

# Baze podataka 1

---

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	Kod kuće
<b>Username</b>	indXY <sup>1</sup>	indXY <sup>1</sup>	*2
<b>Password</b>	ftn	ftn	*2
<b>Role</b>	default	default	default
<b>Host Name</b>	192.168.18.9	192.168.7.204	localhost
<b>Port</b>	1522	1521	1521
<b>Oracle SID</b>	db2016	bp1	
<b>Service name</b>			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)
  - 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]  $\subseteq$  Radnik[Mbr],
  - Radproj[Spr]  $\subseteq$  Projekat[Spr],
- 
- Projekat[Ruk]  $\subseteq$  Radnik[Mbr],
  - Null(Projekat, Ruk) =  $\perp$
- 
- Radnik[Sef]  $\subseteq$  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_ogranicenja> [, ...]);
```

- 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]  $\subseteq$  projekat[Spr],
    - faze\_projekta[Rukfp]  $\subseteq$  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 <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 važno 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;
```

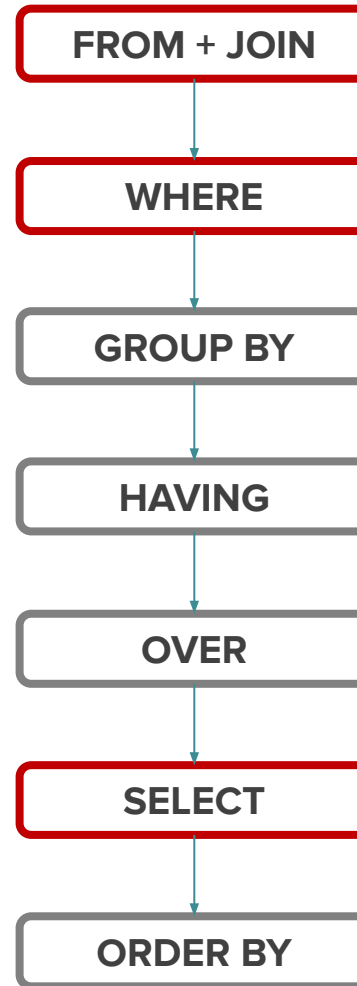
## WHERE <uslov\_selekcije> – Zadatak za vežbu

```
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.

```
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.

```
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');
```

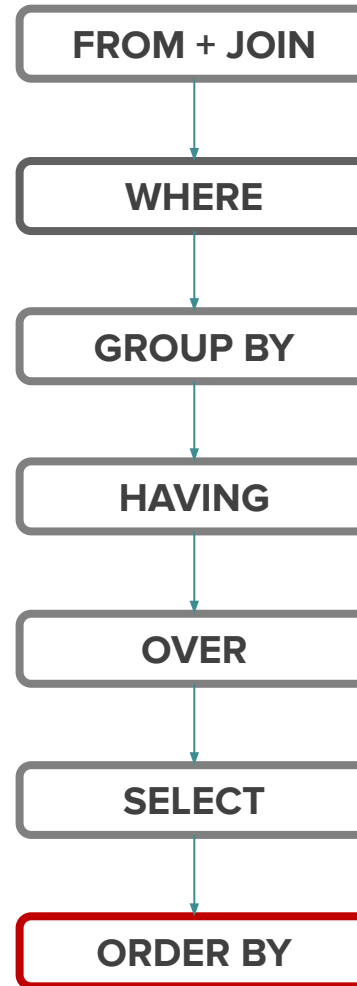
# Uređivanje izlaznih rezultata – ORDER BY – Zadatak za vežbu

```
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
- 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

```
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  $\Theta$  ANY (<lista\_vrednosti>)  
 $\Theta \in \{<, >, <=, >=, !=, =\}$

- 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  $\Theta$  ALL (<lista\_vrednosti>)  
 $\Theta \in \{<, >, <=, >=, !=, =\}$

- Primer:

x  $\neq$  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  $\neq$  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 mesečna 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;
```



# 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.


# 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;
```


# GROUP BY

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



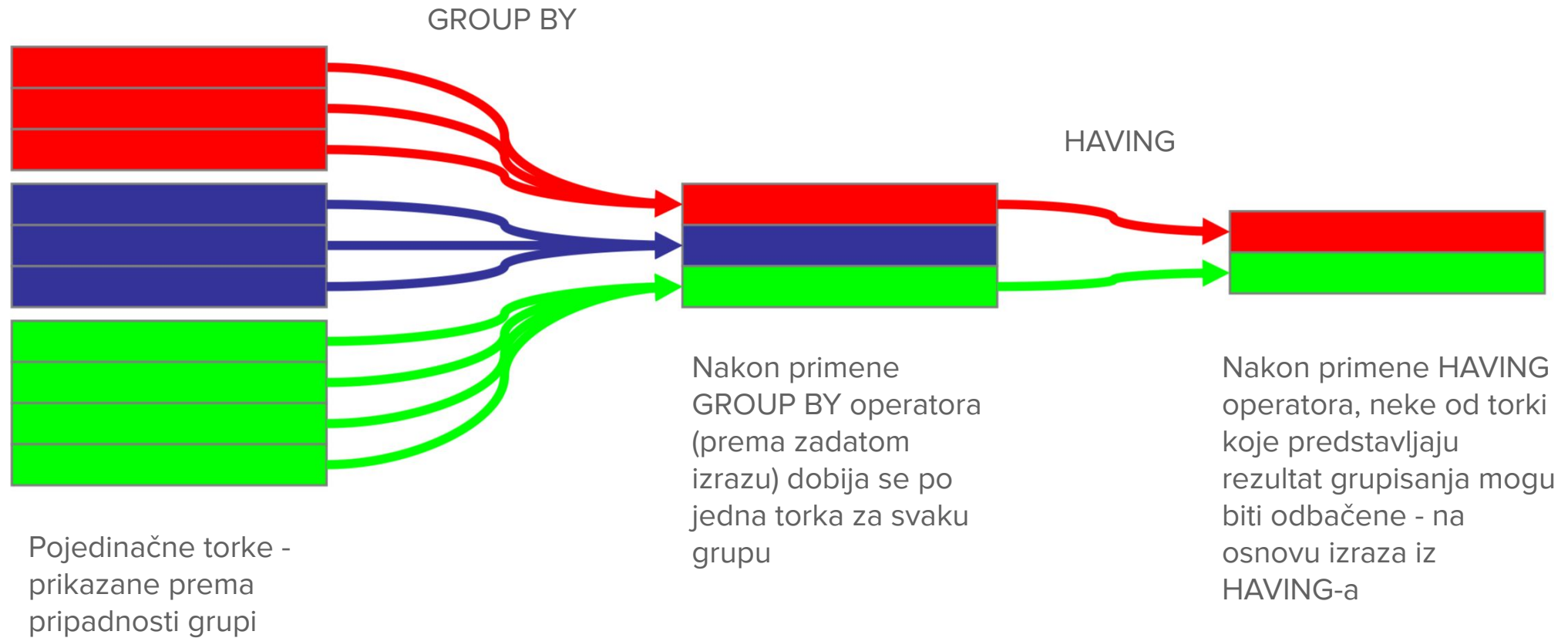
mbr	spr
10	10
20	20
10	30
30	30
30	40

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

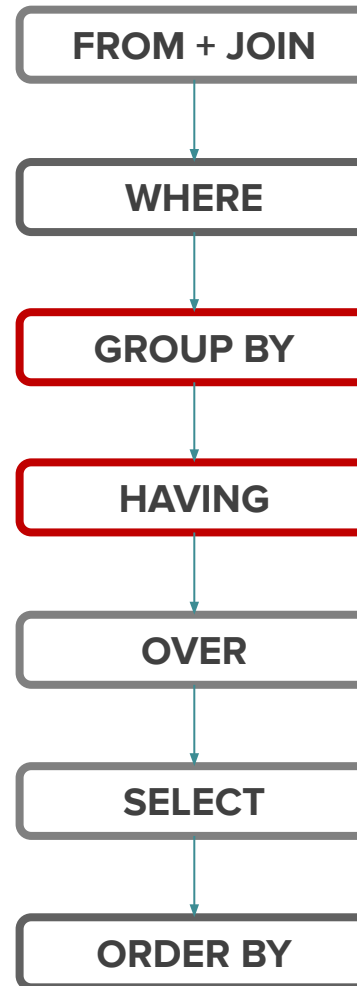


mbr	count(spr)
30	2
20	1
10	2

# GROUP BY i HAVING



## 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 filtrira torke, ne grupe
- Može se koristiti u kombinaciji sa ORDER BY



# Nezavisni ugnježdjeni upiti

- SELECT unutar WHERE druge SELECT naredbe
- Predikatski izrazi:
  - ANY, ALL, IN i EXISTS
- SQL dozvoljava višestruko ugnježdavanje upita

- Izlistati u rastućem redosledu plate mbr, ime, prz i plt radnika koji imaju platu veću od prosečne.

```
SELECT mbr, ime, prz, plt
FROM radnik
WHERE plt > (SELECT AVG(plt) FROM radnik)
ORDER BY plt ASC;
```

# Nezavisni ugnježdjeni upiti

- Izlistati mbr, ime, prz radnika koji rade na projektu sa šifrom 10, a ne rade na projektu sa šifrom 30.

```
SELECT mbr, ime, prz
FROM radnik
WHERE mbr IN
    (SELECT mbr FROM radproj WHERE spr = 10)
AND mbr NOT IN
    (SELECT mbr FROM radproj WHERE spr = 30);
```

- Zašto ne može u jednom ugnježđenom upitu?

# Domaći zadatak

- Izlistati ime, prz i god najstarijeg radnika.

```
SELECT mbr, ime, prz, god
FROM radnik
WHERE god <= all(SELECT god FROM radnik);
```

```
SELECT mbr, ime, prz, god
FROM radnik
WHERE god = (SELECT MIN(god) FROM radnik);
```

# Zadatak za vežbu

- Izlistati nazive projekata na kojima radi bar jedan radnik koji radi i na projektu sa šifrom 60.

```
SELECT p.nap
FROM projekat p
WHERE spr in (SELECT spr
               FROM radproj
               WHERE mbr IN (SELECT mbr
                             FROM radproj
                             WHERE spr=60));
```

# SELECT naredba u listi tabela

```
SELECT *  
FROM (SELECT mbr, ime FROM radnik);
```

# ROWNUM – Zadatak za vežbu

- Prikazati 10 radnika koji imaju najveću platu, sortiranih po plati u opadajućem redosledu.

```
SELECT mbr, plt, ROWNUM  
FROM radnik  
WHERE ROWNUM <= 10  
ORDER BY plt DESC;
```

- Da li je ovo ispravno?

# ROWNUM – Zadatak za vežbu

- **ROWNUM** je vrednost koju torka dobija na osnovu redosleda kojim Oracle dobavlja torke
  - Vrednost kreće od 1
  - Predstavlja pseudokolonu
- 
- Prikazati 10 radnika koji imaju najveću platu, sortiranih po plati u opadajućem redosledu.

```
SELECT mbr, plt, ROWNUM  
FROM (SELECT * FROM radnik ORDER BY plt DESC)  
WHERE ROWNUM <= 10;
```

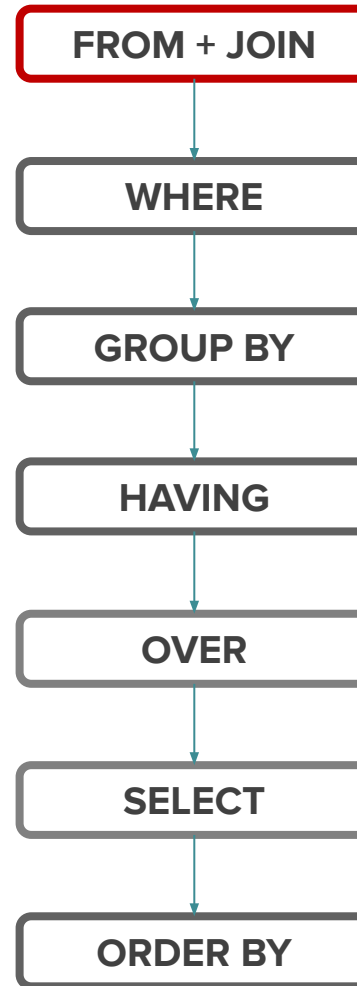
# Spajanje tabela

- Prikazati mbr, prz, ime, plt i brc angažovanja svih radnika koji rade na projektu sa šifrom 10.

```
SELECT r.mbr, prz, ime, plt, brc  
FROM radnik r, radproj  
WHERE spr = 10 AND r.mbr = radproj.mbr;
```



## Redosled izvršavanja klauzula



# Spajanje tabela – Zadatak za vežbu

- Prikazati mbr, ime, prz i plt radnika koji su rukovodioci projekata.

```
SELECT DISTINCT mbr, ime, prz, plt  
FROM radnik, projekat  
WHERE ruk=mbr;
```

## Zadatak za vežbu

- Prikazati imena i prezimena rukovodilaca projekata i broj projekata kojima rukovode.

```
SELECT prz, ime, COUNT(spr)
FROM radnik r, projekat p
WHERE ruk=mbr
GROUP BY mbr, prz, ime;
```

## Zadatak za vežbu

- Prikazati za svakog radnika mbr, prz, ime, ukupan broj projekata i ukupno angažovanje na projektima na kojima radi.

```
SELECT r.mbr, r.prz, r.ime, COUNT(*), SUM(rp.brc)
FROM radnik r, radproj rp
WHERE r.mbr=rp.mbr
GROUP BY r.mbr, r.prz, r.ime;
```

# Spajanje tabela – Zadatak za vežbu

- Izlistati imena, prezimena svih radnika osim rukovodioca projekta sa šifrom 10.

```
SELECT mbr, ime, prz  
FROM radnik r, projekat p  
WHERE p.spr=10 AND r.mbr!=p.ruk;
```

```
SELECT ime, prz, mbr  
FROM radnik  
WHERE mbr != (SELECT ruk  
               FROM projekat  
               WHERE spr=10);
```

# Zadatak za vežbu

- Prikazati imena i prezimena rukovodilaca projekata i broj projekata na kojima rade.

```
SELECT ime, prz, count(rp.spr) bp  
FROM radnik r, radproj rp  
WHERE r.mbr = rp.mbr AND r.mbr IN (SELECT ruk FROM projekat)  
GROUP BY r.mbr, prz, ime;
```

```
SELECT ime,prz,count(DISTINCT rp.spr)  
FROM radnik r,projekat p, radproj rp  
WHERE rp.mbr = r.mbr AND p.ruk = r.mbr  
GROUP BY r.mbr,ime,prz;
```

# Zadatak za vežbu

- Izlistati nazive projekata na kojima se ukupno radi više od 15 časova.

```
SELECT nap  
FROM projekat p, radproj rp  
WHERE p.spr = rp.spr  
GROUP BY p.spr, nap  
HAVING SUM(brc) > 15;
```

# Domaći zadatak

- Izlistati šifre i nazive projekata na kojima radi više od dva radnika.

```
SELECT p.spr, p.nap  
FROM projekat p, radproj rp  
WHERE rp.spr=p.spr  
GROUP BY p.spr, p.nap  
HAVING COUNT(mbr) > 2;
```



# Zadatak za vežbu

- Izlistati nazive i šifre projekata na kojima je prosečno angažovanje veće od prosečnog angažovanja na svim projektima.

```
SELECT p.spr, p.nap  
FROM projekat p, radproj rp  
WHERE rp.spr=p.spr  
GROUP BY p.spr, p.nap  
HAVING AVG(brc) > (SELECT AVG(brc) FROM radproj);
```

# Zadatak za vežbu

- Izlistati nazive i šifre projekata sa najvećim prosečnim angažovanjem.

Primer:

SPR = 10

$$(10+11+9+2)/4 = 8$$

SPR = 20

$$(13+8+8+7)/4 = 9$$

SPR = 30

$$(4+3+4+5+9)/5 = 5$$

Upit bi u ovom slučaju trebalo da vrati projekat sa šifrom 20

```
SELECT p.spr, p.nap
FROM projekat p, radproj rp
WHERE rp.spr=p.spr
GROUP BY p.spr, p.nap
HAVING AVG(brc)>= ALL(SELECT AVG(brc) FROM radproj GROUP BY spr);
```

	SPR	MBR	BRC
1	10	10	10
2	10	50	11
3	10	100	9
4	10	130	2
5	20	20	13
6	20	70	8
7	20	110	8
8	20	120	7
9	30	10	4
10	30	30	3
11	30	50	5
12	30	60	4
13	30	80	9

## Upit sa višestrukom upotrebom iste tabele – Zadatak za vežbu

- Prikazati mbr, ime, prz, plt radnika koji zarađuju više od radnika sa matičnim brojem 40.

```
SELECT r.mbr, r.prz, r.ime, r.plt  
FROM radnik r, radnik r1  
WHERE r.plt > r1.plt AND r1.mbr = 40;
```

## Upit sa višestrukom upotrebom iste tabele – Domaći zadatak

- Prikazati imena, prezimena i plate radnika koji zarađuju bar 1000 dinara manje od rukovodioca projekta na kom radnik radi.

```
SELECT r1.ime, r1.prz, r1.plt, p.nap  
FROM radnik r1, radnik r2, projekat p, radproj rp  
WHERE r1.mbr = rp.mbr AND rp.spr = p.spr AND p.ruk = r2.mbr AND r1.plt + 1000 < r2.plt;
```

## Povezani upiti – Zadatak za vežbu

- Prikazati mbr, ime, prz, plt radnika čiji je broj sati angažovanja na nekom projektu veći od prosečnog broja sati angažovanja na tom projektu.

```
SELECT DISTINCT r.mbr, ime, prz, plt
FROM radnik r, radproj rp1
WHERE r.mbr = rp1.mbr AND rp1.brc > (SELECT AVG(brc)
                                     FROM radproj rp2
                                     WHERE rp2.spr=rp1.spr);
```

# EXISTS – Zadatak za vežbu

- **EXISTS(<lista\_vrednosti>)** - <lista\_vrednosti> nije prazan skup vrednosti
- **NOT EXISTS(<lista\_vrednosti>)** - <lista\_vrednosti> je prazan skup vrednosti

- Ko je najstariji radnik?

```
SELECT ime, prz, god
FROM radnik r
WHERE NOT EXISTS (SELECT mbr
                   FROM radnik r1
                   WHERE r1.god < r.god);
```

# EXISTS – Zadatak za vežbu

- Izlistati radnike koji ne rade ni na jednom projektu. (Ne postoji projekat na kom rade).

```
SELECT mbr, ime, prz
FROM radnik r
WHERE NOT EXISTS(SELECT *
                  FROM radproj rp
                  WHERE r.mbr = rp.mbr);
```

```
SELECT mbr, ime, prz
FROM radnik r
WHERE mbr not in (SELECT rp.mbr
                  FROM radproj rp);
```

# EXISTS – Domaći zadatak

- Izlistati mbr, ime, prz radnika koji ne rade na projektu sa šifrom 10 (ne postoji radnik sa projekta 10 koji je jednak traženom radniku).

```
SELECT mbr, ime, prz
FROM radnik r
WHERE NOT EXISTS(SELECT *
                  FROM radproj rp
                  WHERE r.mbr = rp.mbr AND rp.spr = 10);
```



# EXISTS – Domaći zadatak

- Izlistati radnike koji nisu rukovodioci projekata. (ne postoji projekat kojim rukovodi taj radnik)

```
SELECT mbr, ime, prz  
FROM radnik r  
WHERE NOT EXISTS(SELECT *  
                  FROM projekat  
                  WHERE mbr = ruk);
```

```
SELECT mbr, ime, prz  
FROM radnik r  
WHERE mbr not in (SELECT ruk  
                 FROM projekat);
```

## Povezani upiti – EXISTS – Zadatak za vežbu

- Ko je najmlađi rukovodilac projekata?

```
SELECT DISTINCT mbr, ime, prz, god
FROM radnik r, projekat p
WHERE r.mbr=p.ruk AND NOT EXISTS(SELECT mbr
                                FROM radnik r1, projekat p1
                                WHERE r1.mbr=p1.ruk AND r1.god>r.god);
```

## Unija (UNION) – Zadatak za vežbu

- Izlistati mbr, ime, prz radnika koji rade na projektu sa šifrom 20 ili im je plata veća od prosečne.

```
SELECT mbr, ime, prz
FROM radnik
WHERE mbr IN (SELECT mbr FROM radproj WHERE spr=20)
UNION
SELECT mbr, ime, prz
FROM radnik
WHERE plt > (SELECT AVG(plt) FROM radnik);
```

# Unija (UNION ALL)

- Izlistati mbr, ime, prz radnika koji rade na projektu sa šifrom 20 ili im je plata veća od prosečne.

```
SELECT mbr, ime, prz
FROM radnik
WHERE mbr IN (SELECT mbr FROM radproj WHERE spr=20)
UNION ALL
SELECT mbr, ime, prz
FROM radnik
WHERE plt > (SELECT AVG(plt) FROM radnik);
```

# Presek (INTERSECT)

- Izlistati mbr, ime, prz radnika čije prezime počinje na slovo M ili slovo R i mbr, ime, prz radnika čije prezime počinje na slovo M ili slovo P.

```
SELECT mbr, ime, prz
FROM radnik
WHERE prz LIKE 'M%' OR prz LIKE 'R%'
INTERSECT
SELECT mbr, ime, prz
FROM radnik
WHERE prz LIKE 'M%' OR prz LIKE 'P%';
```

# Razlika (MINUS)

- Izlistati mbr, ime, prz radnika čije prezime počinje na slovo M ili slovo R i mbr, ime, prz radnika čije prezime počinje na slovo M ili slovo P.

```
SELECT mbr, ime, prz
FROM radnik
WHERE prz LIKE 'M%' OR prz LIKE 'R%'
MINUS
SELECT mbr, ime, prz
FROM radnik
WHERE prz LIKE 'M%' OR prz LIKE 'P%';
```

# Prirodno spajanje (NATURAL JOIN)

- Spajanje se vrši na osnovu imena kolona
- Prikazati ime i prz radnika koji rade na projektu sa šifrom 30.

```
SELECT ime, prz  
FROM radnik NATURAL JOIN radproj  
WHERE spr=30;
```

# Unutrašnje spajanje (INNER JOIN)

- Prikazati ime i prz radnika koji rade na projektu sa šifrom 30.

```
SELECT ime, prz  
FROM radnik r INNER JOIN radproj rp ON r.mbr = rp.mbr  
WHERE spr=30;
```



# Spoljno spajanje (OUTER JOIN)

- Levo (LEFT)
- Desno (RIGHT)
- Potpuno (FULL)

# Spoljno spajanje (LEFT OUTER JOIN)

- Prikazati mbr, ime i prz radnika i šifre projekata na kojima rade. Prikazati, takođe, iste podatke i za radnike koji ne rade ni na jednom projektu, pri čemu za šifru projekta treba, u tom slučaju, prikazati nedostajuću vrednost.

```
SELECT r.mbr, ime, prz, spr  
FROM radnik r LEFT OUTER JOIN radproj rp ON r.mbr = rp.mbr;
```

## Spoljno spajanje (LEFT OUTER JOIN) – Zadatak za vežbu

- Prikazati mbr, ime i prz svih radnika i nazive projekata kojima rukovode. Ukoliko radnik ne rukovodi ni jednim projektom ispisati: ne rukovodi projektom.

```
SELECT r.mbr, ime, prz, NVL(nap, 'Ne rukovodi projektom') Projekat  
FROM radnik r LEFT OUTER JOIN projekat p ON r.mbr=p.ruk;
```

## Spoljno spajanje (RIGHT OUTER JOIN) – Zadatak za vežbu

- Prikazati nazive svih projekata i mbr radnika koji rade na njima. Ukoliko na projektu ne radi ni jedan radnik ispisati nulu umesto matičnog broja.

```
SELECT NVL(rp.mbr, 0) "Mbr radnika", nap  
FROM radproj rp RIGHT OUTER JOIN projekat p ON rp.spr=p.spr;
```

```
SELECT NVL(rp.mbr, 0) "Mbr radnika", nap  
FROM radproj rp, projekat p  
WHERE rp.spr(+) = p.spr;
```

## Spoljno spajanje (FULL OUTER JOIN)

```
SELECT NVL(rp.mbr, 0) "Mbr radnika", nap  
FROM radproj rp FULL OUTER JOIN projekat p ON rp.spr=p.spr;
```

## Zadatak za vežbu

- Prikazati za sve radnike i projekte na kojima rade mbr, prz, ime, spr i nap. Za radnike koje ne rade ni na jednom projektu, treba prikazati mbr, prz, ime, dok za vrednosti obeležja spr i nap treba zadati, redom, konstante 0 i "Ne postoji". Urediti izlazni rezultat saglasno rastućim vrednostima obeležja mbr.

```
SELECT r.mbr, r.prz, r.ime, NVL(p.spr, 0) AS spr, NVL(p.nap, 'Ne postoji') AS nap
FROM radnik r, radproj rp, projekat p
WHERE r.mbr = rp.mbr (+) AND rp.spr = p.spr (+)
ORDER BY mbr;
```

```
SELECT r.mbr, r.prz, r.ime, NVL(p.spr, 0) AS spr, NVL(p.nap, 'Ne postoji') AS nap
FROM radnik r LEFT OUTER JOIN radproj rp ON r.mbr=rp.mbr LEFT OUTER JOIN projekat p ON rp.spr=p.spr
ORDER BY mbr;
```

# Domaći zadatak

- Prikazati matične brojeve, imena i prezimena radnika, zajedno sa šiframa projekata na kojima rade. Prikazati, takođe, iste podatke i za radnike koji ne rade ni na jednom projektu, pri čemu za šifru projekta treba, u tom slučaju, prikazati nedostajuću vrednost.

```
SELECT r.mbr, r.prz, r.ime, rp.spr  
FROM radnik r, radproj rp  
WHERE r.mbr = rp.mbr(+);
```

```
SELECT r.mbr, r.prz, r.ime, rp.spr  
FROM radnik r LEFT OUTER JOIN radproj rp ON r.mbr = rp.mbr;
```

# Domaći zadatak

- Prikazati imena i prezimena svih radnika i prezimena njihovih šefova ako ih imaju. Ako nema šefa ispisati: Nema sefa.

```
SELECT r1.ime, r1.prz "Radnik", NVL(r2.prz, 'Nema sefa') Sef  
FROM radnik r1 LEFT OUTER JOIN radnik r2 ON r1.sef=r2.mbr  
ORDER BY r1.prz;
```



# Dekartov proizvod (CROSS JOIN)

- Koristi se ako želimo da napravimo Dekartov proizvod između dve tabele

```
SELECT *  
FROM radnik, projekat;
```

- Ekvivalentno je sa

```
SELECT *  
FROM radnik CROSS JOIN projekat;
```

- Može se dodati uslov na cross join, onda se ponaša kao inner join
- Često se zaborave uslovi spoja prilikom spajanja tabela, pa rezultat bude Dekartov proizvod torki iz spajajućih tabela

# Domaći zadatak

- Za svaku satnicu angažovanja (brc), prikazati koliko radnika radi na nekom projektu sa tom satnicom. Rezultate urediti u opadajućem redosledu satnice.

```
SELECT brc, COUNT(mbr)
FROM radproj GROUP BY brc
ORDER BY brc DESC;
```

# Domaći zadatak

- Za svakog radnika prikazati matični broj, ime, prezime, kao i broj projekata kojima rukovodi, pri čemu je potrebno prikazati isključivo one radnike koji su rukovodioci na manjem broju projekata od prosečnog broja projekata na kojima rade radnici čije se prezime ne završava na “ic”.

```
SELECT mbr, ime, COUNT(spr) br_pr_rukovodi
FROM radnik r LEFT OUTER JOIN projekat p on r.mbr=p.ruk
GROUP BY mbr, ime HAVING COUNT(spr) < (SELECT AVG(COUNT(spr))
                                         FROM radproj rp, radnik r
                                         WHERE rp.mbr = r.mbr
                                         AND prz NOT LIKE '%ic'
                                         GROUP BY r.mbr);
```

# Selekcionni izraz (CASE)

- Prosti (Simple) CASE:

```
CASE expr
  WHEN expr1 THEN return_expr1
  [ WHEN expr2 THEN return_expr2
  WHEN exprn THEN return_exprn]
  [ ELSE else_expr ]
END;
```

- Pretražujući (Searched) CASE:

```
CASE
  WHEN comparison_expr1 THEN return_expr1
  [ WHEN comparison_expr2 THEN return_expr2
  WHEN comparison_exprn THEN return_exprn]
  [ ELSE else_expr ]
END;
```

# Zadatak za vežbu

- Za svakog radnika prikazati mbr, ime, prz, kao kategoriju kojoj pripada na osnovu visine plate. Kategorije po visini plate su sledeće:
  - Plata manja od 10000 – kategorija: '**mala primanja**',
  - plata između 10000 i 20000 – kategorija: '**srednje visoka primanja**',
  - plata između 20000 i 40000 – kategorija: '**visoka primanja**',
  - plata veća od 40000 – kategorija: '**izuzetno visoka primanja**'.
- Takođe, radnike urediti prema kategoriji kojoj pripadaju, u redosledu od najniže ka najvišoj kategoriji po visini plate.

# Zadatak za vežbu

```
SELECT mbr, ime, plt,  
CASE  
    WHEN plt < 10000 THEN 'mala primanja'  
    WHEN plt >=10000 AND plt < 20000 THEN 'srednja primanja'  
    WHEN plt >=20000 AND plt < 40000 THEN 'visoka primanja'  
    ELSE 'izuzetno visoka primanja'  
END AS visina_primanja  
FROM radnik  
ORDER BY  
    CASE visina_primanja  
        WHEN 'izuzetno visoka primanja' THEN 1  
        WHEN 'visoka primanja' THEN 2  
        WHEN 'srednja primanja' THEN 3  
        ELSE 4  
    END DESC, plt ASC;
```

# Selekcionni izraz (CASE) – Napomene

- Može se iskoristiti gde god je dozvoljeno korišćenje izraza
  - Najčešće – u okviru liste kolona ili u okviru ORDER BY klauzule
- Ukoliko se ne iskoristi else, podrazumevana vrednost biće null
- Kod prostog CASE izraza, poređenje sa null vrednošću nije moguće – podrazumevano se koristi poređenje operatorom '=', pa je takav izraz uvek netačan

# Zadatak za vežbu

- Za svakog radnika ispisati mbr, ime, prz, platu i mbr šefa. Pri ispisu treba obezbediti da su radnici uređeni saglasno visini plate, od najviše ka najnižoj, pri čemu bi direktor firme trebalo da se ispiše prvi.

```
SELECT mbr, ime, plt, sef
FROM radnik
ORDER BY
CASE
    WHEN sef IS NULL THEN 1
    ELSE 2
END, plt DESC;
```



# Klauzula WITH

- CTE (engl. *Common Table Expressions*)
- Dodela naziva bloku podupita
- Blok može biti referenciran više puta unutar upita
  - najveća razlika u odnosu na SELECT u listi tabela
- Uvodi svojstvo sastavljanja (eng. *Composability*) u SQL
- Optimizacija upita
  - kao privremena tabela/umetnuti pogled
- Sintaksa

**WITH naziv\_upita AS (SELECT...);**

# WITH – Primer

- Prikazati za svakog radnika angažovanog na projektu mbr, prz, ime, spr i broj drugih radnika koji su angažovani na istom projektu.

```
SELECT r.mbr, r.ime, r.prz, rp1.spr, COUNT(rp2.mbr)-1 ostali
FROM radnik r, radproj rp1, radproj rp2
WHERE r.mbr=rp1.mbr AND rp1.spr=rp2.spr
GROUP BY r.mbr, r.ime, r.prz, rp1.spr;
```

```
WITH projinfo AS (
    SELECT rp.spr, COUNT(rp.mbr) AS rad_broj
    FROM radproj rp GROUP BY rp.spr)
SELECT r.mbr, r.ime, r.prz, rp.spr, pi.rad_broj-1 ostali
FROM radnik r, radproj rp, projinfo pi
WHERE r.mbr=rp.mbr AND rp.spr=pi.spr;
```

## WITH – Zadatak za vežbu

- Prikazati za svakog radnika angažovanog na projektu mbr, prz, ime, spr i udeo u ukupnom broju časova rada na tom projektu (zaokruženo na dve decimale)

```
WITH projinfo AS (  
    SELECT rp.spr, SUM(rp.brc) AS cas_suma  
    FROM radproj rp  
    GROUP BY rp.spr)  
SELECT r.mbr, r.ime, r.prz, rp.spr, ROUND(rp.brc/pi.cas_suma, 2) udeo  
FROM radnik r, radproj rp, projinfo pi  
WHERE r.mbr=rp.mbr AND rp.spr=pi.spr;
```

## WITH – Zadatak za vežbu

- Prikazati mbr, ime i prz rukovodilaca projekata kao i ukupan broj radnika kojima rukovode na projektima

```
WITH rukovodilac AS (  
    SELECT mbr, ime, prz, plt, spr  
    FROM radnik, projekat WHERE mbr=rak),  
projinfo AS (  
    SELECT spr, count(mbr) ljudi  
    FROM radproj GROUP BY spr)  
SELECT ru.mbr, ru.ime, ru.prz, SUM(pi.ljudi) ljudi  
FROM rukovodilac ru, projinfo pi  
WHERE ru.spr=pi.spr  
GROUP BY ru.mbr, ru.ime, ru.prz;
```

## WITH – Domaći zadatak

- Prikazati mbr, ime, prz, plt radnika čiji je broj sati angažovanja na nekom projektu veći od prosečnog broja sati angažovanja na tom projektu

```
WITH projinfo AS (  
    SELECT spr, AVG(brc) prosek  
    FROM radproj GROUP BY spr)  
SELECT distinct r.mbr, r.ime, r.prz, r.plt  
FROM radnik r, radproj rp, projinfo pi  
WHERE r.mbr=rp.mbr AND rp.spr=pi.spr AND rp.brc > pi.prosek;
```

## WITH – Domaći zadatak

- Koliko je ukupno angažovanje svih šefova na projektima?

```
WITH angaz_po_radnicima (mbr, sbrc) AS (  
    SELECT r.mbr, nvl(SUM(rp.brc), 0)  
    FROM radnik r, radproj rp  
    WHERE r.mbr = rp.mbr (+)  
    GROUP BY r.mbr),  
angaz_sefova (mbr, prz, ime, brrad, brsat) AS (  
    SELECT distinct r.sef, r1.prz, r1.ime, count(*), a.sbrc  
    FROM radnik r, radnik r1, angaz_po_radnicima a  
    WHERE r.Sef = r1.Mbr AND r.Sef = a.Mbr  
    GROUP BY r.Sef, r1.Prz, r1.Ime, a.SBrc)  
SELECT SUM(brsat) AS ukangsef  
FROM angaz_sefova;
```

# Pogledi

- Podupiti mogu se trajno sačuvati kao pogledi
  - podupiti koji se koriste kod CTE mogu se koristiti samo dok traje naredba
- Pogodnosti koje pruža korišćenje pogleda:
  - ograničavaju pristup bazi podataka
  - obezbeđuju nezavisnost podataka
  - obezbeđuju višestruke poglede nad istim podacima
  - mogu se brisati bez uklanjanja podataka u osnovnim tabelama
- CTE se koriste za jednokratne upite, dok se pogledi koriste kada se isti podupit često koristi

# Kreiranje, izmena i brisanje definicije pogleda

```
CREATE [OR REPLACE] VIEW <naziv_pogleda> [(alias [, alias]...)]  
AS podupit;
```

- Podupit koji se koristi za definisanje pogleda može biti kompleksan



# Modifikacija pogleda

- Pogledi se modifikuju pomoću OR REPLACE opcije (kreira se pogled, a ako pogled sa tim imenom već postoji, nova definicija zamenjuje staru).
- Dakle, pogled može biti izmenjen bez brisanja postojećeg pogleda.
- Na primer, mogu se dodati alijasi za kolone u pogledu.

# Kreiranje složenog pogleda

- Ukoliko se u upitu pomoću kog se kreira pogled nalaze skupovne funkcije (min, max, avg, sum, count) ili izrazi, u pogledu se moraju definisati alternativna imena za te kolone.

# Brisanje pogleda

```
DROP VIEW pogled;
```

# Pogledi – Zadatak za vežbu

- Napraviti pogled koji će za sve radnike prikazati samo njihova imena, prezimena i platu.

```
CREATE OR REPLACE VIEW plate_radnika (Ime, Prezime, Plata) AS  
    SELECT Ime, Prz, Plt  
    FROM radnik;
```

## Pogledi – Zadatak za vežbu

- Napraviti pogled koji će za sve radnike prikazati Mbr i ukupan broj sati angažovanja radnika na projektima na kojima radi.

```
CREATE OR REPLACE VIEW angaz_po_radnicima (Mbr, SBrc) AS
  SELECT r.Mbr, NVL(SUM(rp.Brc), 0)
  FROM radnik r, radproj rp
  WHERE r.Mbr = rp.Mbr (+)
  GROUP BY r.Mbr;
```

## Pogledi – Zadatak za vežbu

- Napraviti pogled koji će za svakog šefa (rukovodioca radnika) prikazati njegov matični broj, prezime, ime, ukupan broj radnika kojima šefuje i njegovo ukupno angažovanje na svim projektima, na kojima radi. Koristiti prethodno definisani pogled.

```
CREATE VIEW angaz_sefova (Mbr, Prz, Ime, BrRad, BrSat) AS
SELECT r.Sef, r1.Prz, r1.Ime, COUNT(*), a.SBrc
FROM radnik r, radnik r1, angaz_po_radnicima a
WHERE r.Sef = r1.Mbr AND r.Sef = a.Mbr
GROUP BY r.Sef, r1.Prz, r1.Ime, a.SBrc;
```

# Pogledi – Zadatak za vežbu

- Koliko je ukupno angažovanje svih šefova na projektima?

```
SELECT SUM(BrSat) AS UkAngSef  
FROM angaz_sefova;
```

# Sekvencer

- Generator sekvence vrednosti
- Automatski generiše jedinstvene brojeve
- Najčešće se koristi za kreiranje primarnih ključeva (surogat ključevi)
- Sekvenca se generiše i čuva nezavisno od tabele, tako da se jedna sekvenca može koristiti za više tabela

```
CREATE SEQUENCE sequence  
[INCREMENT BY n]  
[START WITH n]  
[{MAXVALUE n | NOMAXVALUE}]  
[{MINVALUE n | NOMINVALUE}]  
[{CYCLE | NOCYCLE}]  
[{CACHE n | NOCACHE}]
```

```
ALTER SEQUENCE sequence ...
```

```
DROP SEQUENCE sequence ...
```



# Sekvencer – Primer

```
CREATE SEQUENCE SEQ_Mbr  
INCREMENT BY 10  
START WITH 240  
NOCYCLE  
CACHE 10;
```

```
INSERT INTO radnik (Mbr, Prz, Ime, God)  
VALUES (SEQ_Mbr.NEXTVAL, 'Misic', 'Petar', SYSDATE);
```

```
SELECT SEQ_Mbr.CURRVAL  
FROM SYS.DUAL;
```

# Tabele u *Oracle* bazi podataka

- Korisničke tabele
  - kolekcije tabela koje kreira i održava korisnik
  - sadrže korisničke informacije
- **Data Dictionary** (rečnik podataka)
  - kolekcija tabela koje kreira i održava Oracle server
  - sadrže informacije baze podataka
  - vlasnik svih tabela u rečniku je SYS korisnik
  - informacije smeštene u rečniku podataka obuhvataju imena korisnika Oracle servera, privilegije dodeljene korisnicima, nazive objekata baze podataka, ograničenja.
  - postoji nekoliko kategorija pogleda rečnika podataka; svaka od njih ima odgovarajući prefiks:
    - USER\_ - ovi pogledi sadrže informacije o objektima čiji je vlasnik korisnik
    - ALL\_ - ovi pogledi sadrže informacije o svim tabelama (objektnim i relacionim) koje su dostupne korisniku
    - DBA\_ - ovi pogledi su zabranjeni, tj. dostupni su samo korisnicima koji imaju DBA ulogu

# Tabele u *Oracle* bazi podataka

- Upiti u rečniku podataka se postavljaju kao i svi ostali upiti.

- Prikazati nazive tabela čiji je vlasnik korisnik.

```
SELECT table_name FROM user_tables;
```

- Prikazati različite tipove objekata čiji je vlasnik korisnik.

```
SELECT DISTINCT object_type FROM user_objects;
```

- Prikazati tabele, pogledе, sinonime i sekvence čiji je vlasnik korisnik.

```
SELECT * FROM user_catalog;
```

# Karakter funkcije

- **LOWER(char)** – za konvertovanje svih znakova u mala slova
- **UPPER(char)** – za konvertovanje svih znakova u velika slova
- **INITCAP(char)** – prvo slovo svake reči u nizu znakova pretvara u veliko slovo, a ostatak reči u mala slova
- **SUBSTR(char, m [,n])** – koristi se za izdvajanje dela niza znakova
- **TRIM(LEADING | TRAILING | BOTH trim\_character FROM trim\_source)** – uklanja početne ili prateće znakove sa početka ili kraja niza znakova
- **LENGTH(char)** – vraća broj znakova u nizu

# Karakter funkcije – Primer

`LOWER('Sva mala slova') → 'sva mala slova'`

`UPPER('Sva velika slova') → 'SVA VELIKA SLOVA'`

`INITCAP('Velika početna slova') → ' Velika Početna Slova'`

`SUBSTR('DobroJutro', 1, 5) → 'Dobro'`

`TRIM('D' FROM 'DobroJutro') → 'obroJutro'`

`LENGTH('DobroJutro') → 10`

# Karakter funkcije – Primer

```
SELECT Mbr, Prz, Ime  
FROM Radnik  
WHERE UPPER(Prz) = 'PETRIC';
```

# Karakter funkcije – Domaći zadatak

- Prikazati radnike čije prezime na početku sadrži prva 3 slova imena, na primer: Petar Petric

```
SELECT *  
FROM radnik  
WHERE prz LIKE  
SUBSTR(IME, 0, 3) || '%';
```

## Karakter funkcije – Domaći zadatak

- Prikazati imena i prezimena radnika tako da se sva imena koja imaju poslednje slovo 'a', prikazuju bez poslednjeg slova.

```
SELECT TRIM(TRAILING 'a' FROM ime)  
FROM radnik;
```



# Karakter funkcije – Primer

- Svim radnicima promeniti ime tako da poslednje slovo bude uvećano. Primer: Ana -> AnA, Marko -> MarkO

`UPDATE` radnik

`SET ime = SUBSTR(ime, 1, LENGTH(ime)-1) || UPPER(SUBSTR(ime, LENGTH(ime), 1));`

# Karakter funkcije – Domaći zadatak

- Prikazati matične brojeve, spojena (konkatenirana) imena i prezimena radnika, kao i plate, uvećane za 17%.

```
SELECT Mbr,  
       Ime || ' ' || Prz "Ime i prezime",  
       Plt * 1.17 Plata  
FROM Radnik;
```

# Karakter funkcije – Domaći zadatak

- Prikazati radnike čije prezime sadrži ime. Na primer Marko Marković, ili Djordje Karadjordjevic

```
SELECT *  
FROM radnik  
WHERE LOWER(prz) LIKE '%' || LOWER(ime) || '%';
```

# Fukcije za konverziju podataka

- **TO\_CHAR(d [, fmt])** – transformiše vrednosti tipa DATE u VARCHAR2, po izboru uz navedeni format datuma
- **TO\_CHAR(n [, fmt])** – transformiše vrednost brojčanog tipa u VARCHAR2, po izboru uz navedeni format broja
- **TO\_DATE(char [, fmt])** – za konvertovanje niza znakova u ekvivalentni datum
- **TO\_NUMBER(char [,fmt])** – za konvertovanje znakovnih vrednosti u numeričke

## Fukcije za konverziju podataka – Zadatak za vežbu

- Za svakog radnika prikazati ime, prz, i projekte na kojima radi. Ako ne radi ni na jednom projektu, napisati 'Ne radi na projektu'. Imena radnika prikazati velikim slovima, a prezimena malim.

```
SELECT UPPER(ime), LOWER(prz), NVL(TO_CHAR(spr), 'Ne radi na projektu') broj_proj  
FROM radnik LEFT OUTER JOIN radproj on radnik.mbr = radproj.mbr;
```

# Funkcije za konverziju podataka – Primer

- Za svakog radnika prikazati datum rođenja u formatu yyyy/mm/dd.

```
SELECT TO_CHAR(god, 'yyyy/mm/dd')  
FROM radnik;
```

# Tabela isplate\_radnicima

Relacioni model:

- Isplate\_radnicima({Mbr, Mesec, Godina, Datum\_isplate, Iznos, Razlog\_isplate}, {Mbr, Datum\_isplate, Razlog\_isplate})
- Isplate\_radnicima[Mbr]  $\subseteq$  Radnik[Mbr],

Opis:

- Mbr - matični broj radnika
- Mesec - naziv meseca kada je isplata izvršena
- Godina - godina kada je isplata izvršena
- Datum\_isplate - datum kada je isplata izvršena
- Iznos - koliko je sredstava isplaćeno
- Razlog\_isplate - razlog zbog kog je vršena isplata

Dodatne napomene:

- Nijedno obeležje ne sme imati null vrednost.
- Razlog\_isplate može uzimati jednu od sledećih vrednosti:
  - 'PLATA\_DEO1', 'PLATA\_DEO2', 'PUTNI\_TROSKOVI', 'BONUS', 'DNEVNICA'

# Analitičke funkcije

- Kod agregacionih funkcija i GROUP BY klauzule, na izlazu operacije grupisanja se za svaku grupu (podskup redova) dobija po jedan red.

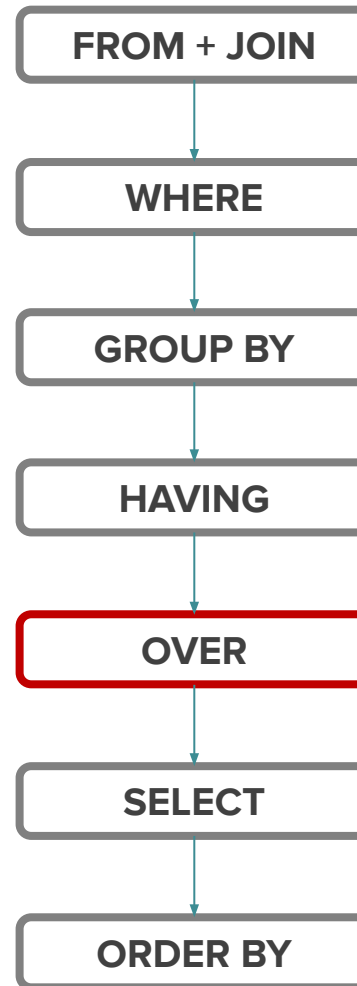
```
SELECT rp.spr, AVG(rp.brc)
FROM radproj rp
GROUP BY rp.spr;
```

- Kod analitičkih funkcija, izračunavanja se takođe vrše na nivou podskupa redova, ali se njihovom primenom ne smanjuje broj redova na izlazu.

```
SELECT r.mbr, r.ime, r.prz, rp.spr, rp.brc, AVG(rp.brc) OVER() AS prosek_brc_ukupni
FROM radnik r INNER JOIN radproj rp ON r.mbr=rp.mbr;
```



## Redosled izvršavanja klauzula



# Analitičke funkcije

`analitička_funkcija([arg]) OVER (analitički_uslovi)`

- analitička\_funkcija – AVG, MAX, MIN, COUNT, ROW\_NUMBER, ...
- analitički\_uslovi
  - sastoje se od sledećih delova:
    - `[ uslov_particionisanja ] [ uslov_uređivanja ] [uslov_odabira_torki]`
      - Uslov particionisanja definiše uslov za podelu ulaznog skupa torki u particije
      - Uslov uređivanja koristi se za uređivanje redova unutar definisanih particija
        - Bitno ako je zadata analitička funkcija osetljiva na redosled redova
      - Ukoliko se nijedan deo ne zada (prazan OVER), svi ulazni redovi biće deo iste particije

# Analitičke funkcije – Primer

- Prikazati mbr, ime, prz radnika angažovanih na projektima. Pored toga, prikazati spr i brc projekata na kojima su angažovani, kao i prosečno angažovanje na tom projektu.

```
SELECT r.mbr, r.ime, r.prz, rp.spr, rp.brc,  
       AVG(rp.brc) OVER (PARTITION BY rp.spr) AS prosek_brc_za_projekat  
FROM radnik r INNER JOIN radproj rp ON r.mbr=rp.mbr;
```

# Kumulativni zbir – Primer

- Prikazati mbr, datum isplate, razlog isplate, isplaćeni iznos, kao i kumulativnu sumu isplaćenog iznosa od početka 2023. godine za radnika sa matičnim brojem 70.

```
SELECT mbr, datum_isplate, razlog_isplate, iznos,  
       SUM(iznos) OVER (ORDER BY datum_isplate) AS kumulativni_zbir  
FROM isplate_radnicima  
WHERE mbr = 70 AND godina = 2023  
ORDER BY datum_isplate;
```

# Analitičke funkcije – Kumulativni zbir

- Ukoliko se u analitičkim uslovima zada uslov uređivanja, zadata funkcija se računa kumulativno, saglasno uređenju redova.
- Može se eksplicitno definisati koji bi redovi trebalo da se uzmu u obzir prilikom ovakvog računanja
  - **uslov\_odabira\_torki** (engl. *windowing clause*)
  - RANGE BETWEEN početna\_tacka AND krajnja\_tacka
  - ROWS BETWEEN početna\_tacka AND krajnja\_tacka
  - Moguće vrednosti za početnu i krajnju tacku:
    - UNBOUNDED PRECEDING (samo početna)
    - UNBOUNDED FOLLOWING (samo krajnja)
    - CURRENT ROW

## Kumulativni zbir – Primer

- Prikazati mbr, ime, prz radnika angažovanih na projektima. Pored toga, prikazati spr i brc projekata na kojima su angažovani, kao i kumulativnu sumu broja sati rada za radnike uređene od najmlađeg do najstarijeg.

```
SELECT r.mbr, r.ime, r.prz, rp.spr, rp.brc,  
       SUM(rp.brc) OVER(partition by rp.spr ORDER BY god DESC  
                        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS kumulativni_zbir  
FROM radnik r INNER JOIN radproj rp ON r.mbr=rp.mbr;
```

# Top n na nivou svake particije – Primer

- Za svaki projekat prikazati podatke o top 2 radnika sa najviše sati angažovanja.

```
WITH uredjeni_radnici AS (  
    SELECT mbr, spr, brc, row_number() OVER(PARTITION BY spr ORDER BY brc DESC) AS rbr  
    FROM radproj  
    ORDER BY spr  
)  
SELECT r.mbr, r.ime, r.prz, ur.spr, ur.brc, ur.rbr  
FROM uredjeni_radnici ur inner join radnik r ON r.mbr=ur.mbr  
WHERE rbr <= 2  
ORDER BY ur.spr, ur.rbr;
```

# Funkcija row\_number

- Ponašanje nalik onom viđenom kod ROWNUM pseudokolone
- Dodeljuje redne brojeve redovima unutar svake particije
- Funkcija je osetljiva na redosled redova
  - Izbacuje grešku ako se ne iskoristi uslov uređivanja
- Ako se ne navede uslov particionisanja u OVER, svi redovi će dobiti jedinstveni redni broj



## Zadatak za vežbu

- Za radnike koji imaju šefa ispisati mbr, ime, prz i rang po opadajućem iznosu plate u okviru istog nadređenog (šefa). Koristiti analitičku funkciju rank().

```
SELECT mbr, ime, prz, sef, plt,  
       rank() OVER (PARTITION BY sef ORDER BY plt DESC) "Rang"  
FROM radnik  
WHERE sef IS NOT NULL;
```

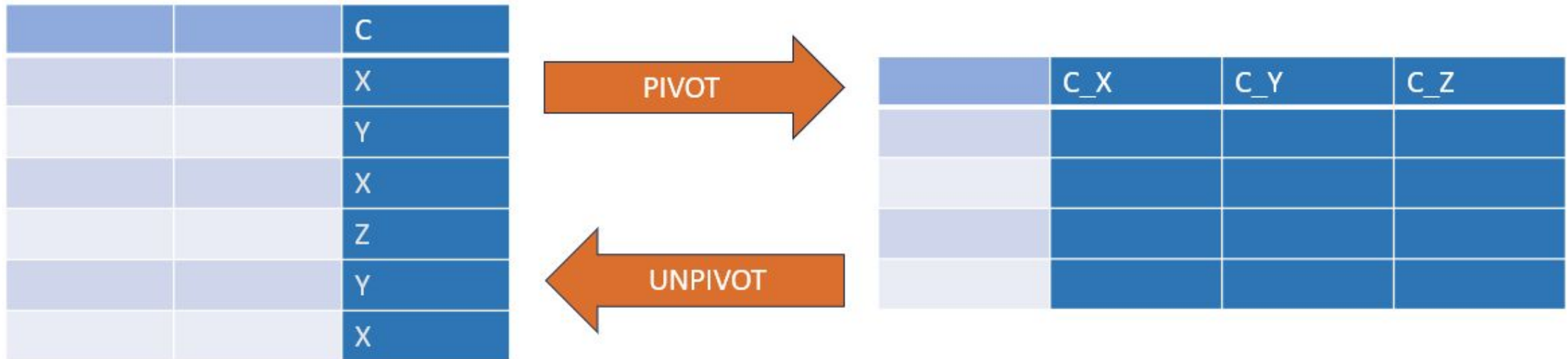
## Zadatak za vežbu

- Za radnike angažovane na projektima ispisati mbr, ime, prz, spr i udeo broja časova njihovog angažmana u ukupnom broju časova na projektu.

```
SELECT r.mbr, ime, prz, spr,  
       ROUND(brc/(SUM(brc) OVER (PARTITION BY spr)), 2) "Udeo"  
FROM radnik r JOIN radproj rp ON r.mbr = rp.mbr;
```

# Pivot i Unpivot

- Klauzule koje se mogu koristiti za “transponovanje” sadržaja tabela
  - Pivot omogućava transponovanje redova u kolone
    - “Uređeniji” prikaz agregiranih vrednosti u odnosu na GROUP BY
  - Unpivot omogućava transponovanje kolona u redove



# Pivot

```
SELECT lista_izraza
FROM lista_tabela
PIVOT (
    pozivi_agregacionih_funkcija_nad_kolonama
    FOR obeležje_pivotiranja
    IN (lista_vrednosti_obeležja_pivotiranja)
)
```

- **pozivi\_agregacionih\_funkcija\_nad\_kolonama** - prilikom transponovanja, vrši se agregiranje podataka
  - Implicitni GROUP BY **nad svim obeležjima koja nisu korišćena u klauzuli**
- **obeležje\_pivotiranja** - definiše obeležje na osnovu čijih vrednosti se vrši pivotiranje
- **lista\_vrednosti\_obeležja\_pivotiranja** - definiše listu vrednosti obeležja pivotiranja za koje bi trebalo da se izvrši transponovanje
  - Svaka vrednost u listi rezultovaće kreiranjem jedne ili više novih kolona
  - Kreira se nova kolona za svaku kombinaciju vrednosti obeležja pivotiranja i pozvanih agregacionih funkcija
    - Imena kolona se automatski određuju na osnovu kombinacije alijasa

# Pivot - Primer

- Za svakog radnika prikazati ukupan iznos isplaćen za svaki pojedinačni razlog isplate, kao i koliko je puta vršena isplata za svaki razlog isplate. Ukupan iznos isplate i broj isplata po razlozima prikazati u vidu kolona.

```
SELECT *  
FROM  
    (SELECT mbr, razlog_isplate, iznos FROM isplate_radnicima)  
PIVOT (  
    SUM(iznos) ukupno,  
    COUNT(razlog_isplate) broj_isplata  
    FOR razlog_isplate IN ('PLATA_DE01' AS PLATA_DE01, 'PLATA_DE02' AS PLATA_DE02,  
        'PUTNI_TROSKOVI' AS PUTNI_TROSKOVI, 'BONUS' AS BONUS, 'DNEVNICA' AS DNEVNICA)  
    )  
ORDER BY mbr;
```

# Pivot - Zadatak za vežbu

- Za svakog radnika prikazati iznos isplaćen za svaki mesec u prvoj polovini 2023. godine. Isplate po mesecima prikazati u vidu kolona.

```
SELECT *
FROM (
    SELECT mbr, mesec, godina, iznos
    FROM isplate_radnicima
    WHERE godina = 2023
)
PIVOT (
    SUM(iznos) ukupno
    FOR mesec
    IN ('January', 'February', 'March', 'April', 'May', 'June')
)
ORDER BY mbr;
```

# Unpivot

```
SELECT lista_izraza
FROM lista_tabela
UNPIVOT [INCLUDE | EXCLUDE NULLS](
    lista_ciljnih_obeležja_za_vrednosti
    FOR kategorizaciono_obeležje
    IN (lista_naziva_obeležja_unpivotiranja)
)
```

- `lista_ciljnih_obeležja_za_vrednosti` - obeležja koja će biti kreirana kako bi se čuvale vrednosti za unpivotirane kolone
- `kategorizaciono_obeležje` - obeležje koje će biti kreirano kako bi se čuvale vrednosti kategorija
- `lista_naziva_obeležja_unpivotiranja` - definiše listu obeležja za koje bi trebalo da se izvrši transponovanje
  - Za svaku vrednost u listi kategorizaciono obeležje će dobiti novu vrednost
  - Poželjno dodeliti alijase kako bi vrednosti kategorizacionog obeležja bile smislene

# Unpivot - Primer

- Kreirati tabelu isplate\_pivotirane

```
CREATE TABLE isplate_pivotirane AS
SELECT *
FROM
    (SELECT mbr, razlog_isplate, iznos FROM isplate_radnicima)
PIVOT (
    SUM(iznos) ukupno,
    COUNT(razlog_isplate) broj_isplata
    FOR razlog_isplate IN ('PLATA_DE01' AS PLATA_DE01, 'PLATA_DE02' AS PLATA_DE02,
        'PUTNI_TROSKOVI' AS PUTNI_TROSKOVI, 'BONUS' AS BONUS, 'DNEVNICA' AS DNEVNICA)
)
ORDER BY mbr;
```



# Unpivot - Primer

- Na osnovu sadržaja tabele ISPLATE\_PIVOTIRANE, ispisati koliko je za svaku kategoriju razloga\_isplate radnicima isplaćeno sredstava. Svaka kategorija trebalo bi da bude ispisana kao poseban red u izveštaju.

```
SELECT *
FROM (SELECT MBR, PLATA_DE01_UKUPNO, PLATA_DE02_UKUPNO, PUTNI_TROSKOVI_UKUPNO ,
BONUS_UKUPNO, DNEVNICA_UKUPNO FROM ISPLATE_PIVOTIRANE)
UNPIVOT EXCLUDE NULLS (
    IZNOS
    FOR RAZLOG
    IN (
        PLATA_DE01_UKUPNO AS 'PLATA_DE01', PLATA_DE02_UKUPNO AS 'PLATA_DE02',
        PUTNI_TROSKOVI_UKUPNO AS 'PUTNI_TROSKOVI', BONUS_UKUPNO AS 'BONUS',
        DNEVNICA_UKUPNO AS 'DNEVNICA'
    )
);
```

# WITH – Rekurzija

- Blok podupita pomoću WITH
- Blok sadrži dva upita vezana preko UNION ALL
  - prvi upit određuje početni skup podataka
  - drugi upit obezbeđuje rekurzivno proširenje skupa putem unije sa tekućim skupom
- Postupak se zaustavlja kada ne dođe do promene skupa prilikom proširenja

```
WITH naziv_upita(lista_obeležja) as
(
    upit1
    UNION ALL
    upit2
)
```

# WITH – Rekurzija – Primer

- Prikazati za svakog radnika sve direktno i indirektno nadređene radnike.

```
WITH hijerarhija(mbr,sef) AS (  
    SELECT mbr, sef  
    FROM radnik  
    UNION ALL  
    SELECT r.mbr, h.sef  
    FROM radnik r, hijerarhija h  
    WHERE r.sef = h.mbr AND h.sef IS NOT NULL  
)  
SELECT * FROM hijerarhija ORDER BY mbr, sef;
```

## WITH – Rekurzija – Zadatak za vežbu

- Prikazati za svakog radnika sve direktno i indirektno podređene radnike.

```
WITH hijerarhija(mbr,pod) AS(  
    SELECT sef, mbr  
    FROM radnik  
    UNION ALL  
    SELECT h.mbr, r.mbr  
    FROM hijerarhija h, radnik r  
    WHERE h.pod = r.sef AND h.mbr IS NOT NULL  
)  
SELECT * FROM hijerarhija ORDER BY mbr, pod;
```

# WITH – Rekurzija – Zadatak za vežbu

- Prikazati za svakog radnika sve direktno i indirektno podređene radnike, ako nema podređenih prikazati null umesto oznake podređenog.

```
WITH hijerarhija(mbr,pod) AS(  
    SELECT sef, mbr  
    FROM radnik  
    UNION ALL  
    SELECT h.mbr, r.mbr  
    FROM hijerarhija h, radnik r  
    WHERE h.pod = r.sef and h.mbr IS NOT NULL  
)  
SELECT r.mbr, h.pod FROM hijerarhija h, radnik r  
WHERE r.mbr = h.mbr(+) ORDER BY mbr, pod;
```

# WITH – Rekurzija – Zadatak za vežbu

- Prikazati za svakog radnika oznaku (ime i prezime) šefa.

```
WITH hijerarhija(mbr,imeprz,sef) AS (  
    SELECT mbr, ime||' '||prz, sef  
    FROM radnik  
    WHERE sef IS NULL  
    UNION ALL  
    SELECT r.mbr, r.ime||' '||r.prz, h.mbr  
    FROM radnik r, hijerarhija h  
    WHERE r.sef = h.mbr  
)  
SELECT * FROM hijerarhija;
```

# WITH – Rekurzija – Search

**SEARCH BREADTH | DEPTH FIRST**

**BY lista\_obeležja**

**SET pseudo-obeležje**

- Klauzula SEARCH
- Definiše poredak redova
  - BREADTH FIRST, DEPTH FIRST
- BY – poredak redova na istom nivou
- SET – vrednost pseudo-obeležja po redosledu redova
  - pseudo-obeležje automatski postaje deo rezultata

# WITH – Rekurzija – Search – Primer

- Prikazati hijerarhiju rukovođenja po nivoima.

```
WITH hijerarhija(mbr,imeprz,sef,nivo) AS (  
    SELECT mbr, ime||' '||prz, sef, 1 AS nivo  
    FROM radnik  
    WHERE sef IS NULL  
    UNION ALL  
    SELECT r.mbr, r.ime||' '||r.prz, h.mbr, nivo+1  
    FROM radnik r, hijerarhija h  
    WHERE r.sef = h.mbr  
)  
SEARCH DEPTH FIRST BY imeprz SET poredak  
SELECT nivo, rpad(' ',3*nivo)||imeprz AS imeprz, mbr, sef  
FROM hijerarhija  
order by poredak;
```



# WITH – Rekurzija – Search – Primer

- Prikazati za svakog radnika lanac rukovođenja.

```
WITH hijerarhija(imeprz,mbr,sef,lanac,glavni) AS (  
    SELECT ime||' '||prz, mbr, sef, '/'||ime||' '||prz AS lanac, ime||' '||prz AS glavni  
    FROM radnik  
    WHERE sef IS NULL  
    UNION ALL  
    SELECT r.ime||' '||r.prz, r.mbr, r.sef, h.lanac||'/'||r.ime||' '||r.prz AS lanac, h.glavni  
    FROM radnik r, hijerarhija h  
    WHERE r.sef = h.mbr  
)  
SELECT imeprz,mbr,sef,lanac,glavni  
FROM hijerarhija;
```

# WITH – Rekurzija – Cycle

CYCLE lista\_obeležja

SET pseudo\_obeležje

TO oznaka\_ciklusa

DEFAULT oznaka\_odsustva\_ciklusa

- Klauzula CYCLE
- Označava cikluse u rekurziji
  - prema proveru zadate liste obeležja
- Oznaka prisustva ili odsustva ciklusa
  - pseudo-obeležje automatski postaje deo rezultata
  - jedan karakter

# WITH – Rekurzija – Cycle — Primer

- Prikazati za svakog radnika lanac rukovođenja, uz proveru postojanja ciklusa.

```
WITH hijerarhija(imeprz,mbr,sef,lanac,glavni) AS (  
    SELECT ime||' '||prz, mbr, sef, '/'||ime||' '||prz AS lanac, ime||' '||prz AS glavni  
    FROM radnik where sef IS NULL  
    UNION ALL  
    SELECT r.ime||' '||r.prz, r.mbr, r.sef, h.lanac|| '/'||r.ime||' '||r.prz AS lanac, h.glavni  
    FROM radnik r, hijerarhija h  
    WHERE r.sef = h.mbr  
)  
SEARCH BREADTH FIRST BY imeprz  
SET poredak CYCLE mbr  
SET ciklus to 'x' default 'o'  
SELECT imeprz,mbr,sef,lanac,glavni, ciklus  
FROM hijerarhija;
```

# WITH – Rekurzija – Cycle — Domaći zadatak

- Prikazati sve podređene za radnika čiji je mbr 70.

```
WITH hijerarhija(imeprz,mbr,sef,lanac,glavni) AS (  
    SELECT ime||' '||prz, mbr, sef, '/'||ime||' '||prz AS lanac, ime||' '||prz AS glavni  
    FROM radnik  
    WHERE mbr = 70  
    UNION ALL  
    SELECT r.ime||' '||r.prz, r.mbr, r.sef, h.lanac||'/'||r.ime||' '||r.prz AS lanac, h.glavni  
    FROM radnik r, hijerarhija h  
    WHERE r.sef = h.mbr  
)  
SEARCH BREADTH FIRST BY imeprz  
SET poredak CYCLE mbr  
SET ciklus to 'x' DEFAULT 'o'  
SELECT imeprz,mbr,sef,lanac,glavni, ciklus  
FROM hijerarhija;
```

# WITH – Rekurzija – Cycle — Domaći zadatak

- Promeniti šefa radnika 70 da bude radnik 140.

```
UPDATE radnik  
SET sef = 140  
WHERE mbr = 70;
```

- Ponovo prikazati sve podređene za radnika 70.
- Poništiti promenu šefa radnika 70.

```
ROLLBACK;
```

# Sparse matrice

- Predstavljaju matrice gde većina elemenata sadrži vrednost 0.
- Velike sparse matrice se pojavljuju u naučnim proračunima prilikom rešavanja parcijalnih diferencijalnih jednačina.
- Primer množenja dve sparse matrice:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 9 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Množenje sparse matrica

```
SELECT a.row_num, b.col_num, SUM(a.value*b.value)
FROM a, b
WHERE a.col_num = b.row_num
GROUP BY a.row_num, b.col_num;
```

# Množenje sparse matrica

```
SELECT a.row_num, a.col_num, a.value
FROM a
WHERE NOT EXISTS (SELECT 0 FROM b WHERE a.col_num = b.col_num and a.row_num = b.row_num);
UNION
SELECT b.row_num, b.col_num, b.value
FROM b
WHERE NOT EXISTS (SELECT 0 FROM a WHERE a.col_num = b.col_num and a.row_num = b.row_num);
UNION
SELECT a.row_num, a.col_num, a.value + b.value
FROM a,b
WHERE a.col_num = b.col_num and a.row_num = b.row_num;
```



# Kraj!

Hvala na pažnji!