cijena je računata tako što se prvo izvuče podatak o cijeni proizvoda, zatim se obračuna pdv, i na kraju se ispiše ukupna cijena sa pdv-om.

Rezultat:

	Kupac	Proizvod	Cijena	Vrsta	PDV	Ukupno
1	Sanja Tica	Sraf	50	Alati i oprema	2	52.00

Slika 8: Rezultat poziva funkcije MetalnaGalanterija.PrikaziKupovinuProiz (za proslijeđeni ProizvodID =1)

Druga funkcija

2.Napraviti funkciju koja će prikazivati statistiku o tomekako su kupljeni proizvodi plaćeni (ako su plaćeni), da li na rate ili gotovinski.

```
ifobject_id('MetalnaGalanterija.statPlacanja','FN')isnotnull
dropfunction MetalnaGalanterija.statPlacanja;
go

createfunction MetalnaGalanterija.statPlacanja
(
	@proizvod asnumeric
)
returnsvarchar(120)
as
begin
declare @rezultat asvarchar(120);
declare @ukupnoProdato asnumeric=(selectsum(porucena_kolicina)from MetalnaGalanterija.FAKTURA f innerjoin
MetalnaGalanterija.NALAZI_SE ns on f.FakturaID=ns.FakturaID where ns.ProizvodID=@proizvod);
declare @vrsta asnumeric=(select VrstaProizvodaID from MetalnaGalanterija.PROIZVOD vp where
ProizvodID=@proizvod)
declare @operacija asnumeric=(select OperacijaID from MetalnaGalanterija.NASTAJE where
ProizvodID=@proizvod and VrstaProizvodaID=@vrsta)
```

```
declare @faktura asnumeric=(select FakturaID from MetalnaGalanterija.NALAZI_SE where ProizvodID=@proizvod)
declare @nacinPlacanja aschar=(select p.VrstaPlacanjaID from
MetalnaGalanterija.NALAZI_SE ns innerjoin MetalnaGalanterija.FAKTURA f on ns.FakturaID=f.FakturaID
innerjoin MetalnaGalanterija.PLACANJE p on f.FakturaID=p.PlacanjeID where ProizvodID=@proizvod);
if (@nacinPlacanja = 2)
begin
declare @rate asnumeric=(selectsum(porucena_kolicina)from MetalnaGalanterija.FAKTURA f innerjoin
MetalnaGalanterija.NALAZI_SE ns on f.FakturaID=ns.FakturaID
innerjoin MetalnaGalanterija.PLACANJE p on ns.FakturaID=p.FakturaID innerjoin
MetalnaGalanterija.VRSTA_PLACANJA vp on p.VrstaPlacanjaID=vp.VrstaPlacanjaID
where ns.ProizvodID=@proizvod and vp.Naziv='0');
set @rezultat ='Od ukupne kolicine prodatih proizvoda:'+convert(varchar,@ukupnoProdato)+', '
+'sve je placeno na rate.'
end
elseif (@nacinPlacanja = 1)
begin
declare @gotovina asnumeric=(selectsum(porucena_kolicina)from MetalnaGalanterija.FAKTURA f innerjoin
MetalnaGalanterija.NALAZI_SE ns on f.FakturaID=ns.FakturaID
innerjoin MetalnaGalanterija.PLACANJE p on ns.FakturaID=p.FakturaID innerjoin
MetalnaGalanterija.VRSTA_PLACANJA vp on p.VrstaPlacanjaID=vp.VrstaPlacanjaID
where ns.ProizvodID=@proizvod and vp.Naziv='1');
set @rezultat =('Od ukupne kolicine prodatih proizvoda: '+convert(varchar,@ukupnoProdato)+', '
+' sve je placeno gotovinski ');
end
else
set @rezultat='Proizvodi jos nije prodat.'
return @rezultat;
end
go
```

Funkcija 2: MetalnaGalanterija.statPlacanja

Rezultat:

	ProizvodID	naziv	Nacin placanja
1	1	Sraf	Od ukupne kolicine prodatih proizvoda: 150, sve j...
2	4	cekic	Od ukupne kolicine prodatih proizvoda: 280, sve j...
3	5	Busilica	Od ukupne kolicine prodatih proizvoda:966, sve j...
4	8	Agregat	Od ukupne kolicine prodatih proizvoda: 320, sve j...
5	12	Trodijelne alu merdevine	Od ukupne kolicine prodatih proizvoda:444, sve j...
6	6	Makaze	Od ukupne kolicine prodatih proizvoda: 320, sve j...
7	11	Kaciga	Od ukupne kolicine prodatih proizvoda: 128, sve j...
8	9	Celicni ekser	Od ukupne kolicine prodatih proizvoda:1000, sve ...
9	10	Produžni kabl	Od ukupne kolicine prodatih proizvoda:485, sve j...
10	2	Anoda	Od ukupne kolicine prodatih proizvoda: 80, sve je...
11	3	Katoda	Proizvodi jos nije prodat.
12	7	Daska	Proizvodi jos nije prodat.

Slika 9: Rezultat poziva funkcije MetalnaGalanterija.statPlacanjaz (za proslijeđeni ProizvodID =1)

Cilj funkcije je da se za svaki kupljeni proizvod ispiše podatak o tome kako je proizvod plaćen. Obzirom da kupac ima mogućnost plaćanja ili na rate ili gotovinski, proizvodi koji su evidentirani na fakturi mogu biti plaćeni ili na rate ili cjelokupan iznos odjednom. Do podatka o načinu plaćanja, dolazimo na osnovu tabele VRSTA_PLACANJA, gdje 2 označava plaćanje na rate, a 1 plaćanje

gotovinski. Kada smo izvukli podatak o tome na koji način je plaćen proizvod čiji id je proslijeđen, vrši se ispitivanje da li je dobijena 1, ili 2, kako bi se provjerila količina proizvoda plaćena na jedan ili drugi način, na osnovu čega se generiše odgovarajuća poruka. Rezultat možemo smjestiti kao string u neku privremenu promjenljivu, što će ujedno biti rezultat koji se ispisuje prilikom poziva funkciji.

PROCEDURE

Prva procedura

1.Za svakog kupca čiji id se nalazi na nekoj fakturi ispitati da li kupac ima pravo na popust, ili ne.
Ukoliko je kupac obavio više od jedne kupovine obračunava se popust od 5 %, i ispisuje se iznos za svaku fakturu,
u suprotnom se ispisuje poruka da kupac nema pravo na popust.

```
ifobject_id('MetalnaGalanterija.popust','P')isnotnull
dropproc MetalnaGalanterija.popust;
go

createproc MetalnaGalanterija.popust
 @kupacid numeric
as
begin
declare @brojPojavljivanja asnumeric=(selectcount(KupacID)from
 MetalnaGalanterija.NALAZI_SE where KupacID=@kupacid)
declare @popust asnumeric= 0.05;
if (@brojPojavljivanja > 1)
begin
declare @kupac asnumeric,
                 @faktura asnumeric
        declare racun_cursor cursorfor
                select KupacID,FakturaID from MetalnaGalanterija.NALAZI_SE where KupacID=@kupacid;
        open racun_cursor;
        fetchnextfrom racun_cursor into @kupac, @faktura
                while@@FETCH_STATUS= 0
                        begin
                        declare @proizvodid asnumeric=(select ProizvodID from
MetalnaGalanterija.NALAZI_SE where KupacID=@kupac and FakturaID=@faktura)
                        declare @racun asnumeric=(select cijena from MetalnaGalanterija.PROIZVOD p
                        innerjoin MetalnaGalanterija.NALAZI_SE ns on p.ProizvodID=ns.ProizvodID where
ns.KupacID=@kupac and FakturaID=@faktura and ns.ProizvodID=@proizvodid)
                        declare @iznos asnumeric= @racun - @racun * @popust;
                        print'Kupac ima pravo na popust od 5% i racun iznosi:'+'
'+convert(varchar,@iznos);
                        fetchnextfrom racun_cursor into @kupac,@faktura
                        end
                close racun_cursor;
                deallocate racun_cursor;
end
```

```
else
        print'Kupac je kupio samo jedan proizvod i nema pravo na popust.';
end;
return;

execute MetalnaGalanterija.popust1;
```

Procedura 1: MetalnaGalanterija.popust

Pozivom ove procedure, provjerava se da li neki kupac ima pravo na popust, na osnovu toga koliko je proizvoda kupio. Obzirom da se u tabeli NALAZI_SE za svakog kupca evidentira po jedan proizvod koji je kupio, potrebno je provjeriti koliko se puta zadati KupacID pojavljuje u tabeli. Ako se pojavljuje više od jednom, kupac ima pravo na popust od 5%.

Prolazak kroz tabelu i provjera da li se dati kupac pojavljuje više puta, realizuje se upotrebom kursora. Kursor je kreiran za 2 obilježja: KupacID i FakturaID, da bismo pored id-a kupca, mogli izvući podatak prvo o id-u proizvoda koji je kupljen, a zatim i o njegovoj cijeni, te se na osnovu toga računa ukupan iznos koji je kupac dužan da plati, sa popustom, u slučaju da je broj kupovina veći od 1. Ako je obavljena samo jedna kupovina, kupac nema pravo na popust i ispisuje se odgovarajuća poruka.

Rezultat:

```
Kupac ima pravo na popust od 5% i racun iznosi: 50
Kupac ima pravo na popust od 5% i racun iznosi: 990
```

Slika 10: Rezultat poziva procedure, ako kupac ima pravo na popust

```
Kupac je kupio samo jedan proizvod i nema pravo na popust.
```

Slika 11: Rezultat poziva procedure, ako kupac nema pravo na popust

Druga procedura

2. Za neku kategoriju proizvoda koja je proslijeđena kao ulazni parametar procedure ispitati koliko proizvoda te kategorije postoji, i ispisati odgovarajuće podatke:

Ako postoje proizvodi date kategorije, u zaglavlju ispisa treba da piše:
‘ U datoj kategoriji (naziv), postoje sledeci proizvodi: '
Redni broj, id proizvoda, naziv, koliko je proizvoda proizvedeno u jednoj seriji, na kojoj fakturi je evidentiran, kojoj vrsti pripada, i u okviru koje proizvodne operacije nastaje, sortirano po nazivu. Ispisati na kraju ukupan broj proizvoda date kategorije i prosječnu cijenu svih proizvoda. Ako nema proizvoda, ili je proslijeđena nepostojeća kategorija ispisati adekvatnu poruku.

```
ifobject_id('ProizvodiKat','P')isnotnull
dropproc ProizvodiKat;
go


createproc ProizvodiKat
@kategorija asvarchar(5)
as
begin
declare @broj asnumeric=(selectcount(ProizvodID)from MetalnaGalanterija.PROIZVOD where kategorija =
@kategorija)
declare @prosjecna asnumeric=(selectavg(cijena)from MetalnaGalanterija.PROIZVOD where
kategorija=@kategorija)
declare @brojac asnumeric=0
declare @proizvodid asnumeric
declare @proizvod asvarchar(25)
declare @serija asnumeric
declare @faktura asnumeric
declare @vrsta asvarchar(25)
declare @operacija asvarchar(50)
if(@broj>0 and @kategorija in(select kategorija from MetalnaGalanterija.PROIZVOD))
begin
        print'U datoj kategoriji'+@kategorija+' postoje sledeci proizvodi:'
        declare proizvodi_cursor cursorfast_forwardfor
        select p.ProizvodID,p.naziv,p.velicina_serije,f.FakturaID,vp.naziv,o.naziv
        from MetalnaGalanterija.PROIZVOD p leftjoin      MetalnaGalanterija.VRSTA_PROIZVODA vp on
p.VrstaProizvodaID=vp.VrstaProizvodaID
        leftjoin MetalnaGalanterija.NASTAJE n on p.ProizvodID=n.ProizvodID
        leftjoin MetalnaGalanterija.PROIZVODNA_OPERACIJA o on n.OperacijaID=o.OperacijaID
        leftjoin MetalnaGalanterija.NALAZI_SE ns on p.ProizvodID=ns.ProizvodID leftjoin
MetalnaGalanterija.FAKTURA f
        on ns.FakturaID=f.FakturaID
        where p.kategorija=@kategorija
        orderby p.naziv
        open proizvodi_cursor
        fetchnextfrom proizvodi_cursor into @proizvodid,@proizvod,@serija,@faktura,@vrsta,@operacija
        while@@FETCH_STATUS= 0
        begin
                set @brojac=@brojac+1
                printconvert(varchar,@brojac)+'. '+convert(varchar,@proizvodid)+', '+@proizvod+',
'+convert(varchar,@serija)+','+
                convert(varchar,@faktura)+','+@vrsta+','+@operacija
                fetchnextfrom proizvodi_cursor into
@proizvodid,@proizvod,@serija,@faktura,@vrsta,@operacija
        end
        close proizvodi_cursor
        deallocate proizvodi_cursor
        print'Ukupno proizvoda date kategorije: '+convert(varchar,@broj)
        print'Prosjecna cijena svih proizvoda ove kategorije: '+convert(varchar,@prosjecna)
end
elseif (@kategorija notin(select kategorija from MetalnaGalanterija.PROIZVOD))
begin
        print'Ne postoji proslijedjena kategorija.'
end
end
```

```
go
```

Procedura 2: MetalnaGalanterija.ProizvodiKat

Cilj izvršavanja ove procedure jeste ispisivanje određenih podataka o proizvodima neke kategorije. Kao ulazni parametar procedure proslijeđuje se kategorija proizvoda, za koju želimo da ispišemo određene podatke. Na osnovu kategorije, moguće je izvući podatke o broju proizvoda koji pripadaju toj kategoriji, i zatim na osnovu tog podatka generisati željene ispise. Ako postoje neki proizvodi koji pripadaju proslijeđenoj kategoriji, onda se, obzirom da je moguće da postoji više takvih proizvoda, koristi kursor. Kursor je definisan za sva obilježja, koja su u postavci zadatka procedure zahtjevana za ispis. Korišćen je left join, da bi se ispisale sve moguće torke, obzirom da ukoliko je neko od obilježja koje želimo da iščitamo null, onda će sql „pregaziti“ čitavu torku, i nijedno obilježje neće biti prikazano, što može da bude kontradiktorno sa brojem proizvoda neke kategorije koji smo dobili pomoću count funkcije. Da bi se izbjegle null vrijednosti, moguće je koristiti coalesce funkciju, koja će, ukoliko je neko obilježje null, proslijediti vrijednost koju je korisnik zadao, umjesto null. Drugi uslov koji treba biti zadovoljen je, da se, ako ne postoji data kategorija evidentirana u sistemu, korisniku ispiše odgovarajuća poruka, da bi u svakom trenutku imao uvid u rezultat izvršavanja pozivane procedure.

Rezultat:

```
U datoj kategorijiPZO postoje sledeci proizvodi:
1. 5, Busilica, 750,3,Alati i oprema,Elektrohemijska obrada
2. 6, Makaze, 500,1,Elektrode,Zarezivanje
Ukupno proizvoda date kategorije: 2
Prosjecna cijena svih proizvoda ove kategorije: 7995
```

Slika 12: Rezultat poziva procedure u slučaju da postoje proizvodi date kategorije

```
Ne postoji proslijedjena kategorija.
```

Slika 13: Rezultat poziva procedure u slučaju da ne postoji proslijeđena kategorija

TRIGERI

Prvi triger

1.Napisati triger koji implementira poslovno pravilo da kupac ne može kupiti neku količinu neke vrste proizvoda, koja je veća nego što je ukupna proizvedena količina te vrste proizvoda na skladištu.

```
ifobject_id('MetalnaGalanterija.KupljenaKolicina','TR')isnotnull
droptrigger MetalnaGalanterija.KupljenaKolicina
go

createtrigger MetalnaGalanterija.KupljenaKolicina
on MetalnaGalanterija.FAKTURA
insteadofinsert,update
as
begin
        ifupdate(porucena_kolicina)
        begin
                declare @faktura asnumeric(8)=(select FakturaID from inserted)
                declare @vrstaproizvoda asnumeric=(select VrstaProizvodaID from
MetalnaGalanterija.NALAZI_SE ns
                innerjoin MetalnaGalanterija.FAKTURA f
                on ns.FakturaID=f.FakturaID where ns.FakturaID=@faktura)
                declare @poruceno asnumeric=(select porucena_kolicina from inserted)
                declare @proizvedeno asnumeric=(selectsum(velicina_serije)from MetalnaGalanterija.PROIZVOD
p
                where p.ProizvodID=@vrstaproizvoda)
                if(@poruceno > @proizvedeno)
                begin
                        print'Kupovina u datim kolicinama nije moguca.'
                end
                elseif (selectcount(*)from deleted)!= 0 and @poruceno<@proizvedeno
                begin
                        update MetalnaGalanterija.FAKTURA
                        set porucena_kolicina=@poruceno
                        where FakturaID=@faktura
                        print'Promjenjena je porucena kolicina na postojecoj fakturi.'
                end
                elseif (@poruceno<@proizvedeno)
                begin
                        declare @nacinpl aschar(1)=(select nacin_placanja from inserted)
                        declare @jedinicamj asvarchar(10)=(select jedinica_mjere from inserted)
                        declare @valuta asvarchar(10)=(select valuta from inserted)
                        declare @tip_fakture asnumeric(18)=(select tip_fakture from inserted)
                        declare @uslov asvarchar(50)=(select uslov from inserted)
                        declare @datum_izdavanja asdate=(select datum_izdavanja from inserted)
                        declare @pdv asnumeric(3)=(select pdv from inserted)
                        insertinto MetalnaGalanterija.FAKTURA
                        values ((select FakturaID from
inserted),@nacinpl,@jedinicamj,@valuta,@poruceno,@tip_fakture,@uslov,
                        @datum_izdavanja,@pdv)
                        print'Evidentirana nova kupovina.'

                end
        end
end
```

```
go
```

Triger 1: MetalnaGalanterija.KupljenaKolicina

Rezultat prilikom poziva insert ili update naredbi :

```
Kupovina u datim kolicinama nije moguca.

(1 row affected)
```

Slika 14: Rezultat okidanja trigera i poruka da zahtjevani uslovi nisu ispunjeni

```
(1 row affected)
Promjenjena je porucena kolicina na postojecoj fakturi.

(1 row affected)
```

Slika 15: Rezultat okidanja trigera i poruka da su zahtjevani uslovi ispunjeni

Cilj ovog trigera je implementacija poslovnog pravila koja nije bila moguća preko prethodno pomenutih mehanizama. Suština je da se ne dozvoli unos (ili nove torke ili modifikacija obilježja postojeće torke) ako to teorijski nije moguće. U ovom slučaju, kupac ne može da poruči onu količinu neke vrste proizvoda koja nije fizički dostupna, odnosno nije proizvedena u tolikoj mjeri, da bi se mogla isporučiti kupcu. Stoga je bilo potrebno ograničiti modifikaciju ili unos ovakvih količina u bazu podataka, kako bi se izbjegle nepravilnosti.

Triger se aktivira prilikom poziva insert ili update operacija, nad tabelom faktura, zato što se podatak o poručenoj količini evidentira u ovoj tabeli.

Da bismo došli do podatka kolika je poručena količina koju korisnik želi da evidentira u tabeli FAKTURA, koristimo tabelu inserted, koja čuva vrijednosti obilježja torke koju smo pokušali da evidentiramo u trenutku pokretanja trigera. Radi provjere zadovoljenja poslovnog pravila, potreban nam je i podatak o tome kolika je količina proizvedenih proizvoda ove vrste. Taj podatak dobijamo pomoću funkcije sum, koja sabira veličine serija iz tabele PROIZVOD, gdje se id vrste proizvoda poklapa sa id-em koji smo dobili na osnovu fakure.

Ako ta poručena količina jeste veća od proizvedene, ispisuje se poruka da takvu kupovinu nije moguće obaviti. Sa druge strane, ako je poručena količina manja od proizvedene, moguće je izvršiti evidentiranje, ali je prvo potrebna provjera da li je u pitanju modifikacija obilježja postojeće torke, ili je u pitanju unos nove torke. Ovo se provjerava takođe preko privremene tabele deleted koja postoji u toku izvršavanja trigera. Ako je ova tabela prazna, znači da nemamo neko staro stanje, da bi bio u pitanju update, već samo neko novo, i tada radimo insert nove torke. U suprotnom, ako u ovoj tabeli postoje neki podaci, radimo update postojeće torke.

Drugi triger

2.Napisati triger koji implementira poslovno pravilo da kupac ne može da isplati iznos koji je dužan za kupljene proizvode u broju rata većem nego što je to moguće (maksimum rata je 10), i to jedino ako je za način plaćanja računa evidentirano plaćanje na rate, a ako je gotovinski, onemogućiti isplatu na rate.

```
ifobject_id('MetalnaGalanterija.trigerPlacanje','TR')isnotnull
droptrigger MetalnaGalanterija.trigerPlacanje
go

createtrigger MetalnaGalanterija.trigerPlacanje
on MetalnaGalanterija.PLACANJE
afterupdate
as
begin
ifupdate(VrstaPlacanjaID)orupdate(broj_rata)
begin
        declare @staravp asnumeric=(select VrstaPlacanjaID from deleted)
        declare @novavp asnumeric=(select VrstaPlacanjaID from inserted)

        declare @starerate asnumeric=(select broj_rata from deleted)
        declare @noverate asnumeric=(select broj_rata from inserted)

        declare @placanje asnumeric=(select PlacanjeID from deleted)


        if(@novavp = 1)
        begin
                if(@noverate isnotnull)
                        begin
                                raiserror('U pitanju je gotovinsko placanje, unos broja rata nije
dozvoljen!',16,0)
                                update MetalnaGalanterija.PLACANJE
                                set broj_rata=null,iznos_rate=null,VrstaPlacanjaID=@staravp
                                where PlacanjeID=@placanje
                        end
                else
                        print'Promjena je dozvoljena.'
        end
        elseif (@novavp = 2)
        begin
                if(@starerate isnull)
                begin
                        raiserror ('Kod placanja na rate, mora se navesti broj rata!',16,0)
                        update MetalnaGalanterija.PLACANJE
                        set VrstaPlacanjaID=@staravp,broj_rata=@starerate
```

```
                where PlacanjeID=@placanje
        end
        else
        if (@noverate <=10)
                begin
                        print'Promjena broja rata je dozvoljena.'
                end
        else
                        begin
                                raiserror('Broj rata mora biti manji od 10!',16,0)
                                update MetalnaGalanterija.PLACANJE
                                set broj_rata=@starerate,VrstaPlacanjaID=@staravp
                                where PlacanjeID=@placanje
                                end
        end
end
end
go
```

Triger 2: MetalnaGalanterija.trigerPlacanje

Rezultat prilikom poziva update naredbi :

```
Msg 50000, Level 16, State 0, Procedure trigerPlacanje, Line 34 [Batch Start Line 141]
Kod placanja na rate, mora se navesti broj rata!
```

Slika 16: Rezultat okidanja trigera i generisanje greške u slučaju narušavanja nekog uslova

```
Promjena broja rata je dozvoljena.
```

Slika 17: Rezultat okidanja trigera i poruka da je izvršena promjena dozvoljena

```
Msg 50000, Level 16, State 0, Procedure trigerPlacanje, Line 46 [Batch Start Line 145]
Broj rata mora biti manji od 10!
```

Slika 16: Rezultat okidanja trigera i generisanje greške u slučaju narušavanja nekog uslova

Cilj ovog trigera je implementacija poslovnog pravila koje se odnosi na način plaćanja iznosa koji je zdaužen na određenoj fakturi. Obzirom da postoje 2 načina plaćanja, gotovinsko, i plaćanje na rate, potrebno je u sistemu ograničiti koje kombinacije se mogu evidentirati u samu bazu podataka. U okviru same definicije i kreiranja tabela, obilježja broj_rata i iznos_rate su opciona, odnosno ako je evidentirano plaćanje gotovinom, onda su null, a ako je u pitanju plaćanje na rate, potrebno je unijeti podatak o tome u koliko rata će se plaćanje izvršiti, i u kom iznosu su te rate.

Za implementaciju korišćen je after triger, kog pokreće modifikacija obilježja VrstaPlacanjaID i broj_rata, iz tabele PLACANJE.

Neophodno je prvo provjeriti da li se promjenila vrsta plaćanja, ili broj rata, od kojih i zavisi ovo pravilo. Obzirom da podatke o stanju prije, i posle izmjene možemo izvući iz tabela inserted i deleted koje postoje za vrijeme izvršavanja trigera, tako je i u ovom slučaju. Zatim se ispituje koja vrsta plaćanja je zadata u okviru naredbe update. Ukoliko je id 1, u pitanju je gotovinsko plaćanje, i nije moguće unijeti broj rata u tom slučaju. Ako je korisnik pokušao, javiće mu se greška, i stanje tabele vratiće se na poslednje validno. Ako nije proslijeđena vrijednost obilježja broj_rata, promjena je dozvoljena, obzirom da je plaćanje gotovinsko, i javiće se poruka da je promjena sprovedena. Ako je vrsta plaćanja 2, znači da je u pitanju plaćanje na rate, stoga je potrebno, i u ovom slučaju, provjeriti zadovoljenost određenih uslova. Prvo, ako je plaćanje na rate, obilježje broj_rata je obavezno, i korisnik ne može da promjeni vrstu na 2, a da ne unese broj rata. Javiće mu se adekvatna greška, i stanje tabele se vraća na poslednje validno. Ako je unešen i broj rata, potrebno je provjeriti koliki je taj broj, jer je maksimalan dozvoljen broj rata 10. Ako je proslijeđeni broj manji od 10, promjena je dozvoljena, u suprotnom javlja se greška, i promjena se poništava, dovodeći tabelu u poslednje konzistentno stanje.

ZAKLJUČAK

Kroz ovaj projekat, cilj je bio samostalno proći kroz postupak razvoja i implementacije baze podataka, na koju bi se, uz određene dorade, ili proširenje funkcionalnosti, mogao osloniti jedan realni sistem u svakodnevnom radu. U ovom konkretnom slučaju, taj sistem bi bio preduzeće koje se bavi proizvodnjom metalne galanterije, stoga je naglasak bio na procesu proizvodnje. Podšema, kojom se projekat najvećim dijelom bavi, odnosi se na prodaju i distribuciju proizvedenih proizvoda i usluga u okviru preduzeća.

Kroz rad na ovom projektu, naučila sam koliko je važno poznavati osnovne koncepte projektovanja baze podataka nekog informacionog sistema, i od koje važnosti ona može da bude u radu nekog preduzeća. U današnje vrijeme, skoro je nezamislivo svakodnevno poslovanje, kako većih i srednjih preduzeća, tako i onih manjih, bez oslanjanja na informacione tehnologije. Jedno veliko preduzeće, poput onog koje se ovde pominje, ne bi moglo normalno funkcionisati bez adekvatnog informacionog sistema, obzirom da kroz njega svakoga dana protekne velika količina podataka, koje je potrebno na nekom mjestu trajno evidentirati, održavati, i na taj način koristiti za unaprijeđenje poslovanja.

Takođe, naučila sam koliko je važno voditi računa o konzistentnosti, sigurnosti i validnosti podataka u okviru baze podataka, što ne bi bilo moguće, čak ni u realnom poslovanju, bez upotrebe mehnizama koji su korišćeni i kroz ovaj projekat, a koji je značajno manjeg obima u odnosu na komercijalne softverske proizvode, ali ipak dovoljan da se razumije potreba za postojanjem ovakvog sistema, svih mehanizama, pravila i ograničenja.

LITERATURA

- Za potrebe pojašnjenja osnovnih pojmova u okviru projekta, korišćeni su slajdovi profesorice Sonje Ristić, obrađivani na predavanjima, u okviru predmeta *Projektovanje baza podataka* i *Sistemi baza podataka*.