BAZE PODATAKA

Predavanje broj: 09

Nastavne teme:

- ✓POGLEDI
- ✓FUNKCIJE

POGLEDI

Koncept baze podataka daje mogućnost da više različitih aplikacija, pa samim tim i više korisnika, pristupa bazi podatka. Pojedinačni korisnik ne mora da zna sve detalje baze podataka u celini: njega mogu zanimati samo neke tabele i samo neki atributi tih tabela.

Tvorac relacionih baza podataka Codd definisao je više tipova relacija, a među njima i tzv. **pogled (view).**

Osnovna razlika između bazne tabele i pogleda je u tome što bazna tabela postoji na memorijskom medijumu računara (disku) dok je pogled, fiktivna virtuelna tabela koja ne postoji u memoriji, ali se može formirati uvek kada nam zatreba, tj. kada je referenciramo. Sa korisničke tačke, nema razlike u pretraživanju ovakve virtuelne tabele i bazne tabele.

POGLEDI

Pogled je virtuelna imenovana tabela koja se kreira naredbom CREATE (SELECT), ali se u bazi podataka memoriše na specifičan način. Definicija pogleda (naziv pogleda, nazive kolona i upit) čuva se u bazi podataka u prevedenom obliku, što obezbeđuje veliku brzinu rada sa pogledima.

Kada korisniku zatreba „pogled“ na bazu podataka, on ga poziva i kreira virtuelnu tabelu. Pri izvršavanju upita nad pogledom, DBMS kombinuje taj upit sa upitom koji definiše pogled i pravi se novi upit koji se izvršava nad baznim tabelama koje čuvaju podatke (fizičke tabele na disku).

Pogled se najčešće kreira nad više tabela. Rad sa pogledima pruža podršku očuvanju integriteta podataka u bazi. Neki DBMS (npr. MS Access) nemaju mogućnost rada sa pogledima.

POGLEDI

Pogledi obezbeđuju:

- Uprošćavanje upita
- Mehanizam za kontrolu pristupa podacima
  - korisnik vidi samo neke podatke
- Bolje performanse
  - pogled se čuva u prevedenom obliku što povećava brzinu rada
- Nezavisnost podataka
  - menjaju se definicije pogleda, a ne aplikativni programi koji koriste podatke iz baze podataka preko pogleda.

MySQL omogućava više SELECT naredbi u pogledu koje su povezane klauzulama UNION ili UNION ALL.

Zašto koristimo poglede?

Sigurnost

- Veoma često može postojati situacija u kojoj želimo da određenim korisnicima baze podataka onemogućimo pun pristup određenoj tabeli ili tabelama.

Zašto koristimo poglede?

Sigurnost

- Tipičan primer ove situacije su tabele za smeštanje podataka o zaposlenima. Mi ćemo svakako želeti da omogućimo svim zaposlenima pristup njihovim ličnim podacima, kao što su ime, prezime, broj telefona ili adresa. Ipak, zaposlenima nije dobro omogućiti pristup podacima o visini zarada i sličnim osetljivim podacima.

Zašto koristimo poglede?

Komfor

- Već smo videli da je neophodno često vršiti spajanje podataka više tabela. Kako bi se olakšao rad korisnicima baze podataka ili programerima koji će pisati aplikacije koje će komunicirati sa bazom, možemo olakšati stvari i kreirati pogled.

POGLEDI

- Imena kolona u pogledu ne moraju biti ista kao imena kolona u tabelama iz kojih se pogled izvodi.
- Pogled se može izbaciti iz baze podataka naredbom DROP VIEW ime_pogleda.
- Ova komanda uklanja pogled i iz relacione šeme i iz rečnika podataka baze podataka.
- Brisanje pogleda nema nikakvog uticaja na tabele i podatke u njima.

POGLEDI

- Poglede treba povremeno izbacivati iz šeme i odmah ih kreirati. Razlog je vrlo prost. Pri kreiranju pogleda vrši se optimizacija njegovog izvršavanja.

- Naravno, optimalno izvršavanje zavisi od trenutnih vrednosti u tabelama.

- Nakon nekog vremena, tabele koje se često ažuriraju u toj meri promene stanje da pogled više nema dovoljnu brzinu.

- Zbog toga se pogled mora izbaciti iz šeme baze podataka (DROP) i odmah iznova napraviti (CREATE). Prilikom ponovnog kreiranja novodobijeni binarni kod će biti optimalan za novo stanje podataka u bazi.

POGLEDI

- Prilikom izmene vrednosti u tabelama koje su korišćene u upitu od kojeg je napravljen pogled, podaci dobijeni upitom nad tim pogledom će uvek biti ažurni.

- Pogledi ne sadrže podatke koji su bili dostupni u tabelama u onom trenutku kada je pogled napravljen, već sadrže definiciju upita koji se pokreće svaki put kada se zahteva dopremanje podataka pogledom.

- Poglede treba koristiti oprezno, jer mogu izazvati loše performanse upita.

POGLEDI

Naredba za kreiranje pogleda CREATE VIEW ima oblik:

CREATE VIEW <ime_pogleda> [(<lista_atributa>)] AS
<SELECT_naredba>

Naredba CREATE VIEW kreira virtuelnu tabelu sa specificiranom listom atributa.

Ako lista atributa nije specificirana, nazivi i tipovi atributa preuzimaju se iz baznih tabela navedenih u SELECT naredbi.

POGLEDI

Naredba za brisanje pogleda DROP VIEW iz baze podataka ima oblik:

DROP VIEW <ime_pogleda>

Naredba za modifikovanje pogleda ALTER VIEW ima oblik:

ALTER VIEW <ime_pogleda> [(<lista_atributa>)] **AS**
<SELECT_naredba>

Naredba ALTER VIEW kreira novu virtuelnu tabelu sa specificiranom listom atributa. Podrazumeva se da je specificirani pogled već kreiran naredbom CREATE VIEW .

Sve uspostavljenje dozvole ostaju na snazi. Sukcesivno izvršavanje naredbi DROP VIEW i CREATE VIEW ima isto dejstvo kao naredba ALTER VIEW, ali zahteva ponovno uspostavljanje prava pristupa.

POGLEDI

- Pogledi su usko vezani sa bezbednošću i pravima korisnika.

- Prilikom kreiranja pogleda, kreator pogleda mora imati prava nad tabelama nad kojima se pogled kreira.

- Sa druge strane, korisnik ne mora imati SELECT prava nad tabelama, sve dok ima prava nad pogledom.

POGLEDI

Kreirati korisnika John i dodeliti mu prava na pogled uzmifilmove

```
CREATE USER 'John'@'localhost' identified by '123'

grant select on uzmifilmove to John
```

Ulogovati se kao John, a zatim isprobati sledeće upite:

```
select * from uzmifilmove
```

Može

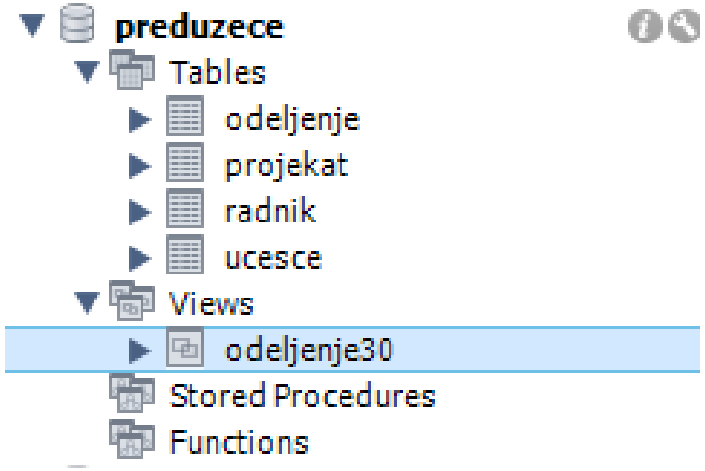
```
select * from film
update uzmifilmove set title = 'ABC' where film_id = 1
```

Ne može

POGLEDI

Kreirati pogled ODELJENJE30 koji sadrži sve podatke o radnicima zaposlenim u odeljenju 30.

```
CREATE VIEW ODELJENJE30 AS
SELECT *FROM RADNIK
WHERE id_odeljenja=30;
```




```
SELECT *FROM ODELJENJE30;
```

Id_radnika	Ime	Prezime	Posao	Kvalif	Rukovodilac	Dat_zap	Premija	Plata	Id_odeljenja
5786	Pavle	Šotra	upravnik	VSS	6789	1983-05-22 00:00:00	NULL	2800	30
5898	Andrija	Ristić	nabavljač	KV	5786	1980-01-20 00:00:00	1200	1100	30
5953	Jovan	Perić	nabavljač	KV	5786	1979-01-12 00:00:00	0	1100	30
6234	Marko	Nastić	analitičar	VSS	5867	1990-12-17 00:00:00	3000	1300	30

POGLEDI

Kreirati pogled **ODELJENJE20** koje sadrži podatke o odeljenju kao i ime, prezime, posao, kvalifikaciju i platu zaposlenih u odeljenju 20.

RADNIK <**#id_radnika**, ime, prezime, posao, kvalif, **rukovodilac$**, dat_zap, premija, plata,**Id_odeljenja$**>

ODELJENJE <**#id_odeljenja**, ime_od, mesto, **sef_odeljenja$**>

POGLEDI

```
CREATE VIEW ODELJENJE20 AS
SELECT O.ime_od, R.ime, R.prezime, R.posao, R.kvalif, R.plata
FROM RADNIK R, ODELJENJE O
WHERE O.id_odeljenja=R.id_odeljenja AND O.id_odeljenja=20
```

```
SELECT *FROM ODELJENJE20;
```

ime_od	ime	prezime	posao	kvalif	plata
Plan	Petar	Vasić	vozač	KV	1300
Plan	Božidar	Ristić	upravnik	VSS	2200
Plan	Slobodan	Petrović	vozač	KV	900
Plan	Mitar	Vuković	savetnik	VSS	2600
Plan	Ivan	Buha	analitičar	VSS	1600

U ovome primeru nisu uvedeni novi atributi, iako ih pogled može imati.
Imena atributa u pogledu i osnovnim tabelama iz kojih je izveden su ostala ista.

POGLEDI

Svaki put kada neko iz odeljenja 20 hoće da obrađuje podatke o zaposlenima u svom odeljenju on jednostavno koristi ovaj pogled.

Primer: Prikaži ime, posao i kvalifikaciju za zaposlene sa visokom stručnom spremom u odeljenju 20:

```
SELECT O20.ime, O20.posao, O20.kvalif
FROM ODELJENJE20 O20
WHERE O20.kvalif='VSS'
```

ime	posao	kvalif
Božidar	upravnik	VSS
Mitar	savetnik	VSS
Ivan	analitičar	VSS

POGLEDI

Kreirati pogled IZVOZ koji sadrži podatke o identifikacionom broju radnika, imenu, prezimenu, broju sati angažovanja i funkciji koju imaju na projektu, samo za radnike koji su angažovani na projektu izvoz.

RADNIK <**#id_radnika**, ime, prezime, posao, kvalif, **rukovodilac$**, dat_zap, premija, plata,**Id_odeljenja$**>

UCESCE<**#id_radnika**, **#id_projekta**, br_sati, funkcija>

PROJEKAT **#id_projekta**, ime_proj, sredstva, rok>

POGLEDI

```
CREATE VIEW IZVOZ AS
SELECT R.Id_radnika, R.ime, R.prezime, U.br_sati, U.funkcija
FROM RADNIK R, UCESCE U
WHERE R.Id_radnika = U.Id_radnika AND
U.Id_projekta= (SELECT P.Id_projekta
                FROM PROJEKAT P
                WHERE P.ime_proj='izvoz');
```

```
SELECT * FROM IZVOZ ;
```

Id_radnika	ime	prezime	br_sati	funkcija
5696	Mirjana	Dimić	2000	ŠEF
5780	Božidar	Ristić	2000	ORGANIZATOR
5867	Svetlana	Grubač	2000	KONSULTANT
5898	Andrija	Ristić	2000	IZVRŠILAC
5932	Mitar	Vuković	1000	ORGANIZATOR
6234	Marko	Nastić	1200	IZVRŠILAC
6789	Janko	Simić	2000	IZVRŠILAC

Ažuriranje baze podataka upotrebom pogleda

Pogledi, osim mogućnosti prikazivanja aktuelnog sadržaja po zadatoj formi, imaju i mogućnost ažuriranja izvora nad kojima su formirani, ali pri tom moraju biti ispoštovana neka pravila.

Ažuriranje baze podataka upotrebom pogleda

Pravila koja se odnose na kreiranje SELECT upita prilikom definisanja pogleda za ažuriranje baze:

- SELECT upit ne sme sadržati ključne reči GROUP BY, DISTINCT, LIMIT, UNION, HAVING
- Pogled mora sadržati sve kolone određene tabele koje su definisane kao primarni ili strani ključevi kao i sve ostale NOT NULL kolone tabele.
- U SELECT listi ne mogu da budu navedene agregatne funkcije i izvedene kolone.
- Pogledi koji grupišu podatke iz više tabela ne mogu se koristiti za izmenu podataka u baznim tabelama.

Ažuriranje baze podataka upotrebom pogleda

Kada se ažurira bazna tabela preko pogleda:

- sve kolone u tabeli koje su definisane kao NOT NULL moraju da prime vrednost kada dobijaju novu ili menjaju vrednost.

- ovo može da se čini eksplicitno:
  - direktnim upisivanjem
  - ažuriranjem NOT NULL vrednosti u koloni
  - oslanjanjem na podrazumevanu vrednost.

Ažuriranje baze podataka upotrebom pogleda

- Pogledi ne skidaju postavljenja ograničenja u osnovnoj tabeli.
- Vrednosti koje se upisuju ili ažuriraju u osnovnoj tabeli moraju da ispunjavaju sva ograničenja postavljena nad njima sa jedinstvenim indeksima, primarnim ključevima, CHECK ograničenjima itd.
- MySQL ne dozvoljava da se unutar pogleda kreira podupit ako se želi vršiti ažuriranje baze sa tim pogledom.

Ažuriranje baze podataka upotrebom pogleda

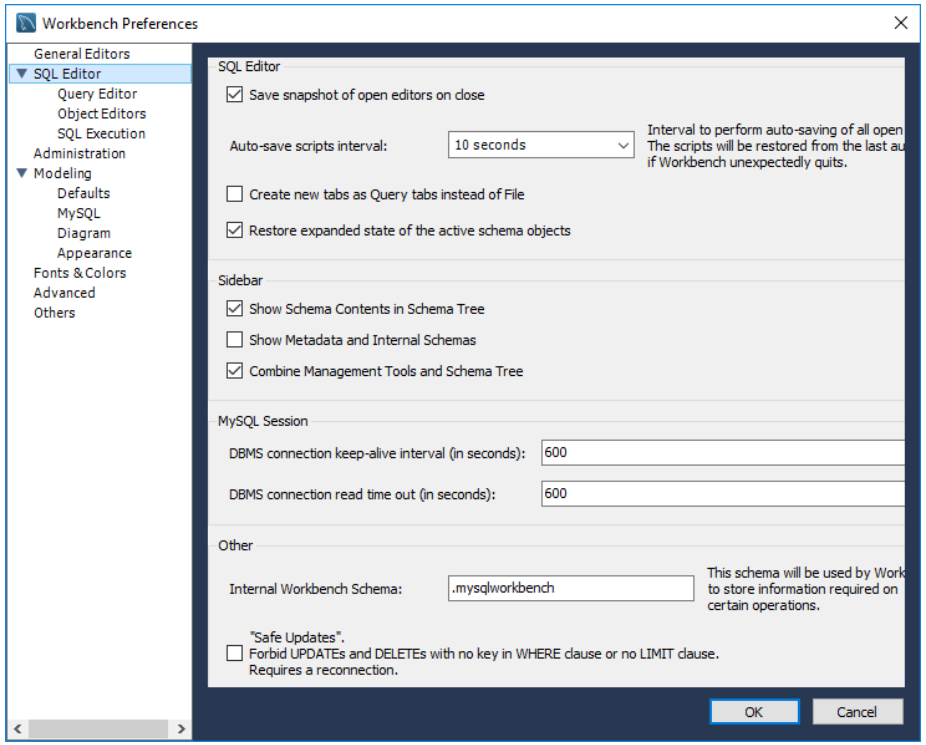
Primer: Kreirati pogled RadnikZarada koji prikazuje ime, platu i premiju zaposlenih. Radnicima koji imaju platu veću od 1900 dodeliti premiju od 1850. Promenu izvršiti preko pogleda.

```
create view RadnikPlata as
select ime, plata, premija
from radnik;

select * from RadnikPlata;

update RadnikPlata
set premija=1850
where plata>1900;
```

Napomena za update u MySQL-u: Isključiti opciju Safe Updates (Edit>Preferances>SQL Editor)



Ažuriranje baze podataka upotrebom pogleda

Da li je moguće dodavanje novog zaposlenog preko pogleda RadnikPlata?

insert into RadnikPlata values (’Lazar', 10000,100)

Ažuriranje baze podataka upotrebom pogleda

```
alter view RadnikPlata as
select id_radnika, prezime, ime, plata, premija
from radnik;
```

```
insert into RadnikPlata
values ( 5000, 'Lazić','Lazar', 10000,100)
```

Ažuriranje baze podataka upotrebom pogleda

```
create view RadnikOdeljenjePlata as
select id_radnika, prezime, ime, plata, ime_od
from radnik, odeljenje
where radnik.id_odeljenja=odeljenje.Id_odeljenja
and plata<2000
```

```
delete from
RadnikOdeljenjePlata
where id_radnika=5900;
```

delete from RadnikOdeljenjePlata where id_radnika=5900
Error Code: 1395. Can not delete from join view 'preduzece.radnikodeljenjeplata'

Funkcije

- Funkcija je potprogram koji na osnovu liste ulaznih parametara izračunava izlazni parameter.
- U opštem slučaju možemo kreirati funkciju koja na izlazu daje rezultat samo na osnovu ulaznih parametara.
- Možemo da kreiramo i funkciju koja na osnovu ulaznih parametara i pristupom nad određenim tabelama rezultat dobija kombinacijom ulaznih parametara i sadržaja tabela.

Funkcije

```
CREATE [DEFINER = { db_korisnik | CURRENT_USER }]
FUNCTION ime_funkcije ([ime_parametra tip_parametra[,...]])
RETURNS tip_parametra_koji_se_vraća
[characteristic ...] telo_funkcije
characteristic:
COMMENT 'string'
| LANGUAGE SQL
| [NOT] DETERMINISTIC
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }
```

Funkcije

db_korisnik	Korisničko ime korisnika koji kreira funkciju, određuje korisnička prava i proverava privilegije korisnika u momentu kada se izvršava funkcija
ime_parametra tip_parametra	Definiše parametar koji se prosleđuje funkciji (npr. plata decimal(17,2))
tip_promenljive_koja_se_vraća	Definiše kog je tipa vrednost promenljive koja se vraća na izlazu funkcije
COMMENT 'string'	Predstavlja komentar koji ne utiče na izvršenje funkcije, ali autoru daje informaciju šta radi funkcija (npr. Funkcija služi da izračuna mesečnu platu zaposlenih)
LANGUAGE	Pokazuje koji programski jezici su dozvoljeni unutar tela funkcije - dozvoljena vrednost je SQL
DETERMINISTIC	pokazuje da je funkcija deterministička tj. da za skup istih ulaznih parametara uvek daje isti rezultat. Suprotno je NOT DETERMINISTIC što se može izostaviti u sintaksi.
CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA	Ključne reči koje definišu "ponašanje" naredbi unutar tela funkcije - CONTAINS SQL pokazuje da funkcija ne sadrži naredbe koje čitaju ili upisuju podatke (npr. SET @x = 1), NO SQL pokazuje da funkcija ne sadrži SQL naredbe, READS SQL DATA pokazuje da funkcija sadrži naredbe koje čitaju podatke (npr SELECT naredba), ali ne i naredbe koje upisuju podatke i MODIFIES SQL DATA pokazuje da funkcija sadrži naredbe koje mogu menjati neke podatke (npr. INSERT ili DELETE)
SQL SECURITY	Ova karakteristika može da sadrži ključne reči DEFINER (onaj koji je definisao funkciju) ili INVOKER (onaj koji je pokrenuo funkciju) kako bi se naznačilo da li će se funkcija izvršiti pod privilegijama korisničkog naloga iza reči DEFINER ili naloga korisnika koji ju je pokrenuo (INVOKER)
telo_funkcije	Predstavlja jednu ili više naredbi u telu funkcije koje se izvršavaju kada se pokrene funkcija

Funkcije

U MySql-u postoje dve vrste funkcija:

- Ugrađene
- Korisnički definisane

Ugrađene funkcije su sve one funkcije koje podrazumeva standardna forma MySQL servera, dok su korisnički definisane one koje su izgrađene naknadno, od strane korisnika.

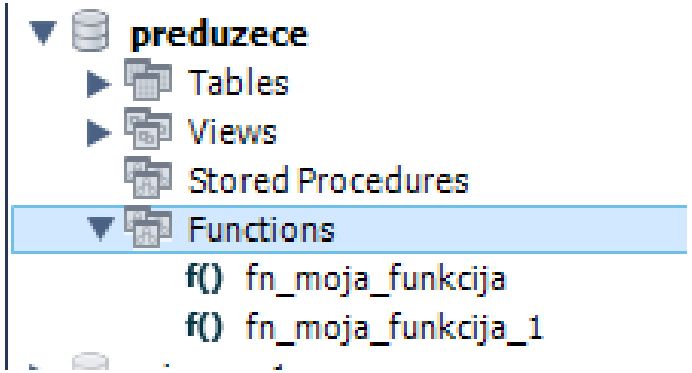
Sve funkcije se generalno dele na skalarne i agregatne.

Kreiranje funkcije

Prilikom kreiranja funkcija neophodno je definisati njen izlazni tip - **naredba RETURNS**.

Vraćanje vrednosti iz funkcije (prikaz izlaza funkcije) – **naredba RETURN**

```
create function fn_moja_funkcija()
returns varchar(20)
return 'Zdravo svete';
```



Kreiranje funkcije

Za funkcije duže od jednog reda neophodno je u MySQL-u koristiti delimiter.

```
DELIMITER //
CREATE FUNCTION fn_moja_funkcija_1()
RETURNS varchar(20)
BEGIN
DECLARE x int;
SET x=10;
RETURN 'POZDRAV';
END //
DELIMITER ;
```

Pozivanje funkcije

Funkcija se poziva pomoću SELECT klauzule što omogućava da bude umetnuta u sam upit.
Na raspolaganju je veliki broj mogućnosti za intervenciju na podacima u trenutku kreiranja izlaza.

Poziv funkcije:

```
select fn_moja_funkcija_1()
```

Parametrizacija funkcije

Parametrizacija funkcije vrši se navođenjem parametara i njihovih tipova. Ako funkcija ima ulazne parametre, u pozivu funkcije je neophodno navesti vrednosti koje se dodeljuju ulaznim parametrima.

```
DELIMITER //
create function fn_moja_funkcija_2(p1 int, p2 int)
returns int
BEGIN
declare p3 int;
set p3=p1+p2;
return p3;
end //
DELIMITER ;
```

select fn_moja_funkcija_2(3,4)

Kako možemo izmeniti MySQL korisničku funkciju

Izmena funkcije moguća samo ako korisnik ima ALTER ROUTINE privilegije i uz primenu ALTER FUNCTION upit

```
ALTER FUNCTION function_name [characteristic ...]
characteristic:
  { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES
SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }
| COMMENT 'string'
```

Klauzula ALTER FUNCTION može da menja samo karakteristike navedene u sintaksi ALTER FUNCTION upit. U klauzuli ALTER FUNCTION moguće je navesti više promena.

Ovom klauzulom nije moguće menjati parametre ili telo funkcije!!!!

U slučaju promene parametara ili tela funkcije:
DROP FUNCTION ime funkcije pa ponovo CREATE FUNCTION ime funkcije

Primer funkcije

Kreirati funkciju fun_Radnik_Staz koja vraća godine staza radnika čije se ime i prezime zadaju.

Primer funkcije

```
DELIMITER $$
CREATE FUNCTION fun_Radnik_Staz (imeU char(30), prezimeU char(30))
RETURNS int
BEGIN
/*deklarisanje izlazne promenljive staz koja predstavlja izlaz funkcije*/
DECLARE  staz int;
/*dodela vrednosti promenljivoj staz*/
SET staz= (select year(curdate())-year(dat_zap)
                from radnik where ime=imeU and prezime=prezimeU);
RETURN staz;
END$$
DELIMITER ;
```

Primer funkcije

```
DELIMITER $$
create function fun_Radnik_Staz (imeU char(30), prezimeU char(30))
returns int
begin
declare  staz int;
/*drugi način dodele vrednosti promenljivoj staz – klauzula
select..into*/
select year(curdate())-year(dat_zap)  into staz
from radnik
where ime=imeU and prezime=prezimeU;
return staz;
END$$
DELIMITER ;
```

Primer funkcije

Poziv funkcije

```
select fun_Radnik_Staz('Mirjana','Dimić')
```

Primer funkcije

Upotrebom funkcije prikazati radnike koji imaju veća primanja od maksimalne plate.

Primer funkcije

```
delimiter //
create function fn_max_plata()
returns float
BEGIN
declare maxPlata float;
select max(plata) into maxPlata
from radnik;
return maxPlata;
end //
DELIMITER ;

select fn_max_plata();

select ime, prezime
from radnik
where plata+ifnull(premija,0)>fn_max_plata()
```

Primer funkcije

Prikazati prezime, ime radniku samo za radnike koji imaju najveću platu u odeljenju Direkcija.

Primer funkcije

```
DELIMITER $$
create function fun_PreduzeceMaxPlata1(imeOdeljenja char(20))
returns float
begin
declare MaxPlata float;
select max(plata) into MaxPlatafrom radnik inner join odeljenje
on radnik.id_odeljenja=odeljenje.id_odeljenja where
odeljenje.ime_od=imeOdeljenja;
return MaxPlata;
END $$
DELIMITER ;

select prezime, ime, fun_PreduzeceMaxPlata1('Direkcija')
from radnik
where Id_odeljenja is not null
```

Primer funkcije

Kreirati funkciju fnIzracunajMaxMesecnuPlatu koja na osnovu ulaznog parametara idbr_in u tabeli RADNIK nalazi odgovarajuće vrednosti iz kolona plata i premija i sabira vrednost plate uvećanu za 14% sa vrednošću premije za zaposlenog sa prosleđenom šifrom radnika.

Primer funkcije

```
DELIMITER $$
CREATE FUNCTION fnIzracunajMaxMesecnuPlatu(idbr_in int)
RETURNS decimal(17,2)
BEGIN
DECLARE MesecniIznos decimal(17,2);
SELECT plata * 1.14 + IFNULL(premija, 0.00) into
MesecniIznos
FROM radnik
WHERE id_radnika = idbr_in;
RETURN ifnull(MesecniIznos, 0.00);
END$$
DELIMITER ;
```

Primer funkcije

Poziv funkcije

```
select fnIzracunajMaxMesecnuPlatu(5662)
```

ZADACI

Kreirati pogled View_Staz koji prikazuje ime, prezime i posao radnika koji imaju više godina staža od Slobodana Petrovića.

```
create view View_Staz as
select ime, prezime, posao, year(curdate())-year(dat_zap) 'Staz’
from radnik
where year(curdate())-year(dat_zap)>
        (select year(curdate())-year(dat_zap)
          from radnik where
          ime='Slobodan' and prezime= 'Petrović' );
```

Izmeniti pogled View_Staz koji prikazuje ime, prezime i posao radnika koji imaju više godina staža od Janka Mančića upotrebom funkcije.

```
/*Ako je kreirana funkcija – ne treba ponovo*/
DELIMITER $$
create function fun_Radnik_Staz (imeU char(30), prezimeU char(30))
returns int
begin
declare  staz int;
select year(curdate())-year(dat_zap) into staz
from radnik
where ime=imeU and prezime=prezimeU;
return staz;
END $$
DELIMITER ;
```

```
select fun_Radnik_Staz_2('Janko','Mančić') as Staz;
```

Staz
26

```
alter view View_Staz as
select ime, prezime, posao, year(curdate())-year(dat_zap) 'Staz’
from radnik
where year(curdate())-year(dat_zap)>
fun_Radnik_Staz_2('Janko','Mančić');
```

select * from View_Staz

ime	prezime	posao	Staz
Petar	Vasić	vozač	41
Aleksandar	Marić	električar	29
Vanja	Kondić	prodavac	28
Jovan	Perić	električar	39
Mirjana	Dimić	čistač	28
Božidar	Ristić	upravnik	35
Pavle	Šotra	upravnik	36
Miloš	Marković	direktor	38
Svetlana	Grubač	savetnik	49
Tomislav	Bogovac	električar	48
Andrija	Ristić	nabavljač	39
Jovan	Perić	nabavljač	40
Marko	Nastić	analitičar	29

U cilju demonstriranja pogleda za složenije potrebe, možemo kreirati upit koji nam pokazuje obračunate plate iz tabela PLATE i PLATE_STAVKE grupisane po svakom mesecu i upoređuje sa maksimalnom mesečnom zaradom po svakom zaposlenom.

Kreirati pogled vwMesecniIzvestajPlata koji po svakom zaposlenom prikazuje primljenu mesečnu zaradu, maksimalnu zaradu koju zaposleni može da primi, procenat primljene zarade u odnosu na maksimalnu mesečnu.

U pogledu koristiti funkciju fnIzracunajMaxMesecnuPlatu za izračunavanje maksimalne mesečne zarade.

```
CREATE VIEW vwMesecniIzvestajPlata AS

SELECT pl.mesec, pl.godina, ps.idbr, concat(r.ime, ' ', r.prezime) as zaposleni,
sum(ps.iznos) as IznosPlata, fnIzracunajMaxMesecnuPlatu(ps.idbr) as MaxMesPlata,
sum(ps.iznos)/fnIzracunajMaxMesecnuPlatu(ps.idbr) * 100 as procenat

FROM plate pl, plate_stavke ps, radnik r

WHERE pl.autoid = ps.autoid AND r.idbr = ps.idbr

GROUP BY pl.mesec, pl.godina, ps.idbr, concat(r.ime, ' ', r.prezime),
fnIzracunajMaxMesecnuPlatu(ps.idbr);

ORDER BY ps.idbr, pl.godina, pl.mesec;));
```

Kreirati pogled View_Plasman koji prikazuje ime, prezime i posao najplaćenijeg radnika sa Dorćola koji radi na projektu plasman.

```
create view View_Projekti as
select ime,prezime,posao
from radnik
where (plata+ifnull(premija,0)) =
   (select max(plata+ifnull(premija,0))
    FROM radnik
    WHERE id_odeljenja= (SELECT id_odeljenja FROM odeljenje
                                    WHERE mesto='Novi Beograd'
                                    and id_radnika in  (SELECT id_radnika FROM ucesce
                                                                    WHERE id_projekta =
                                                                        (SELECT id_projekta
                                                                         FROM projekat
                                                                         where  Ime_proj='Plasman'))));
```