Baze podataka

Vežbe – SQL

Sadržaj

- Rad u učionici
- SQL - uvod i primer
- Jezik za definiciju podataka (DDL)
- Jezik za manipulaciju podacima (DML)
- Upitni jezik – SELECT naredba

Rad u učionici

Rad u učionici

- Baze podataka
  - studentska korisnička šema (user schema)
  - pod nazivom indXY
    - username: indXY
    - password: ftn
  - Gde je ind oznaka studijskog programa, X broj indeksa, a Y godina upisa

Rad u učionici

- Podaci potrebni za konektovanje na bazu

	MI A2-1, MI A2-2, MI A2-3	**Učionice računarskog centra**	**NTP-312, NTP-316, NTP-317**	**Kod kuće**
Username	indXY[1]	indXY[1]	indXY[1]	*[2]
Password	ftn	ftn	ftn	*[2]
Role	default	default	default	default
Host Name	192.168.18.9	192.168.7.204	192.168.18.8	localhost
Port	1522	1521	1521	1521
Oracle SID	db2016	bp1		
Service name			orclpdb1	xepdb1

1 – ind oznaka studijskog programa, X broj indeksa, a Y godina upisa
2 – username i password koji su postavljeni tokom izvršavanja skripta za kreiranje korisnika

SQL - uvod i primer

Uvod

- SQL (engl. *Structured Query Language*)
  - standardni jezik relacionih sistema za upravljanje bazama podataka
  - jezik visokog nivoa deklarativnosti
  - objedinjuje funkcije jezika za definiciju podataka, jezik za manipulaciju podacima i upitni jezik
- Namena i zadaci SQL-a u okviru sistema za upravljanje bazama podataka
  - administratorima baze podataka za obavljanje poslova administracije
  - programerima za izradu aplikacija nad bazom podataka
  - krajnjim korisnicima, za postavljanje upita nad bazom podataka
- SQL se javlja u formama:
  - interaktivnog jezika sistema za upravljanje bazama podataka
  - ugrađenog jezika u jezik III generacije
  - sastavnog dela jezika IV generacije

Uvod

- Saglasno nameni i vrstama korisnika koji ga upotrebljavaju, SQL obezbeđuje realizaciju sledećih zadataka:
  - realizacija implementacione šeme baze podataka i definisanje fizičke organizacije baze podataka (naredbe CREATE, DROP i ALTER)
  - ažuriranje baze podataka putem jezika za manipulaciju podacima (naredbe INSERT, DELETE i UPDATE)
  - izražavanje upita putem upitnog jezika (naredba SELECT)
  - automatsko održavanje rečnika podataka
  - transakcijska obrada podataka (naredbe COMMIT, ROLLBACK, SAVEPOINT)

Uvod

- Saglasno nameni i vrstama korisnika koji ga upotrebljavaju, SQL obezbeđuje realizaciju sledećih zadataka:
  - zaključavanje resursa (naredba LOCK TABLE)
  - zaštita podataka od neovlašćenog pristupa (naredbe GRANT, REVOKE)
  - praćenje zauzeća resursa i performansi rada sistema za upravljanje bazama podataka (naredbe AUDIT, EXPLAIN PLAN)
  - obezbeđenje proceduralnog načina obrade podataka "slog po slog" (naredbe za rad sa kursorom: OPEN, FETCH, CLOSE)
- **Sintaksa SQL-a zavisi od proizvođača sistema za upravljanje bazama podataka!**

Primer – relacioni model

- Radnik({Mbr, Ime, Prz, Sef, Plt, God,Pre}, {Mbr}),
- Projekat({Spr, Nap, Nar, Ruk}, {Spr}),

- Radproj({Spr, Mbr, Brc}, {Spr + Mbr}),
- Radproj[Mbr] ⊆ Radnik[Mbr],
- Radproj[Spr] ⊆ Projekat[Spr],

- Projekat[Ruk] ⊆ Radnik[Mbr],
- Null(Projekat, Ruk) = ⊥

- Radnik[Sef] ⊆ Radnik[Mbr],
- Null(Radnik, Sef) = T

Tabela radnik

- Mbr - matični broj radnika
- Ime - ime radnika
- Prz - prezime radnika
- Sef - maticni broj direktno nadređenog rukovodioca - radnika
- Plt - mesečni iznos plate radnika
- God - Datum rođenja radnika
- Pre – godišnja premija na platu radnika

- Obeležja Mbr, Ime, Prz ne smeju imati null vrednost. Plata ne sme biti manja od 500

Tabela projekat

- Spr - šifra projekta
- Nap - naziv projekta
- Nar - naručilac projekta
- Ruk - rukovodilac projekta

- Obeležja Spr i Ruk ne smeju imati null vrednost, dok obeležje Nap mora imati jedinstvenu vrednost

Tabela radproj

- Spr - šifra projekta
- Mbr - matični broj radnika
- Brc - broj časova nedeljnog angažovanja na projektu

- Sva tri obeležja ne smeju da imaju null vrednost

Jezik za definiciju podataka (DDL)

Kreiranje tabele

```
CREATE TABLE [šema.]<naziv_tabele>
(<naziv_kolone> <tip_podatka> [DEFAULT izraz] [, ...]
CONSTRAINT <naziv_ogranicenja>
<definicija_ograničenja> [, ...]);
```

- Šema - poklapa se sa nazivom korisnika
- DEFAULT opcija:
  - Specificira se predefinisana vrednost za kolonu, koja se koristi ukoliko se prilikom ubacivanja podataka izostavi vrednost za tu kolonu

Naziv tabele i kolone

- mora početi slovom,
- mora biti između 1 i 30 znakova dužine,
- mora sadržati samo velika i mala slova, cifre, _, $ i #,
- ne sme se poklapati sa nazivom nekog drugog objekta koji je kreirao isti korisnik,
- ne sme biti rezervisana reč Oracle servera i
- nazivi nisu case sensitive.

SQL tipovi podataka

Tip podataka	Opis
VARCHAR2(size)	Niz karaktera promenljive dužine, maksimalne dužine size; minimalna dužina je 1, maksimalna je 4000.
CHAR(size)	Niz karaktera fiksne dužine od size bajtova; default i minimalna dužina je 1, maksimalna dužina je 2000.
NUMBER(p, s)	Broj ukupnog broja cifara p, od čega je s cifara iza decimalnog zareza; p može imati vrednosti od 1 do 38.
DATE	Vrednosti za vreme i datum.
LONG	Niz karaktera promenljive dužine do 2 GB. (za kompatibilnost sa starijim verzijama Oracle-a).
CLOB	Niz karaktera promenljive dužine do 4 GB.
BLOB	Binarni podaci do 4 GB.
BFILE	Binarni podaci smešteni u eksternom fajlu do 4 GB.
ROWID	Jedinstvena adresa vrste u tabeli.

Tabela radnik

```
CREATE TABLE radnik (
   Mbr integer NOT NULL,
   Ime varchar(20) NOT NULL,
   Prz varchar(25) NOT NULL,
   Sef integer,
   Plt decimal(10, 2),
   Pre decimal(6, 2),
   God date NOT NULL,
   CONSTRAINT radnik_PK PRIMARY KEY (Mbr),
   CONSTRAINT radnik_FK FOREIGN KEY (Sef) REFERENCES Radnik (Mbr),
   CONSTRAINT radnik_CH CHECK (Plt>500)
);
```

Tabela projekat

```
CREATE TABLE projekat (
   Spr integer not null,
   Ruk integer not null,
   Nap varchar(30),
   Nar varchar(30),
   CONSTRAINT projekat_PK PRIMARY KEY (Spr),
   CONSTRAINT projekat_FK FOREIGN KEY (Ruk) REFERENCES Radnik (Mbr),
   CONSTRAINT projekat_UK UNIQUE (Nap)
);
```

Tabela radproj

```
CREATE TABLE radproj (
   Spr integer NOT NULL,
   Mbr integer NOT NULL,
   Brc integer NOT NULL,
   CONSTRAINT radproj_PK PRIMARY KEY (Spr, Mbr),
   CONSTRAINT radproj_rad_FK FOREIGN KEY (Mbr) REFERENCES radnik(Mbr),
   CONSTRAINT radproj_prj_FK FOREIGN KEY (Spr) REFERENCES projekat(Spr)
);
```

Tabela faze_projekta – Zadatak za vežbu

- Kreirati tabelu faze_projekta
  - faze_projekta({Spr , Sfp, Rukfp, Nafp, Datp}, {Spr+ Sfp})
    - faze_projekta[Spr] ⊆ projekat[Spr],
    - faze_projekta[Rukfp] ⊆ radnik[Mbr]

- Sfp - šifra faze projekta,
- Spr - sifra projekta,
- Rukfp - rukovodilac faze projekta,
- Nafp - naziv faze projekta,
- Datp - datum početka faze projekta

- Obeležja Spr i Sfp ne smeju imati null vrednost.
- Obeležje Nafp mora imati jedinstvenu vrednost.

Tabela faze_projekta – Zadatak za vežbu

```
CREATE TABLE faze_projekta (
   Spr integer not null,
   Sfp integer not null,
   Rukfp integer,
   Nafp varchar2(20),
   Datp date,
   CONSTRAINT faze_projekta_PK PRIMARY KEY (spr, sfp),
   CONSTRAINT faze_projekta_fk1 FOREIGN KEY (spr) REFERENCES projekat(spr),
   CONSTRAINT faze_projekta_fk2 FOREIGN KEY (rukfp) REFERENCES radnik(mbr),
   CONSTRAINT faze_projekta_uk UNIQUE(nafp)
);
```

Izmena definicije tabele

- ALTER TABLE
- Alter table iskaz služi za:
  - dodavanje nove kolone,
  - modifikaciju postojeće kolone,
  - definisanje podrazumevane vrednosti za novu kolonu,
  - brisanje kolone i
  - dodavanje oraničenja.

ALTER TABLE

```
ALTER TABLE <naziv_tabele>
ADD (<naziv_kolone> <tip_podatka> [DEFAULT izraz]
   [, <naziv_kolone> <tip_podatka>]...);

ALTER TABLE <naziv_tabele>
MODIFY (<naziv_kolone> <tip_podatka> [DEFAULT izraz]
      [, <naziv_kolone> <tip_podatka>]...);
```

ALTER TABLE

```
ALTER TABLE <naziv_tabele>
DROP COLUMN (<naziv_kolone>);

ALTER TABLE <naziv_tabele>
ADD CONSTRAINT <naziv_ogranicenja>
<definicija_ogranicenja>;
```

Izmena definicije tabele – Zadatak za vežbu

- U tabelu faze_projekta dodati atribut:
  - Datz - datum završetka faze projekta
  - Datz ne sme biti manji od Datp

```
ALTER TABLE faze_projekta
    ADD datz date
    ADD CONSTRAINT dat_ch CHECK (datp<=datz);

ALTER TABLE faze_projekta
    ADD(datz date, CONSTRAINT dat_ch CHECK (datp<=datz));

ALTER TABLE faze_projekta
    ADD datz date;
ALTER TABLE faze_projekta
    ADD CONSTRAINT dat_ch CHECK (datp<=datz);
```

Brisanje definicije tabele

```
DROP TABLE <naziv_tabele>;
```

Brisanje definicije tabele – Zadatak za vežbu

- Izbrisati tabelu faze_projekta.

```
DROP TABLE faze_projekta;
```

Jezik za manipulaciju nad podacima (DML)

Ažuriranje baze podataka

- INSERT
- DELETE
- UPDATE

Ažuriranje baze podataka

- INSERT – dodavanje nove torke

```
INSERT INTO <naziv_tabele> [(<lista_obeležja >)]
VALUES (<lista_konstanti >) | SELECT ...;
```

Ažuriranje baze podataka

- INSERT – dodavanje nove torke

```
INSERT INTO radnik (mbr, ime, prz, plt, sef, god)
VALUES (201, 'Ana', 'Jovic', 30000, null, '18-AUG-1971');

INSERT INTO projekat (spr, nap, ruk)
VALUES (90, 'P1', 201);

INSERT INTO radproj (mbr, spr, brc)
VALUES (201, 90, 5);
```

Skriptovi

- Pokrenuti odgovarajuće skriptove za punjenje baze podataka
  - radnik.sql
  - radproj.sql
  - projekat.sql

Ažuriranje baze podataka

- DELETE – brisanje postojećih torki

```
DELETE FROM <naziv_tabele>
[WHERE (<uslov_selekcije>)];
```

Ažuriranje baze podataka

- DELETE – brisanje postojećih torki

```
DELETE FROM radnik;

DELETE FROM radnik
WHERE mbr = 701;

DELETE FROM radproj;
```

Ažuriranje baze podataka – Zadatak za vežbu

- Probati brisanje torke koja je referencirana od strane neke druge torke.

Ažuriranje baze podataka

- UPDATE – modifikacija postojećih torki

```
UPDATE <naziv_tabele>
SET <obeležje>= <aritm_izraz>
{,<obeležje>= <aritm_izraz>}
[WHERE (<uslov_selekcije>)];
```

Ažuriranje baze podataka

- UPDATE – modifikacija postojećih torki

```
UPDATE radnik
SET plt = plt*1.2;

UPDATE radnik
SET plt = plt*1.2
WHERE mbr = 201;
```

Transakcija

- Najmanja jedinica obrade podataka, takva da
  - prevodi bazu podataka iz jedno u drugo (ne nužno različito) konzistentno stanje, s obzirom na implementirana ograničenja
  - sadrži operacije upita ili/i operacije ažuriranja podataka u bazi podataka
- Efekti izvođenja transakcije se, na kraju, u celosti
  - potvrđuju (COMMIT) i tada postaju vidljivi ostalim korisnicima u sistemu, ili
  - poništavaju (ROLLBACK) i ostavljaju obrađivani deo baze podataka u stanju kakvo je važilo neposredno pre početka njenog izvođenja

Upitni jezik – SELECT naredba

Izražavanje upita i osnovna struktura naredbe SELECT

- Sve vrste upita se u SQL-u izražavaju putem naredbe SELECT. Osnovna struktura SELECT naredbe je:

```
SELECT * | <lista_obeležja>
FROM <lista_tabela>
[WHERE <uslov_selekcije>];
```

  - <lista_obeležja> sadrži obeležja nad kojima se formira rezultat upita,
  - <lista_tabela> sadrži nazive tabela potrebne za realizaciju upita,
  - <uslov_selekcije> izražava uslov selekcije podataka iz tabela koje su navedene iza službene reči FROM
  - sadržaj naveden unutar simbola „[“ i „]“ označava da je taj deo sintakse opcioni tj neobavezan

Upiti nad jednom tabelom – Primer

- Izlistati sadržaj svih tabela

```
SELECT * FROM radnik;

SELECT * FROM projekat;

SELECT * FROM radproj;
```

Upiti nad jednom tabelom – Zadatak za vežbu

- Prikazati imena i prezimena svih radnika.

```
SELECT ime, prz
FROM radnik;
```

DISTINCT – Zadatak za vežbu

```
SELECT [DISTINCT] <lista_obeležja>
FROM <lista_tabela>
[WHERE <uslov_selekcije>];
```

- Izlistati različita imena radnika, tj. imena bez ponovljenih vrednosti.

```
SELECT DISTINCT ime
FROM radnik;
```

DISTINCT – Zadatak za vežbu

- Izlistati različita imena i prezime radnika, tj. imena i prezimena radnika bez ponovljenih vrednosti.

```
SELECT DISTINCT ime, prz
FROM radnik;
```

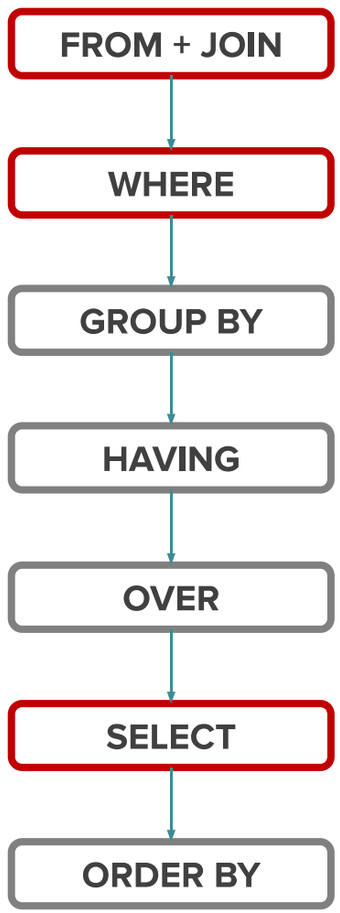
WHERE <uslov_selekcije> – Zadatak za vežbu

```
SELECT *|[DISTINCT] <lista_obeležja>|izraz
FROM <lista_tabela>
WHERE <uslov_selekcije>;
```

- Izlistati mbr, ime i prezime radnika koji imaju platu veću od 25000.

```
SELECT mbr, ime, prz
FROM radnik
WHERE plt>25000;
```

Redosled izvršavanja klauzula



Aritmetički izrazi – Zadatak za vežbu

```
SELECT *|[DISTINCT] <lista_obeležja>|izraz
FROM <lista_tabela>
WHERE <uslov_selekcije>;
```

- Izlistati za svakog radnika mbr, ime, prezime, mesečnu i godišnju platu.

```
SELECT mbr, ime, prz, plt, plt*12
FROM radnik;
```

NULL vrednost – Zadatak za vežbu

x `IS NULL` – x je nula vrednost

x `IS NOT NULL` – x nije nula vrednost

- Izlistati mbr, ime i prezime radnika koji nemaju šefa.

```
SELECT mbr, ime, prz
FROM radnik
WHERE sef IS NULL;
```

- Izlistati mbr, ime i prezime radnika koji imaju šefa.

```
SELECT mbr, ime, prz
FROM radnik
WHERE sef IS NOT NULL;
```

BETWEEN – Zadatak za vežbu

- Izlistati mbr, ime, prezime radnika čija je plata između 20000 i 24000 dinara.

```sql
SELECT mbr, ime, prz
FROM radnik
WHERE plt BETWEEN 20000 AND 24000;

SELECT mbr, ime, prz
FROM radnik
WHERE plt >= 20000 AND plt <= 24000;
```

NOT BETWEEN – Zadatak za vežbu

- Izlistati ime, prezime i godinu rođenja radnika koji nisu rođeni između 1973 i 1980.

```
SELECT ime, prz, god
FROM radnik
WHERE god NOT BETWEEN '01-JAN-1973' AND '31-DEC-1980';
```

LIKE – Zadatak za vežbu

<obeležje> LIKE <uzorak>

- LIKE operator se koristi za poređenje stringova
- Sadržaj baze podataka razlikuje mala i velika slova
- Karakter „_” zamenjuje tačno jedan proizvoljan karakter
- Karakter „%” zamenjuje niz od nula ili više proizvoljnih karaktera

- Izlistati mbr, ime, prezime radnika čije prezime počinje na slovo M.

```
SELECT mbr, ime, prz
FROM radnik
WHERE prz LIKE 'M%';
```

LIKE – Zadatak za vežbu

- Izlistati mbr, ime, prezime radnika čije ime sadrži slovo a na drugoj poziciji.

```
SELECT mbr, ime, prz
FROM radnik
WHERE ime LIKE '_a%';
```

LIKE – Zadatak za vežbu

- Izlistati imena radnika koja počinju na slovo E. Imena ne bi trebalo da se ponavljaju.

```
SELECT DISTINCT ime
FROM radnik
WHERE ime LIKE 'E%';
```

LIKE – Zadatak za vežbu

- Izlistati radnike koji u svom prezimenu imaju slovo E (e).

```
SELECT mbr, ime, prz
FROM radnik
WHERE prz LIKE '%e%' OR prz LIKE '%E%';
```

NOT LIKE – Zadatak za vežbu

<obeležje> NOT LIKE <uzorak>

- Izlistati matične brojeve, imena i prezimena radnika čije ime ne počinje slovom A.

```
SELECT mbr, ime, prz
FROM radnik
WHERE ime NOT LIKE 'A%';
```

IN – Zadatak za vežbu

- Izlistati matične brojeve radnika koji rade na projektima sa šifrom 10, 20 ili 30.

```sql
SELECT DISTINCT mbr
FROM radproj
WHERE spr IN (10, 20, 30);

SELECT DISTINCT mbr
FROM radproj
WHERE spr = 10 OR spr = 20 OR spr = 30;
```

IN – Zadatak za vežbu

- Izlistati matične brojeve radnika koji rade na projektu sa šifrom 10 ili rade 2, 4 ili 6 sati.

```
SELECT DISTINCT mbr
FROM radproj
WHERE brc IN (2, 4, 6) OR spr = 10;
```

NOT IN – Zadatak za vežbu

- Izlistati matične brojeve, imena i prezimena radnika koji se ne zovu Ana ili Sanja.

```
SELECT mbr, ime, prz
FROM radnik
WHERE ime NOT IN ('Ana','Sanja');
```

Uređivanje izlaznih rezultata – ORDER BY – Zadatak za vežbu

```
SELECT * | [DISTINCT] <lista_obeležja> | izraz
FROM <lista_tabela>
WHERE <uslov_selekcije>
ORDER BY <podlista_obeležja>;
```

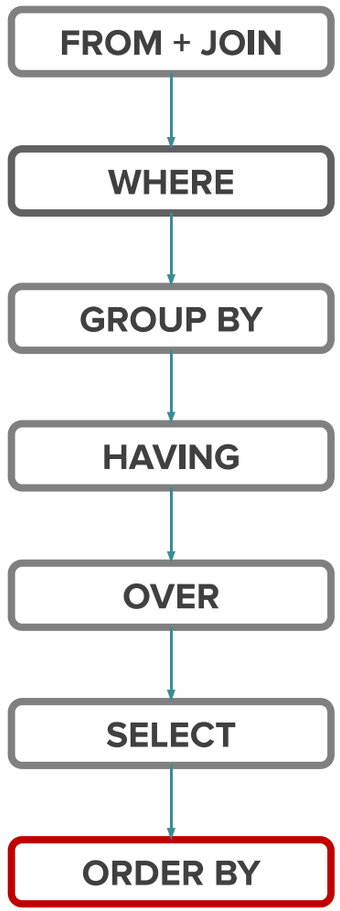
- ORDER BY je uvek poslednja klauzula naredbe SELECT
- ASC označava sortiranje u rastućem redosledu
- DESC označava sortiranje u opadajućem redosledu
- Sortiranje u rastućem redosledu je podrazumevano

Uređivanje izlaznih rezultata – ORDER BY – Zadatak za vežbu

- Prikazati matične brojeve, imena i prezimena radnika koji imaju šefa sortirano po prezimenu.

```
SELECT mbr, ime, prz
FROM radnik
WHERE sef IS NOT null
ORDER BY prz ASC;
```

Redosled izvršavanja klauzula



Primeri upotrebe klauzule ORDER BY

```
SELECT mbr, ime, prz, plt
FROM radnik
ORDER BY prz, ime;
```

```
SELECT mbr, ime, prz, plt
FROM radnik
ORDER BY prz ASC, ime ASC;
```

```
SELECT mbr, ime, prz, plt
FROM radnik
ORDER BY prz ASC, ime DESC;
```

Primeri upotrebe klauzule ORDER BY

```sql
SELECT mbr, prz, ime
FROM radnik
ORDER BY 2, 3, plt;

SELECT mbr, prz, ime
FROM radnik
ORDER BY 2, 3, plt*1.17;
```

ORDER BY – Zadatak za vežbu

- Prikazati matične brojeve, imena, prezimena i plate radnika po opadajućem redosledu iznosa plate.

```
SELECT mbr, ime, prz, plt Plata
FROM radnik
ORDER BY Plata DESC;
```

ANY – Zadatak za vežbu

```
WHERE x ϴ ANY (<lista_vrednosti>)
   ϴ ∈ {<, >, <=, >=, !=, =}
```

- Primer:

  x = ANY (<lista_vrednosti>)

  x je jednako makar jednoj vrednosti u <lista_vrednosti>

- Prikazati matične brojeve, imena, prezimena i platu radnika koji se zovu Pera ili Moma.

```
SELECT mbr, ime, prz, plt
FROM radnik
WHERE ime = ANY ('Pera', 'Moma');
```

ALL – Zadatak za vežbu

```
WHERE x ϴ ALL (<lista_vrednosti>)
   ϴ ∈ {<, >, <=, >=, !=, =}
```

- Primer:

  x != ALL (<lista_vrednosti>)

  x je različito od svake vrednosti u <lista_vrednosti>

- Prikazati matične brojeve, imena, prezimena i platu radnika koji se ne zovu Pera ili Moma.

```
SELECT mbr, ime, prz, plt
FROM radnik
WHERE ime != ALL ('Pera', 'Moma');
```

Upotreba skupovnih funkcija – Zadatak za vežbu

- Prikazati matične brojeve i plate radnika uvećane za NULL vrednost.

```
SELECT mbr, plt + null
FROM radnik;
```

- Prikazati matične brojeve i plate radnika uvećane za godišnju premiju (pre).

```
SELECT mbr, plt + pre
FROM radnik;
```

Funkcija NVL(izraz, konstanta) – Zadatak za vežbu

- Prikazati matične brojeve i plate radnika uvećane za godišnju premiju. Ukoliko za nekog radnika vrednost premije ne postoji, smatrati da ona iznosi 0.

```
SELECT mbr, plt + NVL(pre, 0)
FROM radnik;
```

Funkcija COALESCE (i1, i2, ..., in)

- Funkcija COALESCE() prihvata niz argumenata i vraća prvi za koji vredi da je različit od NULL

```
SELECT COALESCE(NULL, 1) -- return 1
FROM dual;
```

- Ukoliko su argumenti istog tipa, povratna vrednost funkcije COALESCE() će biti tog tipa
- Ukoliko su argumenti različitog tipa, funkcija COALESCE() će izvršiti implicitnu konverziju svih argumenta na tip prvog argumenta koji nije NULL, ukoliko je konverzija nemoguća Oracle će izbaciti grešku.

Funkcija COUNT – Zadatak za vežbu

- **COUNT(*)** – vraća ukupan broj selektovanih torki
- **COUNT(<obeležje>)** – vraća ukupan broj selektovanih torki za koje vrednost <obeležja> nije null vrednost
- **COUNT(DISTINCT <obeležje>)** – vraća ukupan broj različitih torki za koje vrednost <obeležja> nije null vrednost

- Koliko ima radnika?

```
SELECT COUNT(*)
FROM radnik;
```

Koliko ima šefova?

```
SELECT COUNT(DISTINCT sef)
broj_sefova
FROM radnik;
```

Funkcije MAX i MIN – Zadatak za vežbu

- **MAX(<obeležje>)** – vraća maksimalnu vrednost za <obeležje>, uzimajući u obzir sve selektovane torke
- **MIN(<obeležje>)** – vraća minimalnu vrednost za <obeležje>, uzimajući u obzir sve selektovane torke

- Prikazati minimalnu i maksimalnu platu radnika.

```
SELECT MIN(plt) minimalna_plt, MAX(plt) maksimalna_plt
FROM radnik;
```

Funkcija SUM – Zadatak za vežbu

- **SUM(<obeležje>)** – vraća zbir vrednosti datog <obeležja> za sve selektovane torke, uključujući višestruko ponavljanje istih torki
- **SUM(DISTINCT <obeležje>)** – vraća zbir vrednosti datog <obeležja> za sve različite selektovane torke
- SUM funkcija ingoriše null vrednosti iz skupa

- Prikazati broj radnika i ukupnu mesečnu platu svih radnika.

```
SELECT COUNT(mbr) "Broj radnika", SUM(plt) "Ukupna mesecna plata"
FROM radnik;
```

Funkcija AVG – Zadatak za vežbu

- **AVG(<obeležje>)** – vraća srednju vrednost datog <obeležja> za sve selektovane torke, uključujući višestruko ponavljanje istih torki
- **AVG(DISTINCT <obeležje>)** – vraća srednju vrednosti datog <obeležja> za sve različite selektovane torke
- AVG funkcija ingoriše null vrednosti iz skupa

- Prikazati broj radnika, prosečnu mesečnu platu i ukupnu godišnju platu svih radnika.

```
SELECT COUNT(*) "Broj radnika", AVG(plt) "Prosečna plata",
       12*SUM(plt) "Godišnja plata"
FROM radnik;
```

Funkcija ROUND – Zadatak za vežbu

- **ROUND(<izraz>, <broj_decimala>)** - vraća zaokruženu vrednost datog <izraza> na dati <broj_decimala>

- Prikazati prosečnu platu svih radnika pomnoženu sa koren iz 2 (1,41) zaokruženo na dve decimale.

```
SELECT ROUND(AVG(plt*1.41), 2)
FROM radnik;
```

Skupovne funkcije nad isključivo NULL vrednostima

- Prikazati ukupnu premiju svih radnika čiji je matični broj veći od 100.

```
SELECT SUM(pre)
FROM radnik
WHERE mbr > 100;
```

- Šta je rezultat SUM, AVG, MAX, MIN funkcija kada su u skupu sve null vrednosti?
  - Rezultat je null.
  - Rezultat COUNT funkcije u tom slučaju je 0.

GROUP BY

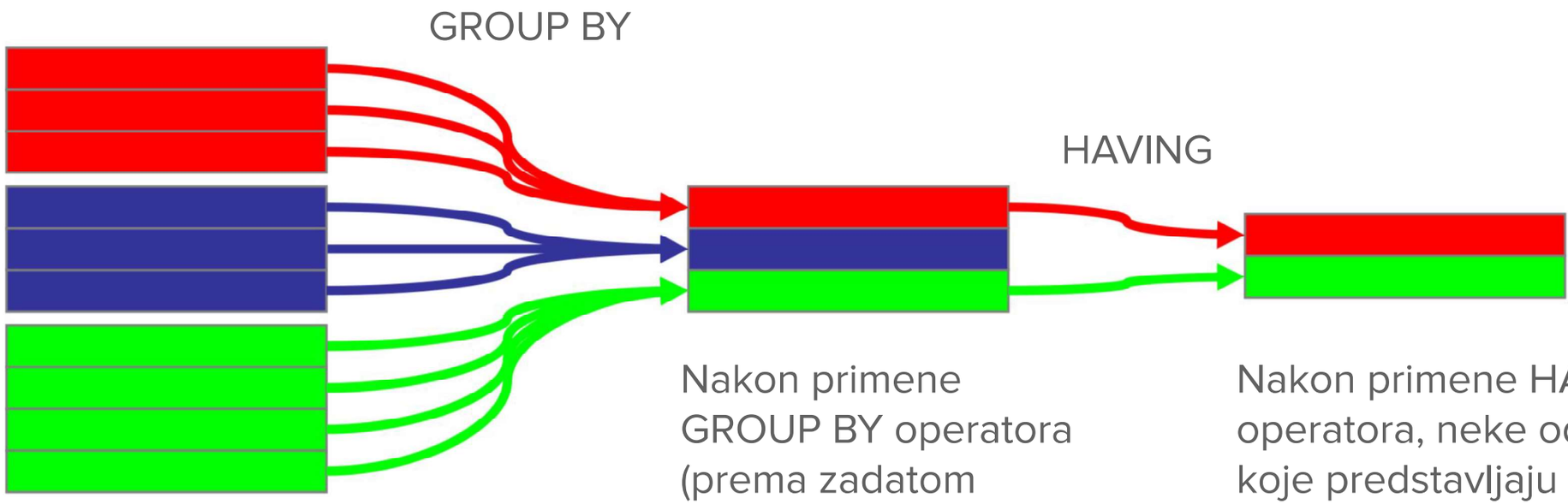
```
SELECT mbr, spr
FROM radproj
WHERE mbr < 40;
```


mbr	spr
10	10
20	20
10	30
30	30
30	40

```
SELECT mbr, count(spr)
FROM radproj
WHERE mbr < 40
GROUP BY mbr;
```


mbr	count(spr)
30	2
20	1
10	2

GROUP BY i HAVING

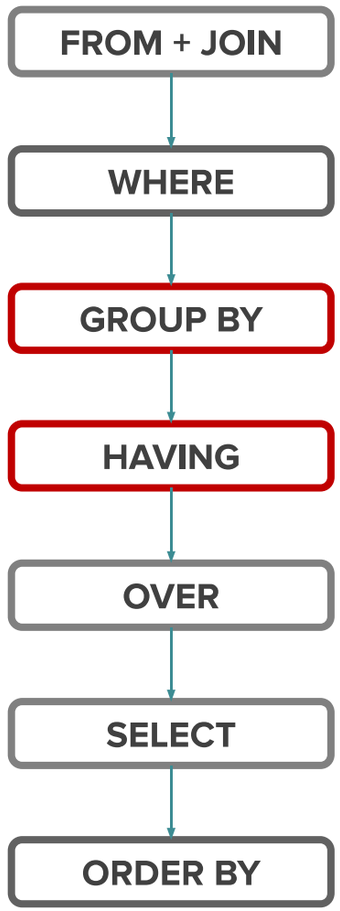


Pojedinačne torke - prikazane prema pripadnosti grupi

Nakon primene GROUP BY operatora (prema zadatom izrazu) dobija se po jedna torka za svaku grupu

Nakon primene HAVING operatora, neke od torki koje predstavljaju rezultat grupisanja mogu biti odbačene - na osnovu izraza iz HAVING-a

Redosled izvršavanja klauzula



GROUP BY – Zadatak za vežbu

- Prikazati koliko radnika radi na svakom projektu i koliko je ukupno angažovanje na tom projektu?

```
SELECT spr, COUNT(mbr), SUM(brc)
FROM radproj
GROUP BY spr;
```

HAVING – Zadatak za vežbu

- Izlistati mbr radnika koji rade na više od dva projekta, pored mbr-a, prikazati i broj projekata na kojima radnici rade.

```
SELECT mbr, COUNT(spr)
FROM radproj
GROUP BY mbr
HAVING COUNT(spr) > 2;
```

GROUP BY – napomene

- Najčešće se koristi u kombinaciji sa skupovnim funkcijama (MIN, MAX, COUNT, AVG...)
- Svaka kolona koja se nađe među izrazima SELECT klauzule, osim onih kolona koji su pod skupovnom funkcijom se mora naći i u izrazima GROUP BY klauzule
  - Npr. COUNT(spr) se može naći u izrazima u SELECT klauzuli, a spr se ne mora naći naveden u izrazima koji pripadaju GROUP BY klauzuli
    - Ovakva upotreba i jeste najčešća
- Grupe se mogu filtrirati korišćenjem HAVING ključne reči
  - WHERE filtira torke, ne grupe
- Može se koristiti u kombinaciji sa ORDER BY