RAČUNARSKA GIMNAZIJA

MATURSKI RAD

iz modela i baza podataka

BAZA PODATAKA ZA FAKULTET

Učenik:
Staša Đorđević IV1

Mentor:
Duša Vuković

Beograd, maj 2021

Sadržaj

Uvod

Relacione baze podataka

Pojam **baza podataka** se koristi u računarstvu da se opiše sistem koji služi za efikasno i sigurno čuvanje i obradu podataka.

Relacione baze podataka su skup međusobno povezanih podataka. Podaci su skladišteni uz pomoć tabela jednostavnih struktura.

Entiteti su objekti koji predstavljaju jednu zasebnu celinu o kojoj čuvamo podatke. Oni predstavljaju skup instanca koje su od određenog značaja za neko preduzeće. Svakom entitetu odgovara jedna tabela u bazi podataka.

Svaki entitet ima svoje **instance** sa kojima se susrećemo u poslovanju. Instanca je konkretan primerak i jednoj instanci odgovara jedan red u tabeli.

Svaki entitet opisan je **atributima**. Njegova vrednost za konkretnu instancu entiteta su podaci koji se čuvaju u bazi podataka.

Primarni jedinstveni identifikator je atribut koji svaka instanca mora imati i jedinstven je. Obeležava se sa tarabicom (#). Može biti prirodan i veštački. Prirodni su oni koji postoje i van baze podataka, dok veštačke uvodimo samo zbog potreba baze podataka. Takođe se mogu podeliti na proste i složene. Kada se sastoji od jednog atributa onda je prost, a kada se sastoji od kombinacije nekoliko – složen.

Osim primarnog jedinstvenog identifikatora, ostali atributi mogu biti obavezni ili opcioni. Obavezne obeležavamo sa zvezdicom (*), a opcione kružićem (○).

Entiteti se povezuju **vezama**. Svaka veza spaja dva različita entiteta, osim u slučaju rekurzivne veze, kada se entitet spaja sam sa sobom. Svaka veza ima opcionalnost i kardinalnost.

Opcionalnost nam govori da li neka instanca mora da bude u vezi sa instancom drugog entiteta.

Kardinalnost nam govori da li je jedna instanca jednog entiteta u vezi sa jednim ili sa više instanci drugog entiteta te veze. Zbog toga razlikujemo tri tipa veze:

1. jedan prema jedan (**1:1**)
2. jedan prema više (**1:M**)

3. više prema više (**M:M**).

Kada imamo vezu više prema više, potrebno je da uvedemo novi entitet koji će nju da predstavlja. Taj entitet ima primarne ključeve susednih entiteta, koji su njemu istovremenno primarni i strani ključevi.

Modelovanje

Opis poslovanja

Svakom fakultetu je potrebna baza podataka gde će se čuvati podaci o svim zaposlenima, studentima, predmetima i ispitima. Pomoću kreirane baze podataka lakše će se pretraživati ovi podaci. Kada osoba upiše fakultet njeni podaci se unose u bazu. Kada se neko zaposli na fakultetu takođe. Za svaki predmet potrebno je uneti ko je sve od nastavnog osoblja zadužen za njega. Kada se održi neki ispit, u bazu se upisuju rezultati i podaci o njemu.

Opis modela

Pravimo bazu podataka za fakultet. Prvo kreiramo nezavisne entitete koji ne sadrže strane ključeve, a to su STUDENT, POZICIJA i GODINA. Studenti imaju broj indeksa koji im se dodeljuje pri upisu i sastoji se od trocifrenog broja, slova koje je početno slovo smera na kojem je student, i poslednje dve cifre koje označavaju dvocifreni broj godine koje se student upisao na fakultet. Broj indeksa je jedinstven za svakog studenta i on je primarni ključ, tako da ispred njega stoji #. Entitet STUDENTI takođe sadrži godinu upisa, ime, prezime, mejl adresu, adresu stanovanja, jmbg i broj telefona. Svi ovi atributi su obavezni i pored njih stoji *. Entitet POZICIJA označava na kojoj je radnoj poziciji zaposleni, i ona sadrži id_pozicije koji je primarni ključ, pa ispred piše #. Sadrži naziv pozicije koji je obavezan i opis koji je opcioni i zato se ispred nalazi ○. Entitet GODINA sadrži redni_broj koji je primarni ključ, ESP_bodovi koji označavaju broj bodova koliko nosi ta godina i opis koji je opcioni.

Entitet NASTAVNO_OSOBLJE je povezan sa entitetom POZICIJA. Jedan zaposleni može da bude na jednoj poziciji, ali više zaposlenih može da bude na istoj poziciji. Zato smo stavili vezu jedan prema više (1:M) između ova dva entiteta. NASTAVNO_OSOBLJE pored stranog ključa sadrži atribute id_zaposlenog koji je primarni ključ, ime, prezime, jmbg, adresa, datum_rođenja, mejl_adresa, datum_zaposlenja, plata, broj_telefona koji su obavezni i diplomu koja je opcioni atribut.

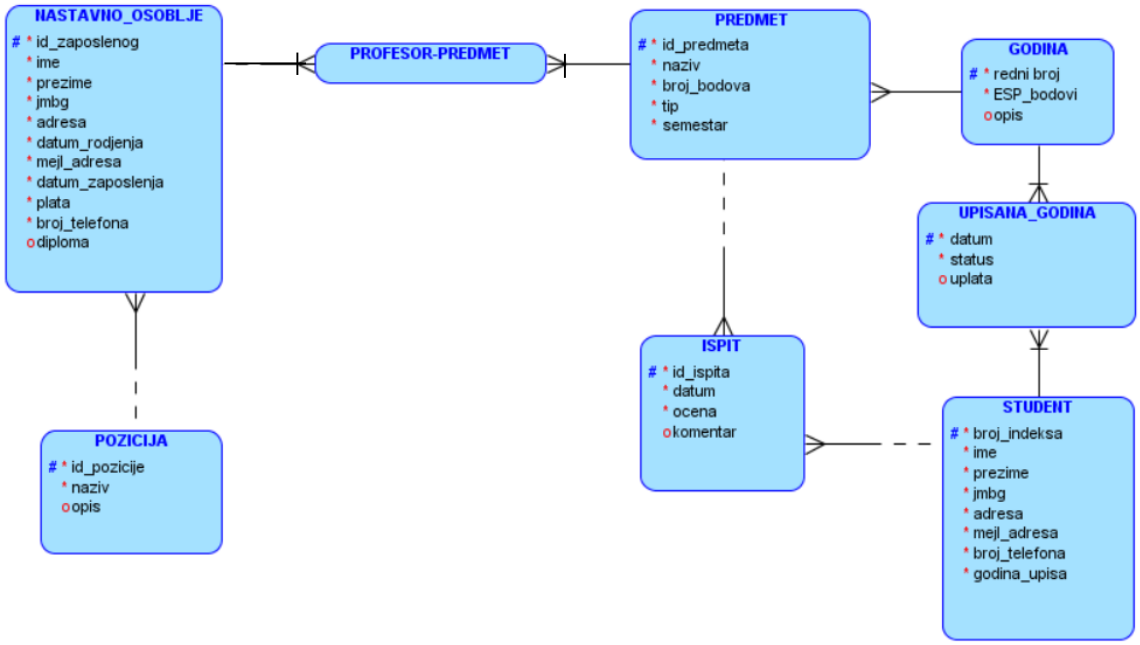
Entitet PREDMET je povezan sa entitetom GODINA vezom jedan prema više – u jednoj godini ima više predmeta, ali jedan predmet je samo u jednoj godini. Sadži primarni ključ id_predmeta i obavezne atribute naziv, broj_bodova, tip i semestar.

Entiteti PREDMET i NASTAVNO_OSOBLJE povezani su vezom više prema više (M:M), zato sto jedan profesor može da predaje više predmeta, ali i jedan predmet može da predaje više

profesora. Zbog toga uvodimo novi entitet PROFESOR-PREDMET koji sadrži primarne ključeve entiteta PREDMET i NASTAVNO_OSOBLJE.

Entitet ISPIT je povezan sa entitetom PREDMET. Sadrži primarni ključ id_ispita, atribute datum i ocena koji su obavezni i komentar koji je opcioni.

Prikaz dijagrama



Mapiranje

Mapiranje predstavlja pripremu za kreiranje baze podataka. Detaljno se opisuju sve tabele koje će se uneti u bazu. To podrazumeva tabelarni prikaz svih kolona, njihovih tipova podataka i spisak svih ograničenja kao što su primarni i strani ključevi i opcionalnost atributa. Za svaki entitet koji se nalazi u modelu, kreira se jedna tabela.
Kada je ključ primaran pišemo **pk** (skraćeno od primary key), a kada je strani pišemo **fk** (skraćeno od foreign key).
Nazivi tabela moraju biti u množini.

STUDENTI		
ključ	opcionalnost	naziv kolone
pk	*	broj_indeksa
	*	ime
	*	prezime
	*	jmbg
	*	adresa
	*	mejl_adresa
	*	broj_telefona
	*	godina_upisa

PREDMETI		
ključ	opcionalnost	naziv_kolone
pk	*	id_predmeta
fk	*	redni_broj
	*	naziv
	*	broj_bodova
	*	tip
	*	semestar

NASTAVNO_OSOBLJE		
ključ	opcionalnost	naziv_kolone
pk	*	id_zaposlenog
fk	*	id_pozicije
	*	ime
	*	prezime
	*	jmbg
	*	adresa
	*	datum_rodjenja
	*	mejl_adresa
	*	datum_zaposlenja
	*	plata
	*	broj_telefona
	o	diploma

PROFESORI-PREDMETI		
ključ	opcionalnost	naziv kolone
pk, fk1	*	id_zaposlenog
pk, fk2	*	id_predmeta

Uzorak podataka

Nakon mapiranja, pravimo uzorak podataka. Uzorak podataka predstavlja unošenje konkretnih vrednosti u tabele.

Kao i kod mapiranja, nazive tabela pišemo u množini.

GODINE		
redni_broj	ESP_bodovi	opis
1	60	-
2	60	-
3	60	-
4	60	Završna godina

POZICIJE		
id_pozicije	naziv	opis
11	redovni profesor	VIII stepen stručne spreme
22	vanredni profesor	VIII stepen stručne spreme
33	docent	VIII stepen stručne spreme
44	asistent	-
55	saradnik	-

NASTAVNO OSOBLJE											
id_zaposlenog	id_pozicije	ime	prezime	jmbg	adresa	datum_rođenja	mejl_adresa	datum_zaposlenja	plata	broj_telefona	diploma
100	11	Marko	Đurić	0110970456663	Cara Dušana 57	1.10.1970.	mdjuric1@gmail.com	10.10.2005.	80000	0654323431	matematičar
200	11	Milica	Perić	1203968349121	Bregalnička 8	12.3.1968.	mperic1@gmail.com	17.2.2010.	80000	0693234919	informatičar
300	33	Viktor	Jurić	2204973545029	Nemanjina 12	22.4.1973.	vjuric1@gmail.com	13.8.2013.	85000	063323978	matematičar
400	44	Sava	Dimić	0503992715848	Opatijska 17	5.3.1992.	sdimic1@gmail.com	2.9.2018.	40000	0657390010	informatičar
500	22	Sonja	Marić	2505989715646	Žička 41	25.5.1989.	smaric1@gmail.com	29.10.2016.	70000	062847881	informatičar

STUDENTI							
broj_indeksa	ime	prezime	jmbg	adresa	mejl_adresa	broj_telefona	god_upisa
046R17	Milan	Đurić	1202998715942	Čingrijina 5	mdjuric@gmail.com	0654356821	2017
232I17	Petar	Stojić	112998713333	Balkanska 2	pstojic@gmail.com	062345688	2017
093R18	Sara	Matić	1406999848378	Cerska 12	smatic@gmail.com	0694721232	2018
103P19	Ana	Mitić	2210000483432	Moravska 98	amitic@gmail.com	0652998965	2019
019S18	Đorđe	Perić	3012999574720	Ustanička 27	djperic@gmail.com	063454190	2018

PREDMETI					
id_predmeta	redni_broj_god	naziv	broj_bodova	tip	semestar
10	1	Geometrija 1	4	obavezan	1
20	2	Diskretna matematika	5	izborni	4
30	1	Programiranje 2	8	obavezan	2
40	3	Verovatnoća	6	obavezan	5
50	2	Geometrija 2	6	obavezan	3

UPISANE GODINE				
datum	redni_broj	broj_indeksa	status	uplata
13.9.2017.	1	046R17	budžet	-
10.10.2018.	2	046R17	budžet	-
15.10.2019.	3	046R17	budžet	-
2.9.2017.	1	232I17	budžet	-
1.11.2018.	2	232I17	samofinansiranje	uplaćeno
29.10.2019.	2	232I17	samofinansiranje	uplaćeno
10.9.2018.	1	093R18	budžet	-
28.9.2019.	2	093R18	budžet	-
2.9.2019.	1	103P19	samofinansiranje	uplaćeno
6.9.2018.	1	019S18	samofinansiranje	uplaćeno
30.10.2019.	2	019S18	samofinansiranje	uplaćeno

ISPITI					
id_ispita	id_predmeta	broj_indeksa	datum	ocena	komentar
12	10	019S18	12.12.2018.	8	-
23	10	046R17	1.12.2017.	6	poništen
13	30	046R17	14.4.2018.	4	-
43	40	232I17	10.11.2019.	9	-
24	30	046R17	10.6.2018.	7	-

Kreiranje baze

APEX okruženje

Oracle Application Express (APEX) je relacioni sistem za upravljanje bazama podataka kompanije Oracle. U njemu može da se programira pomoću jezika PL/SQL. Razvojno okruženje je jednostavno za korišćenje i dostupno preko veba.

Jezik SQL

Tvorac jezika SQL je Donald Čemberlin, a nastao je u IBM-ovoj istraživačkoj laboratoriji 1974. godine. To je specijalizovan jezik za rad sa sistemima za upravljanje bazama podataka.

Tipovi podataka koji se najčešće koriste su:

- **NUMBER** – Označava broj. U zagradi može da se navede ograničenje, na primer NUMBER(2) označava da broj ima do dve cifre. Ukoliko se pored njega navede još jedan broj, on označava broj decimala.
- **VARCHAR2** – Kratak tekst. Takođe se u zagradi može navesti ograničenje. Na primer VARCHAR2(50) označava tekstualni podatak dužine najviše 50 karaktera.

- **CHAR** – Kratak tekst fiksirane dužine. U zagradi može da se navede broj znakova. Na primer CHAR(5) označava tekstualni podatak dužine tačno pet znakova.
- **CLOB** – Veliki tekstualni podatak.
- **DATE** – Datum.
- **CURRENCY** – Valuta.

Komanda CREATE

Komanda CREATE TABLE se koristi kada kreiramo tabelu i potrebno je pored komande napisati naziv tabele koju kreiramo, a onda u zagradi navesti nazive kolona i kojeg je tipa podatak koji smo uneli.
Kada kreiramo tabele, bitan je redosled. Prvo treba kreirati one nezavisne od drugih, zatim one koje sadrže strani ključ iz tabele koja je već kreirana.

Kreirana tabela pozicije, koja je nezavisna od drugih

```
CREATE TABLE pozicije
(
    id_pozicije NUMBER(10) PRIMARY KEY,
    naziv VARCHAR2(50) NOT NULL,
    opis VARCHAR2(50)
)
```

Kada tabela sadrži strani ključ, potrebno je pored kolone koja ga sadrži napisati ključnu reč **REFERENCES**, nakon koje se navodi ime tabele sa kojom je povezana, a u zagradi se navodi naziv kolone na koju strani ključ pokazuje.

Kreirana tabela nastavno_osoblje, koja sadrži strani ključ id_pozicije

```
CREATE TABLE nastavno_osoblje
(
    id_zaposlenog NUMBER(10) PRIMARY KEY,
    ime VARCHAR2(20) NOT NULL,
    prezime VARCHAR2(20) NOT NULL,
    jmbg CHAR(13) NOT NULL,
    adresa VARCHAR2(50) NOT NULL,
    datum_rodjenja DATE NOT NULL,
    mejl_adresa VARCHAR2(50) NOT NULL,
    datum_zaposlenja DATE NOT NULL,
    plata NUMBER(10) NOT NULL,
    broj_telefona VARCHAR2(20) NOT NULL,
    diploma VARCHAR2(50),
    id_pozicije NUMBER(10) NOT NULL REFERENCES pozicije(id_pozicije)
)
```

Kada imamo barovane veze između dva entiteta i kreiranu novu tabelu koja je oslikava, gde je strani ključ ujedno i primarni u njoj tabeli, tada ključnom reči **PRIMARY KEY** i navođenjem u zagradi naziva tih ključeva uvodimo složeni primarni ključ.

Kreirana tabela profesori-predmeti, gde se vidi veza M:M između entiteta nastavno_osoblje i predmet

```
CREATE TABLE profesori_predmeti
(
    id_zaposlenog NUMBER(10) NOT NULL REFERENCES nastavno_osoblje(id_zaposlenog),
    id_predmeta NUMBER(10) NOT NULL REFERENCES predmeti(id_predmeta),
    PRIMARY KEY (id_zaposlenog,id_predmeta)
)
```

Komanda INSERT

Nakon kreiranja tabele, potrebno je da se podaci unesu komandom **INSERT INTO**. Unosimo red po red, i bitno je da je su podaci nabrojani po redosledu koji odgovara redosledu kolona kad se kreirala tabela.
Podatke tipa DATE, VARCHAR2, CHAR unosimo tako što vrednost stavimo između navodnika.
Kada je podatak tipa NUMBER nisu nam potrebni navodnici.
Kada red ne sadrži neki podatak jer je opcioni, umesto njegove vrednosti unosimo reč *null*.

```
INSERT INTO pozicije
VALUES(11,'redovni profesor','VIII stepen strucne spreme')
```

Podrazumevani format za datum je mm/dd/yyyy. Ako želimo da ga unesemo po željenom šablonu koristimo funkciju **TO_DATE** koja će tekstualni podatak da prebaci u datum po datom šablonu.
Primer: *TO_DATE('01.10.1970.', 'dd.mm.yyyy.')*

```
INSERT INTO nastavno_osoblje
VALUES(100,'Marko','Djuric','0110970456663','Cara Dusana 57','10/01/1970',
'mdjuric1@gmail.com','10/10/2005',80000,'0654323431','matematicar',11)
```

Komanda UPDATE

Ako želimo da ažuriramo unete podatke, koristimo komandu **UPDATE**.

Na primer, ako je student čiji je broj indeksa 103P19 promenio broj telefona, i novi broj je 063454911, tu izmenu ćemo ovako uneti u bazu:

```
UPDATE studenti
SET broj_telefona='063454911'
WHERE broj_indeksa='103P19'
```

Jednom UPDATE komandom možemo promeniti i više redova. Na primer, ako želimo svim zaposlenima koji su na poziciji čiji je identifikacioni broj 11 da podignemo platu za 10%, uradićemo to na sledeći način:

```
UPDATE nastavno_osoblje
SET plata=plata*1.1
WHERE id_pozicije=11
```

Komanda DELETE

Komandom **DELETE** brišemo podatke iz tabele.

Na primer, ako bismo želeli da obrišemo studenta čiji je broj indeksa 103P19, uradićemo to na sledeći način:

```
DELETE FROM studenti
WHERE broj_indeksa='103P19'
```

Upit SELECT

Najčešće korišćena komanda za pretragu podataka iz baze je **SELECT** upit. Kod svakog upita treba da zadamo koje pokatke želimo da prikažemo, iz koje tabele i koji su uslovi da bi neki podatak bio prikazan.

Pomoću upita SELECT i odgovarajućeg koda posle njega, možemo dobiti sve željene podatke iz tabele.

Kada želimo da vidimo sve podatke iz neke tabele, napisaćemo SELECT * FROM (naziv tabele).

Ovako bismo prikazali sadržaj cele tabele ispiti:

```
SELECT * FROM ispiti
```

Rezultat:

ID_ISPITA	DATUM	OCENA	KOMENTAR	BROJ_INDEKSA	ID_PREDMETA
23	12/01/2017	6	ponisten	046R17	10
43	11/10/2019	9	-	232I17	40
13	04/14/2018	4	-	046R17	30
24	06/10/2018	7	-	046R17	30
12	12/12/2018	8	-	019S18	10

Primeri SELECT upita

1. Napisati upit kojim se prikazuju svi podaci o predmetima koji nose više od 5 bodova.

```
SELECT * FROM predmeti WHERE broj_bodova>5
```

2. Napisati upit kojim se prikazuju imena i prezimena studenata u jednoj koloni.

```
SELECT ime ||' '|| prezime "Ime i prezime studenta" FROM studenti
```

Stringove spajamo operatorom ||. Ako želimo da promenimo naziv kolone u rezultatu, pod navodnicima ćemo napisati kako želimo da nam se zove kolona.

3. Napisati upit kojim se prikazuju indentifikacioni brojevi zaposlenih koji predaju obavezne predmete, kao i nazive tih predmeta.

```
SELECT id_zaposlenog, naziv FROM nastavno_osoblje JOIN profesori_predmeti
USING (id_zaposlenog) JOIN predmeti USING (id_predmeta) WHERE tip='obavezan'
```

Kada imamo kombinaciju podataka koji se nalaze u različitim tabelama, koristimo **JOIN** da ih spojimo. Kada se strani i primarni ključ zovu isto koristimo **JOIN USING**, a kada se zovu različito **JOIN ON**.

4. Napisati upit kojim se prikazuju nazivi i brojevi bodova predmeta iz prve godine.

```
SELECT naziv,broj_bodova FROM predmeti WHERE redni_broj=1
```

5. Napisati upit kojim se prikazuju nazivi predmeta iz kojih je student sa imenom Milan polagao ispit.

```
SELECT naziv FROM predmeti JOIN ispiti USING (id_predmeta)
JOIN studenti USING (broj_indeksa) WHERE ime='Milan'
```

6. Napisati upit kojim se prikazuju naziv predmeta i ocena sa ispita koje je polagao student sa prezimenom Djuric, sortirano po ocenama.

```
SELECT naziv, ocena FROM ispiti JOIN studenti USING (broj_indeksa)
JOIN predmeti USING(id_predmeta) WHERE prezime='Djuric' ORDER BY ocena
```

Sortiranje se vrši pomoću **ORDER BY** klauzule.

Jednostavne funkcije

U upitima se mogu koristiti dve vrste funkcija: jednostavne i grupne. Jednostavne se izvršavaju tako što se primenjuju na svaki red pojedinačno, dok grupne vraćaju jedan rezultat za više redova.

7. Prikazati koliko meseci radi na fakultetu zaposleni koji se zove Sava.

```
SELECT ROUND(MONTHS_BETWEEN(SYSDATE,datum_zaposlenja))
FROM nastavno_osoblje WHERE ime='Sava'
```

Za rad sa datumima koristimo funkcije **SYSDATE** i **MONTHS_BETWEEN**. SYSDATE vraća trenutni datum, a MONTHS_BETWEEN računa broj meseci izmedju dva datuma zadata u zagradi. Funkcija **ROUND** zaokružava broj.

8. Prikazati ocenu i komentar svakog ispita iz predmeta Geometrija 1. Ukoliko nema komentara, napisati 'Nema komentara'.

```
SELECT ocena, NVL(komentar,'Nema komentara') FROM ispiti
JOIN predmeti USING (id_predmeta) WHERE naziv='Geometrija 1'
```

Funkcija **NVL** analizira prvi parameter i vraća njegovu vrednost ako postoji, a ako je null onda vraća vrednost zapisanu kao drugi parameter u zagradi.

Grupne funkcije

9. Napisati upit koji prikazuje prosečnu, najveću i najmanju platu nastavnog osoblja.

```
SELECT AVG(plata), MAX(plata), MIN(plata)
FROM nastavno_osoblje
```

Funkcija **AVG** računa prosečnu vrednost i radi samo sa brojevima. Funkcije **MAX** i **MIN** vraćaju maksimalnu, odnosno minimalnu vrednost i rade sa svim tipovima podataka čije vrednosti mogu da se upoređuju.

10. Napisati upit koji prebrojava koliko ispita je polagao student sa brojem indeksa 232I17.

```
SELECT COUNT(*) FROM ispiti WHERE broj_indeksa='232I17'
```

Funkcija **COUNT** prebrojava redove.

11. Napisati upit koji za svakog studenta prikazuje koliko ispita je polagao.

```
SELECT COUNT(*),broj_indeksa
FROM ispiti GROUP BY broj_indeksa
```

GROUP BY se koristi za grupisanje redova. U ovom primeru smo grupisali prema broju indeksa, da bi se za svakoga prebrojalo koliko puta je izašao na ispit.

12. Napisati upit koji vraća broj indeksa i prosek za svakog studenata koji ima prosečnu ocenu veću od 7.

```
SELECT broj_indeksa, AVG(ocena) FROM ispiti
GROUP BY broj_indeksa HAVING AVG(ocena)>7
```

Grupne funkcije mogu da se koriste u **SELECT** i **HAVING** delu upita, ali ne mogu u **WHERE** delu.

Jezik PL/SQL

O jeziku

PL/SQL predstavlja kombinaciju SQL jezika zajedno sa proceduralnim karakteristikama programskih jezika. PL/SQL je skraćenica za Procedural Language extension to SQL. Neke od stvari koje možemo da kreiramo pomoću njega su neimenovani blokovi, procedure i funkcije. Pored naredbi jezika SQL, ovaj jezik nam pruža korišćenje i drugih elemenata kao što su grananja i ciklusi.
Sve naredbe u ovom jeziku se grupišu u blokove. Struktura osnovnih blokova se sastoji iz zaglavlja, deklaracije promenljivih, izvršnog dela i dela za obradu izuzetaka. Anonimni blokovi nemaju zaglavlje. Deklaracija promenljivih (**DECLARE**) i obrada izuzetaka (**EXCEPTION**) nisu uvek obavezni.

Komanda SELECT INTO

Komanda **SELECT INTO** se koristi kada se uzimaju podaci iz tačno jednog reda.

1. Prikazati ime i prezime studenta sa brojem indeksa 103P19.

```
DECLARE
    v_student VARCHAR2(50);
BEGIN
    SELECT ime||' '||prezime INTO v_student
    FROM studenti WHERE broj_indeksa='103P19';
    DBMS_OUTPUT.PUT_LINE('Ime i prezime studenta: '||v_student);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Ne postoji student sa tim brojem indeksa');
END
```

2. Prikazati ime i prezime studenta čiji se broj indeksa unosi sa tastature.

```
DECLARE
    v_student VARCHAR2(50);
    v_broj_indeksa studenti.broj_indeksa%TYPE;
BEGIN
    v_broj_indeksa:=:Broj_indeksa;
    SELECT ime||' '||prezime INTO v_student
    FROM studenti WHERE broj_indeksa=v_broj_indeksa;
```

```
    DBMS_OUTPUT.PUT_LINE('Ime i prezime studenta: '||v_student);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Ne postoji student sa tim brojem indeksa');
END
```

Kursori

Ukoliko želimo da prikažemo podatke iz više redova, moramo koristiti **kusor**.

Za rad sa kursorima neophodni su sledeći koraci:

- kursor se kreira u DECLARE odeljku i veže za SELECT upit
- kursor se otvara u telu bloka
- pomoću njega se čita jedan po jedan red dok se ne pročitaju svi redovi odgovarajućeg SELECT upita
- na kraju se kursor zatvori

3. Prikazati ime, prezime, broj indeksa i broj telefona svakog studenta. Ime i prezime prikazati u jednom redu.

```
DECLARE
    v_student VARCHAR2(50);
    v_broj_indeksa studenti.broj_indeksa%TYPE;
    v_telefon studenti.broj_telefona%TYPE;
    CURSOR kursor_student IS SELECT ime||' '||prezime,
    broj_telefona FROM studenti;
BEGIN
    OPEN kursor_student;
    LOOP
    FETCH kursor_student INTO v_student, v_telefon;
    EXIT WHEN kursor_student%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Ime i prezime studenta: '||v_student);
    DBMS_OUTPUT.PUT_LINE('Broj telefona studenta: '||v_telefon);
    END LOOP;
    CLOSE kursor_student;
END
```

4. Prikazati za svakog studenta datum kada se upisao u svaku godinu.

```
DECLARE
    v_student VARCHAR2(50);
    v_datum upisane_godine.datum%TYPE;
    v_redni_broj upisane_godine.redni_broj%TYPE;
```

```
    CURSOR kursor_student IS SELECT ime||' '||prezime, datum, redni_broj
    FROM studenti JOIN upisane_godine USING (broj_indeksa);
BEGIN
    OPEN kursor_student;
    LOOP
    FETCH kursor_student INTO v_student, v_datum,v_redni_broj;
    EXIT WHEN kursor_student%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Student '||v_student||' je upisao '||
    v_redni_broj||'. godinu datuma '||v_datum);
    END LOOP;
    CLOSE kursor_student;
END
```

Procedure

Procedura je imenovani blok koji obavlja određeni zadatak. Definiše se niz akcija koje će se obaviti nakon poziva procedure.

5. Kreirati proceduru koja na osnovu identifikacionog broja zaposlenog vraća njegovo ime, prezime, datum zaposlenja i platu.

Kreiranje procedure:

```
CREATE OR REPLACE PROCEDURE zaposleni_podaci(p_id_zaposlenog
      nastavno_osoblje.id_zaposlenog%TYPE,
      p_ime OUT nastavno_osoblje.ime%TYPE,
      p_prezime OUT nastavno_osoblje.prezime%TYPE,
      p_datum_zaposlenja OUT nastavno_osoblje.datum_zaposlenja%TYPE,
      p_plata OUT nastavno_osoblje.plata%TYPE) AS
BEGIN
      SELECT ime, prezime, datum_zaposlenja, plata INTO p_ime, p_prezime,
      p_datum_zaposlenja, p_plata
      FROM nastavno_osoblje WHERE id_zaposlenog=p_id_zaposlenog;
END
```

```
Procedure created.
```

Poziv procedure:

```
DECLARE
      v_ime nastavno_osoblje.ime%TYPE;
```

```
      v_prezime nastavno_osoblje.prezime%TYPE;
      v_datum_zaposlenja nastavno_osoblje.datum_zaposlenja%TYPE;
      v_plata nastavno_osoblje.plata%TYPE;
BEGIN
      zaposleni_podaci(100,v_ime,v_prezime,v_datum_zaposlenja,v_plata);
      DBMS_OUTPUT.PUT_LINE(v_ime||' '||v_prezime||' se zaposlio datuma '||
      v_datum_zaposlenja||'. ');
      DBMS_OUTPUT.PUT_LINE('Njegova plata iznosi: '||v_plata);
END
```

```
Marko Djuric se zaposlio datuma 10/10/2005.
Njegova plata iznosi: 80000
```

Funkcije

Funkcije predstavljaju procedure koje vraćaju neku vrednost.

6. Kreirati funkciju koja vraća prosečnu platu nastavnog osoblja fakulteta.

Kreiranje funkcije:

```
CREATE OR REPLACE FUNCTION prosecna_plata RETURN NUMBER AS v_iznos NUMBER;
BEGIN
      SELECT ROUND(AVG(plata),2) INTO v_iznos FROM nastavno_osoblje;
      RETURN v_iznos;
END
```

Poziv funkcije:

```
BEGIN
      DBMS_OUTPUT.PUT_LINE(prosecna_plata);
END
```

Zaključak

Za fakultet je veoma bitno da ima ispravnu i dobro uređenu bazu podataka. Ta baza treba da sadrži sve informacije o zaposlenima na fakultetu, o studentima, predmetima ali i o dešavanjima na njemu kao što su ispiti. Važno je napraviti preglednu i dobro popunjenu bazu podataka gde su svi podaci dobro i jasno organizovani. Bitno je da u svakom trenutku može brzo da se dođe do željenih podataka. Treba da bude omogućena laka izmena podataka. Svake godine se upisuje dosta novih

studenata i potrebno je da se unesu novi podaci, zbog čega je ključno imati dobru bazu koja će to maksimalno uprostiti. Pomoću baze ima se uvid u sve aktivnosti osoba na fakultetu.

Literatura

- https://www.plsqltutorial.com
- https://www.w3schools.com/sql/default.asp
- https://apex.oracle.com/en
- Skripta za modele i baze podataka, Računarska gimnazija, Duša Vuković, šk. 2020/21. god
- Kurs Modeli i baze podataka na Moodle platformi, https://learning.rg.edu.rs/
- https://support.microsoft.com/sr-latn-rs/access
- https://sr.wikipedia.org/Релационе_базе_података
- http://www.matf.bg.ac.rs