Baze podataka

Predavanja

4. SQL (1. dio)

Ožujak, 2021.

SQL - Uvod

- objedinjuje funkcije jezika za definiciju podataka (DDL) i jezika za rukovanje podacima (DML)
- razvoj započeo 70-tih godina
  - IBM San José Research Laboratory (California, USA)
- *Structured Query Language* je standardni jezik relacijskih baza podataka (*database language*)
  - 1986. godine - SQL-86 ili SQL1 (prva verzija standarda)
  - 1992. godine - SQL-92 ili SQL2
  - 1999. godine - SQL:1999
  - 2003. godine - SQL:2003
  - 2008. godine - SQL:2008
  - 2011. godine - SQL:2011
  - 2016. godine – SQL:2016
- proizvođači komercijalnih sustava često ugrađuju i svoje nestandardne DDL i DML naredbe
  - programski kod postaje neprenosiv između različitih SQL sustava
  - otežava se usaglašavanje oko budućih standarda.

SQL - Uvod

- neproceduralnost - naredbom je dovoljno opisati što se želi dobiti kao rezultat - nije potrebno definirati kako do tog rezultata doći
- u SUBP ugrađeni optimizator upita pronalazi najefikasniji način obavljanja upita

zupanija

sifZup	nazZup
2	Primorsko-goranska
7	Varaždinska
4	Istarska

mjesto

pbr	nazMjesto	sifZup
42000	Varaždin	7
51000	Rijeka	2
52100	Pula	4
42230	Ludbreg	7

- ispisati podatke o mjestima u Varaždinskoj županiji. Rezultate poredati prema nazivu mjesta

```
SELECT mjesto.* FROM mjesto, zupanija
   WHERE mjesto.sifZup = zupanija.sifZup
     AND nazZup = 'Varaždinska'
   ORDER BY nazMjesto;
```

SQL - Vrste objekata

• Baza podataka	*Database*
• Relacija (tablica)	*Table*
• Atribut (stupac, kolona)	*Column*
• Virtualna tablica (pogled)	*View*
• Integritetsko ograničenje	*Constraint*
• Indeks	*Index*
• Pohranjena procedura	*Stored Procedure*
• Okidač	*Trigger*

SQL - Identifikatori

- Identifikatori (imena objekata) se formiraju iz slova, znaka '_' i znamenki. Prvi znak od ukupno 128 značajnih (signifikantnih) znakova mora biti slovo ili znak '_'

- ispravno formirani identifikatori

```
stud
ispiti2000godine
stud_ispit
_1mjesec
```

- neispravno formirani identifikatori

```
_11.mjesec
11mjesec
stud-ispit
```

SQL - Rezervirane riječi

- SQL je "neosjetljiv" (*case insensitive*) na razliku između velikih i malih slova kada su u pitanju rezervirane riječi (SELECT, UPDATE, DELETE, FROM, WHERE, ...) i identifikatori

```
SELECT * FROM mjesto
   WHERE sifZupanija = 7
```

≡

```
select * FrOm MJesto
   wHERE SIFZupanIJA = 7
```

- Međutim, razlika između velikih i malih slova **postoji** kad su u pitanju nizovi znakova

'Ivan' ≠ 'IVAN'

SQL - Format naredbi

- SQL je jezik slobodnog formata naredbi (jednako kao C)

```
SELECT * FROM mjesto
   WHERE sifZupanija = 7
```

≡

```
      SELECT
*
    FROM
 mjesto                 WHERE
sifZupanija = 7
```

SQL - Korištenje komentara

- "blok komentari" (jednako kao u programskom jeziku C)
  - dio teksta omeđen oznakama `/*` i `*/`

```
/* ovo je komentar koji se
        proteže kroz više redaka teksta */
```

- "linijski komentari"
  - mjesto u retku na kojem se nalaze znakovi `--` predstavlja početak komentara koji se proteže do kraja retka

```
-- ovo je komentar
SELECT * FROM mjesto  -- ovo je komentar
   WHERE pbr = 10000  -- ovo je komentar
```

SQL - Tipovi podataka

- **INTEGER**
  - cijeli broj pohranjen u 4 bajta u aritmetici dvojnog komplementa. Dopušteni raspon brojeva određen je intervalom

    $[-2^{n-1},\ 2^{n-1} - 1]$ n=32
  - dakle, raspon brojeva je:

    [-2147483648, 2147483647]

```
Konstante:
5     -30000     0     1765723712     NULL
```

- **SMALLINT**
  - cijeli broj pohranjen u 2 bajta. Raspon brojeva koji se mogu prikazati je [-32768, 32767]

```
Konstante:
5     -30000     0     NULL
```

SQL - Tipovi podataka

- **CHAR(m)**
  - znakovni niz (*string*) s definiranom duljinom. Npr: CHAR(24).

```
Konstante:
'Ana'     '12345'     NULL
'Dvostruki navodnik " unutar niza'
'Jednostruki navodnik '' unutar niza'
```

    - pri unosu niza čija je duljina <m, preostala mjesta se pune prazninama (prateće praznine)
- **VARCHAR(m)**
  - znakovni niz (*string*) varijabilne duljine, s unaprijed definiranom maksimalnom duljinom. Npr: VARCHAR(24).
    - nema pratećih praznina

- **uočite:** koriste se **jednostruki** navodnici (drugačije nego u jeziku C)

SQL - Tipovi podataka

- **NCHAR(m)**
  - jednako kao i CHAR tip podatka, ali omogućava ispravno leksikografsko uređenje nizova znakova koji sadrže znakove iz nacionalnih kodnih stranica (*character set*). Koristi se onda kada se predviđa potreba za leksikografskim poretkom nizova znakova u kojima se pojavljuju specifični nacionalni znakovi (Č, Ć, Š, Đ, Ž, ...), npr. za atribut prezime

- **NVARCHAR(m)**
  - kao NCHAR tip podatka ali varijabilne duljine

  - NCHAR i NVARCHAR nisu podržani u PostgreSQL
  - Zastarjeli koncept – danas tipovi CHAR i VARCHAR omogućavaju korištenje nacionalnih kodnih stranica

SQL - Tipovi podataka

- **REAL**
  - odgovara tipu podatka **`float`** u jeziku C (IEEE-754 format prikaza - jednostruka preciznost)

```
Konstante:
23    -343.23    232.233E3    23.0e-24    NULL
```

- **DOUBLE PRECISION**
  - odgovara tipu podatka **`double`** u jeziku C (IEEE-754 format prikaza - dvostruka preciznost)

```
Konstante:
23    -343.23    232.233E3    23.0e-302    NULL
```

SQL - Tipovi podataka

- **NUMERIC(m, n) ili DECIMAL(m, n)**
  - ekvivalentni tipovi u PostgreSQL-u
  - **m** - ukupni broj znamenki (*precision*)
  - **n** - broj znamenki iza decimalne točke (*scale*, n ≤ m)
  - npr., NUMERIC(15, 3) predstavlja decimalni broj s ukupno najviše 15 znamenki, od toga se najviše 3 znamenke nalaze iza decimalne točke
  - razlikuje se od **`float`** ili **`double`** tipa podatka u jeziku C
    - ako se za pohranu broja 1.3 koristi tip podatka NUMERIC(2,1), broj će biti pohranjen **bez numeričke pogreške**
    - ako se za pohranu broja 1.3 koristi tip podatka **`float`** u jeziku C, u memoriji će se zapravo pohraniti broj 1.299999523162842 (num. pogreška zbog karakteristika IEEE-754 formata pohrane)

Konstante - primjer za NUMERIC(7, 2):

```
5    8.1      -12345.67     0     NULL
```

SQL - Tipovi podataka

- **DATE**
  - podaci ovog tipa se uvijek prikazuju u obliku datuma (npr. 13.12.2019). Ovaj tip podatka omogućava korištenje sljedećih operacija zbrajanja i oduzimanja:
    - **`dat1 - dat2`** rezultat je podatak tipa INTEGER - broj dana proteklih između *dat2* i *dat1*
    - **`dat + cijeliBroj`** rezultat je podatak tipa DATE - izračunava koji datum je *cijeliBroj* dana nakon dana *dat*
    - **`dat - cijeliBroj`** rezultat je podatak tipa DATE - izračunava koji datum je *cijeliBroj* dana prije dana *dat*

Konstante:
`'17.2.2017'` `'16.07.1969'` **`NULL`**

  - omogućuje kontrolu ispravnosti datuma:

```
INSERT INTO utrka (datum) VALUES ('30.2.2020');
```

```
ERROR: date/time field value out of range: "30.2.2020"
```

SQL - Tipovi podataka

Ostali vremenski tipovi podataka - rezolucija p<=6 (mikrosekunda)

- TIMESTAMP [(p)] – vremenski trenutak = datum + vrijeme

Primjer: `'13.03.2017 14:24:54' '13.03.2017 14:24:54.678901'`

- TIME [(p)] – vrijeme – sati, minute, sekunde na najviše 6 decimala
  – vrijeme u danu
  – ne znamo u kojem danu!

Primjer: `'14:24:54' '14:24:54.678901'`

- INTERVAL [fields] [(p)] – interval – mogući rasponi: od godina do sekunda na najviše 6 decimala

  trajanje nečega – npr. trajanje filma, starost

  fields - year, month, day, hour, minute, second, year to month, day to hour, day to minute, day to second, hour to minute, hour to second, minute to second

  Ako definicija sadrži i **fields** i **p - fields** mora sadržavati sekunde

```
Primjer: '5 YEAR 2 MONTH 1 DAY' - 5 godina 2 mjeseca i 1 dan
         '2 14:24:54'     - 2 dana 14 sati 24 minute i 54 sekunde
           '00:4:54'      - 4 minute i 54 sekunde
```

Primjer - interval

Hrvatski skijaš Filip Zubčić pobjednik je velelslalomske utrke vožene u japanskoj Niigata Yuzawa Naebi

Izvor: tPortal, 20.2.2020.

1. FILIP ZUBČIĆ (HRV) 2:37,25
2. Marco Odermatt (CHU) 2:37,99 +00,74
3. Tommy Ford (USA) 2:38,32 +01,07

trajanje vožnje - **interval**

kašnjenje - **interval**

Primjeri - Interval

```
CREATE TABLE utrka
    (ime VARCHAR(30))
     , drzava VARCHAR(3)
     , vrsta VARCHAR(30)
     , datum DATE
     , vrijeme INTERVAL MINUTE TO SECOND(2)
     , kasnjenje INTERVAL MINUTE TO SECOND(2));

INSERT INTO utrka VALUES
('Filip Zubčić','HRV','veleslalom','20-02-2020' '2:37.25','0');
INSERT INTO utrka values
('Marco Odermatt','CHE','veleslalom','20-02-2020','2:37.99','00.74');
INSERT INTO utrka VALUES
('Tommy Ford','USA','veleslalom','20-02-2020','2:38.32’,'01.07’);
SELECT * FROM utrka;
```

Data Output

	ime character varying (30)	drzava character varying (3)	vrsta character varying (30)	datum date	vrijeme interval	kasnjenje interval
1	Filip Zubčić	HRV	veleslalom	20.02.2020	00:02:37.25	00:00:00
2	Marco Odermatt	CHE	veleslalom	20.02.2020	00:02:37.99	00:00:00.74
3	Tommy Ford	USA	veleslalom	20.02.2020	00:02:38.32	00:00:01.07

Eksplicitna pretvorba tipova podataka

- Dvije ekvivalentne sintakse:

  CAST (expression AS type)

  expression::type

```
CREATE TABLE skSati (
   rbr          INTEGER
 , skSatOd      TIMESTAMP
 , skSatDo      TIMESTAMP);
```

skSati

rbr	skSatOd	skSatDo
1	14.03.2017 08:15:00.012345	14.03.2017 09:00:00.012345
2	14.03.2017 09:15:00.012345	14.03.2017 10:00:00.012345

```
SELECT CAST (skSatOd AS TIME(0)) vrijeme
     , skSatOd::TIMESTAMP(0) datumIVrijeme
   FROM skSati
```

vrijeme	datumIVrijeme
08:15:00	14.03.2017 08:15:00
09:15:00	14.03.2017 09:15:00

```
SELECT CURRENT_DATE,
       '18.3.2019',
       '18.3.2019'::DATE,
       '18.3.2019'::TIMESTAMP;
```

konverzijom datuma u tip TIMESTAMP,
dobit ćemo PONOĆ toga dana - (00:00:00)

! Važno kod ispitivanja pripadnosti nekom periodu

	current_date date	?column? text	date date	timestamp timestamp without time zone
1	18.03.2019	18.3.2019	18.03.2019	18.03.2019 00:00:00

Funkcije (*function expression*)

- ABS
- MOD
- ROUND
- SUBSTRING
- UPPER
- LOWER
- TRIM
- CHAR_LENGTH
- OCTET_LENGTH
- EXTRACT
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CURRENT_USER

Funkcije (*function expression*)

- **ABS (*num_expression*)**
  - računa apsolutnu vrijednost izraza

num_expression – mora biti numerički tip podatka (INTEGER, NUMERIC, FLOAT, ...)

rezultat funkcije – tip podatka ovisi o tipu podatka ulaznog argumenta

- **MOD (*dividend, divisor*)**
  - računa ostatak dijeljenja djeljenika i djelitelja (djelitelj ne smije biti 0)
  - argumenti ne moraju biti cjelobrojni

dividend (djeljenik) – numerički tip podatka (INTEGER, DECIMAL, NUMERIC, FLOAT, ...)
divisor (djelitelj) – numerički tip podatka (INTEGER, DECIMAL, NUMERIC, FLOAT, ...)

rezultat funkcije – tip podatka ovisi o tipu podatka ulaznih argumenta

Funkcije (*function expression*)

- **ROUND (*expression[, rounding_factor]*)**
  - zaokružuje vrijednost izraza *(expression)*
  - ako se ne navede *rounding_factor,* uzima se da je njegova vrijednost 0

expression *(izraz koji se zaokružuje)* –
numerički tip podatka (INTEGER, NUMERIC, FLOAT, …)
rounding_factor *(preciznost na koju se vrši zaokruživanje)* – cjelobrojni tip podatka

rezultat funkcije – tip podatka ovisi o tipu podatka ulaznog argumenta (expression) i o rounding_factor

Funkcije (*function expression*)

- **SUBSTRING (*source_string FROM start_position [FOR length]*)**
  - vraća podniz zadanog niza
  - ako se *length* ne navede vraća se podniz koji počinje na *start_position*, a završava gdje i niz *source_string*

source_string	–	zadani niz čiji se podniz traži funkcijom mora biti izraz tipa niza znakova
start_position	–	broj koji predstavlja poziciju prvog znaka podniza u zadanom nizu *source_string;* mora biti izraz cjelobrojnog tipa
length(duljina)	–	broj znakova koje funkcija treba vratiti počevši od *start_position;* mora biti izraz cjelobrojnog tipa

Funkcije (*function expression*)

- **UPPER (*expression*)**
  - sva mala slova (a-z) koja se pojavljuju u zadanom nizu *expression* zamjenjuje odgovarajućim velikim slovima (A-Z)

- **LOWER (*expression*)**
  - sva velika slova (A-Z) koja se pojavljuju u zadanom nizu *expression* zamjenjuje odgovarajućim malim slovima (a-z)

expression – zadani niz nad kojim se vrši pretvorba slova
mora biti izraz tipa niza znakova

Funkcije (*function expression*)

- **TRIM (*expression*)**
  - funkcija vraća niz znakova koji nastaje tako da se s početka i kraja niza *expression* izbace sve praznine

- **CHAR_LENGTH (*expression*)**
  - funkcija vraća broj znakova u zadanom nizu *expression* ignorira prateće (*trailing*) praznine

- **OCTET_LENGTH (*expression*)**
  - funkcija vraća broj byte-ova zadanog niza *expression* uključujući i prateće praznine

expression – mora biti izraz tipa niza znakova

Funkcije (*function expression*)

- **CURRENT_USER**
  - funkcija vraća *login* korisnika koji je trenutno prijavljen za rad sa bazom podataka
- **CURRENT_DATE**
  - funkcija vraća podatak tipa DATE koji predstavlja današnji datum (dobiven iz operacijskog sustava)
- **CURRENT_TIME**
  - funkcija vraća podatak tipa TIME **s vremenskom zonom** koji predstavlja trenutno vrijeme. Podatak je oblika:

    hh:mm:ss.xxxxxx – npr. 13:36:56.225091+01:00
- **CURRENT_TIMESTAMP**
  - funkcija vraća podatak tipa TIMESTAMP koji predstavlja (*current*) datum i vrijeme **s vremenskom zonom**. Podatak je oblika:
  - DD.MM.YYYY HH:MI:SS.xxxxxx npr. "2018-03-15 13:39:19.89626+01"

Funkcije (*function expression*)

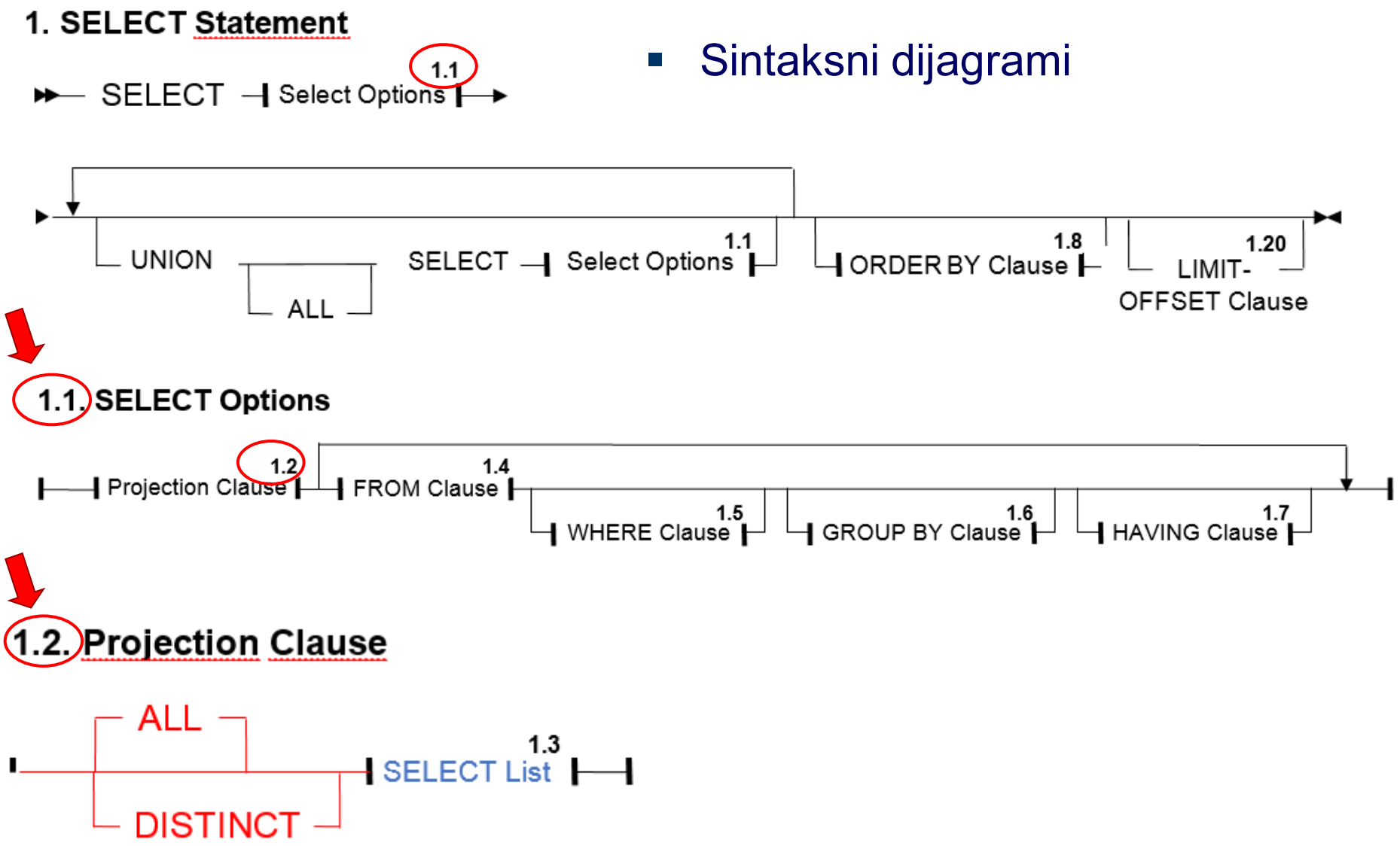
- **EXTRACT (field FROM source)**
  - funkcija vraća
    redni broj godine, mjeseca, dana, tjedna **za datum** sadržan u ***source***
    sata, minute, sekunde u danu **za vrijeme** sadržano u ***source***
- Za ***field***
  = dow
  - funkcija vraća redni broj dana u tjednu za zadani datum
    (0 – nedjelja, 1 – ponedjeljak, 2 – utorak, itd…)

  = doy
  - funkcija vraća redni broj dana u godini za zadani datum

field – *year, month, day, hour, minute, second, week*
dow, doy,
source – izraz tipa TIMESTAMP ili INTERVAL

SELECT Statement

- Sintaksni dijagrami



Projection Clause

- Primjeri:

student

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Kolar	52000
107	Ban	10000

```
SELECT ALL prez
          , postbr
  FROM student;
```

≡

```
SELECT prez
     , postbr
  FROM student;
```

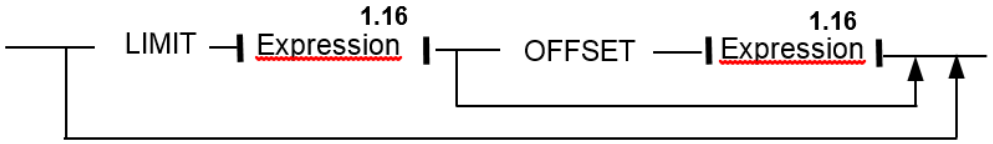
prez	postBr
Kolar	52000
Horvat	10000
Kolar	52000
Ban	10000

```
SELECT DISTINCT prez
              , postbr
  FROM student;
```

prez	postBr
Kolar	52000
Horvat	10000
Ban	10000

LIMIT- OFFSET Clause

1.20 LIMIT-OFFSET Clause



student

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Kolar	52000
107	Ban	10000

- Primjeri:

```
SELECT *
  FROM student
 LIMIT 2;
```

matBr	prez	postBr
102	Horvat	10000
105	Kolar	52000

Ne zna se koje dvije n-torke će se dobiti kao "prve dvije" - poredak n-torki u relaciji (niti u SQL tablici) nije definiran

```
SELECT DISTINCT prez
  FROM student
LIMIT 2 OFFSET 1;
```

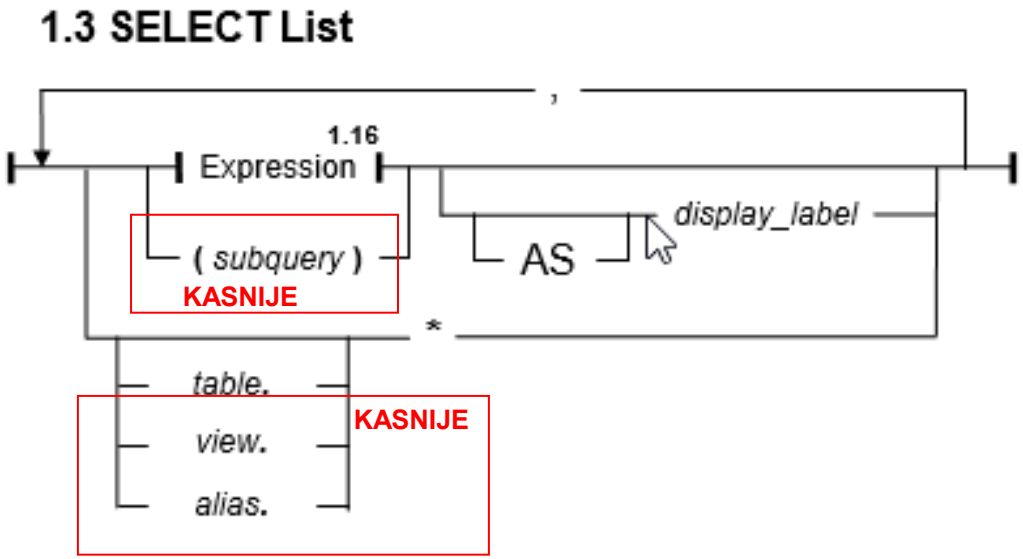
prez
Horvat
Ban

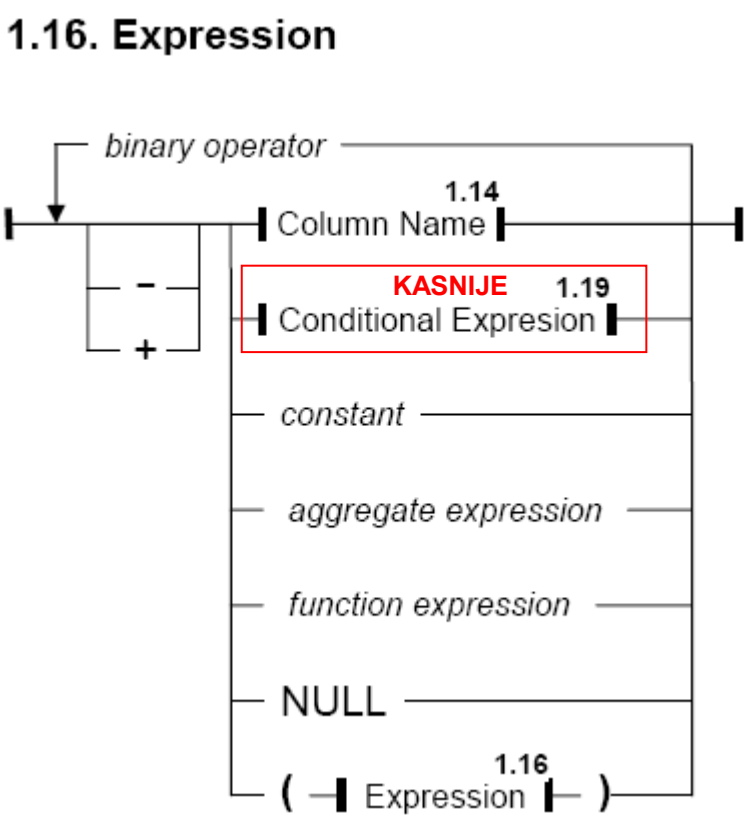
Ne zna se koje su to "prve dvije" n-torke nakon jedne preskočene

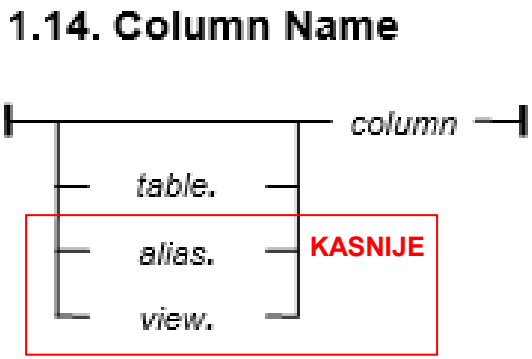
```
SELECT *
  FROM student
LIMIT 100;
```

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Kolar	52000
107	Ban	10000

SELECT List


1.3 SELECT List
,
1.16
Expression
(subquery)
KASNIJE
AS
display_label
*
table.
view.
alias.
KASNIJE


1.16. Expression
binary operator
1.14
Column Name
-
+
KASNIJE
1.19
Conditional Expresion
constant
aggregate expression
function expression
NULL
1.16
(
Expression
)


1.14. Column Name
column
table.
alias.
view.
KASNIJE

SELECT List

- Primjer:

mjesto

pbr	nazMjesto	sifZup
42000	Varaždin	7
52100	Pula	4

```
SELECT mjesto.pbr, *, pbr, mjesto.*
    FROM mjesto
```


pbr	pbr	nazMjesto	sifZup	pbr	pbr	nazMjesto	sifZup
42000	42000	Varaždin	7	42000	42000	Varaždin	7
52100	52100	Pula	4	52100	52100	Pula	4

U ovom primjeru rezultat nije relacija!

Izraz (*Expression*)

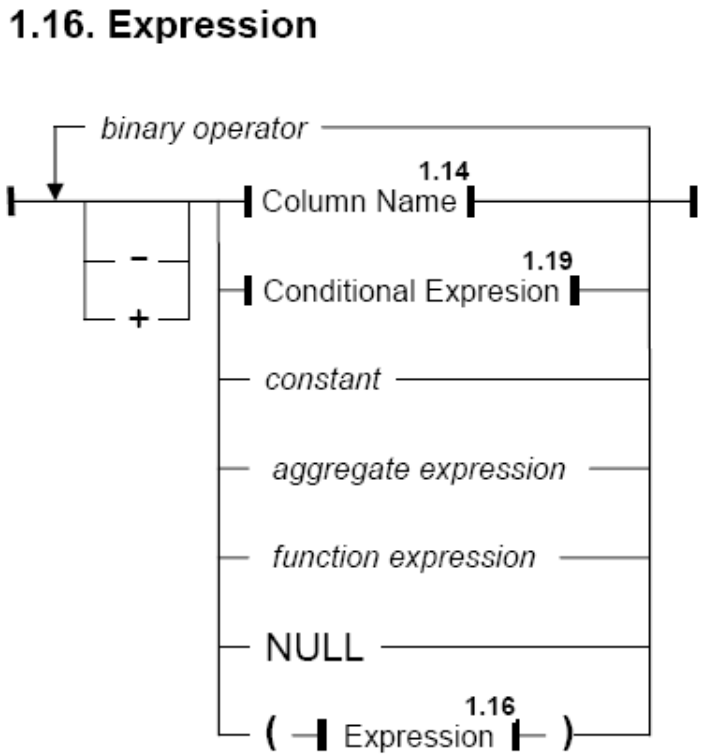
- Unarni operatori

  + –

- Binarni operatori

  + – * /

  || ulančavanje nizova znakova
  (nadovezivanje, konkatenacija)



- Redoslijed obavljanja operacija u složenim izrazima određuje se prema istim pravilima kao u programskom jeziku C (implicitni redoslijed obavljanja se može promijeniti upotrebom okruglih zagrada)
- Konverzija tipova podataka tijekom evaluacije izraza obavlja se prema sličnim pravilima kao u programskom jeziku C

Izraz (primjeri)

- unarni, binarni operatori i konstante

```
CREATE TABLE bodovi (
   mbr      INTEGER
 , ime      VARCHAR(10)
 , prez     VARCHAR(10)
 , bodLab   INTEGER
 , bodMI    NUMERIC(4,1));
```

bodovi

mbr	ime	prez	bodLab	bodMI
100	Ana	Novak	12	67.2
107	Ivo	Ban	17	54.3

```
SELECT mbr,
       bodLab + bodMI,
       (bodLab + bodMI) / 100
  FROM bodovi;
```

expression

mbr	?column?	?column?
100	79.2	0.792
107	71.3	0.713

```
SELECT mbr, - bodLab
  FROM bodovi;
```

mbr	?column?
100	-12
107	-17

```
SELECT mbr, ime || prez
  FROM bodovi;
```

mbr	?column?
100	AnaNovak
107	IvoBan

```
SELECT mbr || '-' || ime ||
       ' ' ||prez
  FROM bodovi;
```

?column?
100-Ana Novak
107-Ivo Ban

Eksplicitna pretvorba tipova podataka

- Dvije ekvivalentne sintakse:

  CAST (expression AS type)

  expression::type

```
CREATE TABLE skSati (
   rbr         INTEGER
 , skSatOd     TIMESTAMP
 , skSatDo     TIMESTAMP);
```

skSati

rbr	skSatOd	skSatDo
1	14.03.2017 08:15:00.012345	14.03.2017 09:00:00.012345
2	14.03.2017 09:15:00.012345	14.03.2017 10:00:00.012345

```
SELECT CAST (skSatOd AS TIME(0)) vrijeme
     , skSatOd::TIMESTAMP(0)     datumIVrijeme
  FROM skSati
```

vrijeme	datumIVrijeme
08:15:00	14.03.2017 08:15:00
09:15:00	14.03.2017 09:15:00

Funkcije (primjeri) – matematičke funkcije

```
CREATE TABLE upl_ispl (
   rbr     INTEGER
 , racun   INTEGER
 , datum   DATE
 , iznos   NUMERIC(9,2));
```

upl_ispl

rbr	racun	datum	iznos
1	123456	22.02.2007	-120.00
2	878341	23.02.2007	173.47

```
SELECT rbr, ABS(iznos)
  FROM upl_ispl;
```

rbr	?column?
1	120.00
2	173.47

Ispisuje apsolutne vrijednosti iznosa

```
SELECT rbr, ROUND(iznos, 1)
  FROM upl_ispl;
```

rbr	?column?
1	-120.0
2	173.5

Ispisuje iznose zaokružene na jednu decimalu

```
SELECT rbr, MOD(iznos, 10)
  FROM upl_ispl;
```

rbr	?column?
1	0
2	3.47

Ispisuje ostatak dijeljenja iznosa sa 10

Funkcije (primjeri) – funkcije s nizovima

```
CREATE TABLE student (
    jmbag    VARCHAR(10)
  , ime      VARCHAR(25)
  , prezime  VARCHAR(25));
```

student

jmbag	ime	prezime
0036368145	Tomislav	Babić
0036369296	Linda	Jurić

Ispisati jmbag i inicijale studenata

```
SELECT jmbag
     , SUBSTRING(ime FROM 1 FOR 1) || '.' ||
       SUBSTRING(prezime FROM 1 FOR 1)|| '.'
  FROM student;
```

jmbag	?column?
0036368145	T.B.
0036369296	L.J.

Ispisati imena velikim slovima, a prezimena malim slovima

```
SELECT UPPER(ime)
     , LOWER(prezime)
  FROM student;
```

?column?	?column?
TOMISLAV	babić
LINDA	jurić

Funkcije (primjeri) – funkcije s nizovima

```
CREATE TABLE student (
   jmbag   CHAR(10)
 , ime     CHAR(10)
 , prezime CHAR(10));
```

student

jmbag	ime	prezime
0036368145	Tomislav	Ban
0036369296	Linda	Kekez

Ispisati imena i prezimena studenata iz kojih su izbačene praznine

```
SELECT ime, prezime
     , TRIM(ime) ||
       TRIM(prezime)
     , ime ||
       prezime
  FROM student;
```

Ime	prezime	?column?	?column?
Tomislav	Ban	TomislavBan	TomislavBan
Linda	Kekez	LindaKekez	LindaKekez

Neočekivano ponašanje

```
SELECT TRIM(ime) || ' ' || TRIM(prezime) AS imeiprezime
  FROM student;
```

imeiprezime
Tomislav Ban
Linda Kekez

Ispisati korisničko ime i broj znakova koji ga čine

```
SELECT CURRENT_USER
     , CHAR_LENGTH(CURRENT_USER);
```

?column?	?column?
postgres	8

S obzirom da neke operacije nad tipom podataka CHAR() daju neočekivani rezultat, u svim primjerima i zadatcima za nizove znakova koristit će se isključivo VARCHAR() tip podatka.

Funkcije (primjeri) – funkcije s nizovima

```
CREATE TABLE student (
    jmbag    VARCHAR(10)
  , ime      VARCHAR(25)
  , prezime  VARCHAR(25));
```

student

jmbag	ime	prezime
0036368145	Tomislav	Božanić
0036369296	Linda	Kekez

Ispisati broj znakova u prezimenu i broj znakova u prezimenu nad kojim je primijenjena TRIM funkcija.

S obzirom da se kod tipa podatka VARCHAR ne dodaju prateće praznine, funkcija CHAR_LENGTH će za originalni niz znakova i niz iz kojeg su primjenom TRIM funkcije izbačene prateće praznine dati jednak rezultat.

```
SELECT CHAR_LENGTH(prezime)
     , CHAR_LENGTH(TRIM(prezime))
  FROM student;
```

?column?	?column?
7	7
5	5

Ispisuje broj znakova i broj okteta (bajtova) koji čine prezime

```
SELECT CHAR_LENGTH(prezime)AS znakova
     , OCTET_LENGTH(prezime)AS okteta
FROM student;
```

znakova	okteta
7	9
5	5

Zbog utf8:
ž-2 okteta
ć-2 okteta

Operacije s vremenskim trenutcima i intervalima (prema SQL standardu)

Operand 1	Operator	Operand 2	Tip rezultata
Datetime	-	Datetime	Interval**
Datetime	+ ili -	Interval	Datetime
Interval	+	Datetime	Datetime
Interval	+ ili -	Interval	Interval
Interval	* ili /	Numeric	Interval
Numeric	*	Interval	Interval

Napomena:
Naziv Datetime predstavlja tipove podataka DATE, TIME i TIMESTAMP.

**Posebno za DATE tip podatka u PostgreSQL dobijemo rezultat je tipa INTEGER pretpostavljamo zbog kompatibilnosti prema starijim verzijama

Aritmetički Date/Time Operatori - PostgreSQL

Operacije	Dozvoljeni operatori (α)	Tip trezultata
DATE α DATE	-	INTEGER
DATE α TIME	+	TIMESTAMP
DATE α TIMETZ	+	TIMESTAMP WITH TIMEZONE
DATE α INT4	+ -	DATE
TIME α DATE	+	TIMESTAMP
TIME α INTERVAL	+ -	TIME
TIMETZ α DATE	+	TIMESTAMP WITH TIMEZONE
TIMETZ α INTERVAL	+ -	TIMETZ
TIMESTAMP α TIMESTAMP	-	INTERVAL
TIMESTAMP α INTERVAL	+ -	TIMESTAMP
INTERVAL α TIME	+	TIME

http://etutorials.org/SQL/Postgresql/Part+I+General+PostgreSQL+Use/Chapter+2.+Working+with+Data+in+PostgreSQL/DateTime+Values/

Temporalni tipovi podataka s vremenskom zonom

- TIME WITH TIME ZONE [(p)] ili TIMETZ [(p)]
- TIMESTAMP WITH TIME ZONE [(p)] ili TIMESTAMPTZ [(p)]

Primjeri: `'19:28:50.328304+01:00'` `'20.03.2019 19:28:50.328304 CET'`

U ovom kolegiju se nećemo baviti temporalnim podatcima s vremenskom zonom.

Rezultate funkcija CURRENT_TIME i CURRENT_TIMESTAMP, čiji rezultat uključuje vremensku zonu, svest ćemo na odgovarajuće tipove bez vremenske zone:

- CURRENT_TIME ::TIME(x),
- CURRENT_TIMESTAMP ::TIMESTAMP(x)

Funkcije (primjeri) – funkcije s datumom

```
CREATE TABLE nastavnik (
   sifNastavnik          INTEGER
 , datumZaposlenOd       DATE
 , datumZaposlenDo       DATE);
```

nastavnik

sifNastavnik	datumZaposlenOd	datumZaposlenDo
1	22.01.2000	28.02.2019
2	01.06.2010	25.03.2019

Napomena: pretpostavka je da je sljedeći upit izveden dana 20.03.2019.

Broj dana koji je protekao nakon prestanka zaposlenja nastavnika

```
SELECT sifNastavnik
     , CURRENT_DATE - datumZaposlenDo
  FROM nastavnik;
```

sifNastavnik	?column?
1	20
2	-5

Ispisuje dan, mjesec i godinu zaposlenja nastavnika, dan u tjednu i dan u godini

```
SELECT EXTRACT(DAY   FROM datumZaposlenOd) d
     , EXTRACT(MONTH FROM datumZaposlenOd) m
     , EXTRACT(YEAR  FROM datumZaposlenOd) g
     , EXTRACT(DOW   FROM datumZaposlenOd) dow
     , EXTRACT(DOY   FROM datumZaposlenOd) doy
FROM nastavnik;
```

d	m	g	dow	doy
22	1	2000	6	22
1	6	2010	2	152

Funkcije (primjeri) – funkcije s datumom

```
CREATE TABLE nastavnik (
    sifNastavnik          INTEGER
  , datumZaposlenOd       DATE
  , datumZaposlenDo       DATE);
```

nastavnik

sifNastavnik	datumZaposlenOd	datumZaposlenDo
1	22.01.2000	28.02.2019
2	01.06.2010	25.03.2019

Ispisati datum koji odgovara sljedećem danu nakon prestanka zaposlenja nastavnika

```
SELECT EXTRACT(DAY FROM datumZaposlenDo)+1   || '.' ||
       EXTRACT (MONTH FROM datumZaposlenDo)  || '.' ||
       EXTRACT (YEAR FROM datumZaposlenDo)
  FROM nastavnik
```

?column?
29.02.2019
26.03.2019

← ?

2019. nije prijestupna!

```
SELECT (EXTRACT(DAY FROM datumZaposlenDo)+1   || '.' ||
        EXTRACT (MONTH FROM datumZaposlenDo)  || '.' ||
        EXTRACT (YEAR FROM datumZaposlenDo))::DATE
  FROM nastavnik
```

ERROR: date/time field value out of range: "29.2.2019"

Ispravno rješenje:

```
SELECT datumZaposlenDo+1
  FROM nastavnik;
```

?column?
01.03.2019
26.03.2019

Primjeri ispisa datuma

```
SELECT CURRENT_DATE,
       '18.3.2019',
       '18.3.2019'::DATE,
       '18.3.2019'::TIMESTAMP;
```

	current_date date	?column? text	date date	timestamp timestamp without time zone
1	18.03.2019	18.3.2019	18.03.2019	18.03.2019 00:00:00

Trenutno vrijeme – CURRENT_TIME

- Funkcija CURRENT_TIME vraća vrijeme (dobiveno iz operacijskog sustava) u mikrosekundama, s vremenskom zonom, npr. 20:03:46.286634+01:00
- Primjeri:
  - trenutno vrijeme u mikrosekundama bez vremenske zone:
    CURRENT_TIME :: TIME(6) 20:03:46.286634
  - trenutno vrijeme u sekundama bez vremenske zone:
    CURRENT_TIME :: TIME(0) 20:03:46
  - 2 sata i 10 minuta nakon trenutnog vremena
    CURRENT_TIME + '2 hours 10 min’:: INTERVAL 22:13:46.286634+01:00
  - 2 sata i 10 minuta nakon trenutnog vremena – u sekundama, bez vremenske zone:
    CURRENT_TIME ::TIME(0) + '2 hours 10 min’:: INTERVAL 22:13:46

Analogno vrijedi i za funkciju CURRENT_TIMESTAMP

Funkcije (primjeri) – funkcije s vremenom i vremenskim trenutkom

```
CREATE TABLE predavanje (
   sifPredmet       INTEGER
 , oznGrupa         VARCHAR(10)
 , predavanjeOd     TIMESTAMP
 , predavanjeDo     TIMESTAMP);
```

predavanje

sifPredmet	oznGrupa	predavanjeOd	predavanjeDo
1	P02	26.03.2019 12:15:00	26.03.2019 14:00:00
2	P05	26.03.2019 16:15:00	26.03.2019 17:00:00

Ispisuje vremenski trenutak (TIMESTAMP) i vrijeme (TIME) u trenutku izvođenja upita, vrijeme koje će proteći od trenutka izvođenja upita do početka predavanja i trajanje predavanja.

```
SELECT CURRENT_TIMESTAMP::TIMESTAMP(0)                    sada,
       CURRENT_TIME::TIME(0)                              vrijemeSada,
       predavanjeOd - CURRENT_TIMESTAMP::TIMESTAMP(0)     doPocetka,
       predavanjeDo - predavanjeOd                        trajanje
 FROM predavanje
```

sada	vrijemeSada	doPocetka	trajanje
18.03.2019 18:45:06	18:45:06	7 days 17:29:54	01:45:00
18.03.2019 18:45:06	18:45:06	7 days 21:29:54	00:45:00

Funkcije (primjeri) – funkcije s datumom

Ispisuje vremenski trenutak star godinu dana

```
SELECT (EXTRACT(DAY FROM CURRENT_DATE ) || '.' ||
        EXTRACT(MONTH FROM CURRENT_DATE) || '.' ||
       (EXTRACT(YEAR FROM CURRENT_DATE)-1) || ' ' ||
        CURRENT_TIME)::TIMESTAMP(0) AS vrijemeStaroGodinudana
```

vrijemestarogodinudana
20.03.2018 18:48:56

Loše rješenje

Napomena: pretpostavka je da je gornji upit izveden dana 20.03.2019.

Bolje rješenje prethodnog problema

Ispisuje vremenski trenutak star godinu dana

```
SELECT CURRENT_TIMESTAMP::TIMESTAMP(0) - '1 YEAR'::interval
            AS prijeGodinuDana;
```

prijegodinudana timestamp without time zone
20.03.2018 18:48:56

Ispisuje vremenski trenutak za deset mjeseci i tri tjedna udaljen od sadašnjeg.

```
SELECT CURRENT_TIMESTAMP::TIMESTAMP(0)
         + '10 MONTHS 3 WEEKS'::interval
         AS za10MjTriTjedna;
```

Za10mjtritjedna timestamp without time zone
10.02.2020 18:49:27

Napomena: pretpostavka je da su gornji upiti izvedeni dana 20.03.2019.

Primjeri s intervalima

Interval se unosi kao: **quantity unit [*quantity unit*...] [*direction*]**
quantity je broj,
unit: *microsecond, millisecond, second, minute, hour, day, week, month, year, decade, century, millennium*;
direction može biti *ago*, ili može biti ispušten.
Dani, sati, minute i sekunde mogu se navesti bez jedinica.

```
CREATE TABLE intervali(
        interval1 INTERVAL,
        interval2 INTERVAL YEAR TO MONTH,
        interval3 INTERVAL DAY TO SECOND,
        interval4 INTERVAL HOUR TO SECOND);
INSERT INTO intervali VALUES
 ('5 years 2 months 1 day', '3 years 5 months', '2 14:24:54', '0:4:54');
SELECT * FROM intervali;
```

	interval1 interval	interval2 interval	interval3 interval	interval4 interval
1	5 years 2 mons 1 day	3 years 5 mons	2 days 14:24:54	00:04:54

Funkcije (primjeri) – funkcije s vremenskim trenutkom i intervalom

```
CREATE TABLE predavanje (
   sifPredmet     INTEGER
 , oznGrupa       VARCHAR(10)
 , datum          DATE
 , vrijemePoc     TIME
 , trajanje       INTERVAL);
```

predavanje

sifPredmet	oznGrupa	datum	vrijemePoc	trajanje
1	P02	26.03.2019	12:15:00	00:45:00
1	P02	26.03.2019	13:15:00	00:45:00

Ispisuje vrijeme početka (TIME) i vremenski trenutak (TIMESTAMP) kraja predavanja.C

```
SELECT vrijemePoc,
        datum + vrijemePoc + trajanje  krajPredavanja
  FROM predavanje
```

vrijemePoc	krajPredavanja
12:15:00	26.03.2019 13:00:00
13:15:00	26.03.2019 14:00:00

Funkcije i NULL vrijednosti

- Neka je binarni operator $\alpha \in \{ +, -, *, /, || \}$, a X i Y su izrazi
  - ako jedan ili oba operanda X, Y poprimaju NULL vrijednost, tada je rezultat izraza X α Y također NULL vrijednost

- Neka je unarni operator $\beta \in \{ +, - \}$, a X je izraz
  - ako operand X poprima NULL vrijednost, tada je rezultat izraza β X također NULL vrijednost

- Slično vrijedi i za funkcije
  - ako se kao jedan ili više argumenata funkcije zada NULL vrijednost, rezultat funkcije će također biti NULL vrijednost

Funkcije i NULL vrijednosti (primjer)

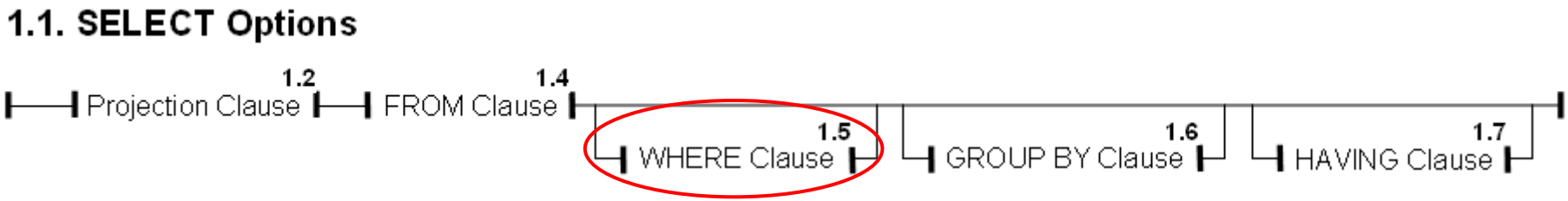
bodovi

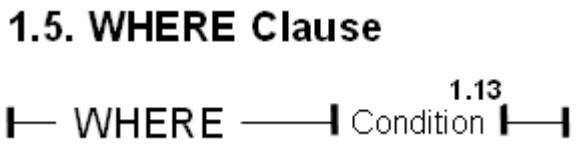
mbr	prez	bodLab
101	Novak	12
103	Ban	NULL
107	NULL	21
109	Kolar	NULL

```
SELECT mbr
     , MOD(bodLab,10) AS ostatak
     , SUBSTRING(prez FROM 1 FOR 2) AS podniz
  FROM bodovi;
```

mbr	ostatak	podniz
101	2	No
103	NULL	Ba
107	1	NULL
109	NULL	Ko

WHERE Clause



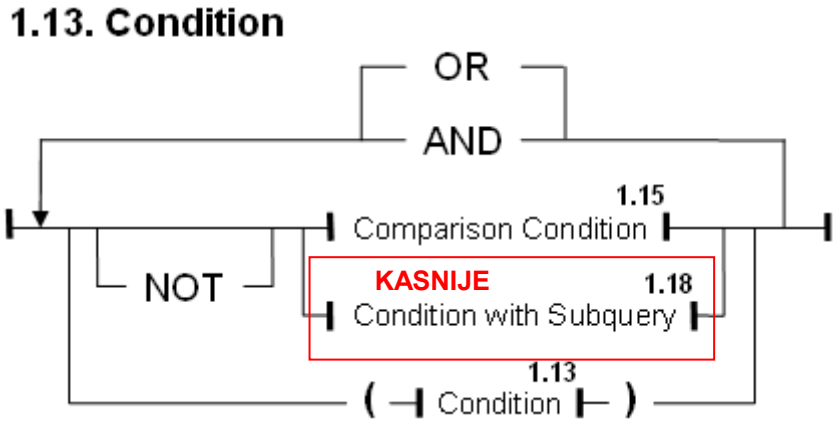


- Vrijednosti svake n-torke iz relacije *table* se uvrštavaju u *Condition* (a to je u stvari predikat). Ako je dobiveni sud istinit (*true*), n-torka se pojavljuje u rezultatu
- Mogući rezultati izračunavanja uvjeta: *true*, *false*, *unknown*

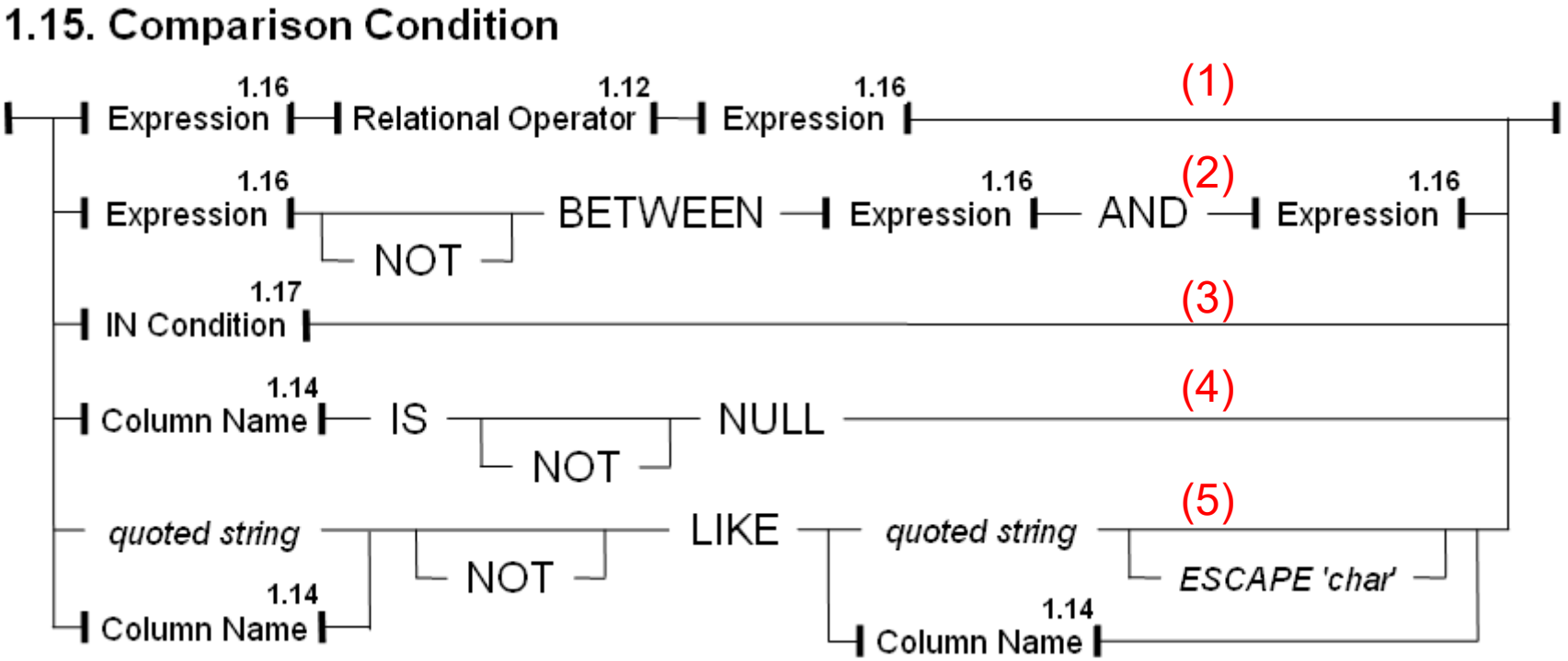
Condition (ponavljanje)

- **`SELECT` *`SELECT List`* `FROM` *`table`* `[WHERE` *`Condition`*`]`**
- Uvjet (*Condition*) se sastoji od operanada, operatora i zagrada
  - operandi su:
    - imena atributa iz relacije *table*
    - konstante
  - operatori su:
    - operatori usporedbe: **`<`** **`<=`** **`=`** **`<>`** **`>`** **`>=`**
    - logički operatori: **`AND`** **`OR`** **`NOT`**

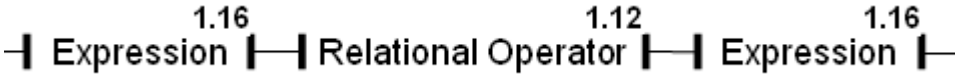
- SQL omogućava dodatne oblike za opisivanje uvjeta

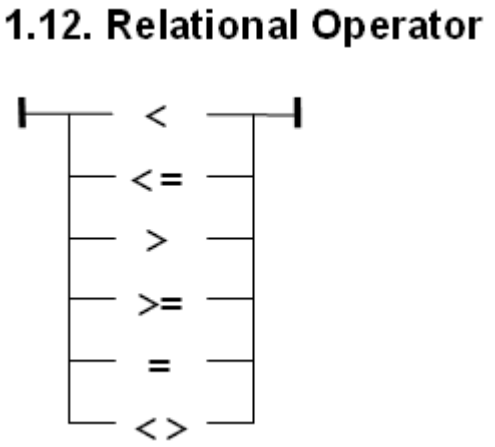


Uvjet usporedbe (*Comparison Condition*)



Uvjet usporedbe (*Comparison Condition*) (1)





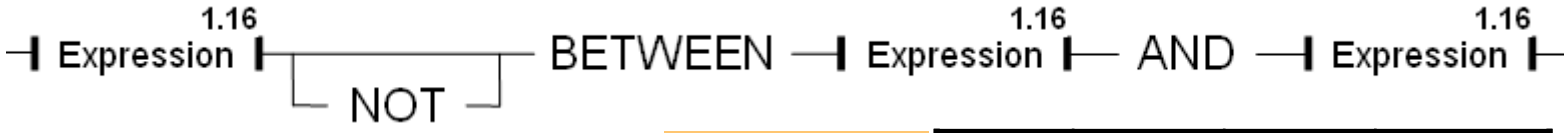
student

matBr	ime	prez	postBr
100	Ivan	Kolar	52000
102	Ana	Horvat	10000
105	Jura	NULL	21000

```
SELECT * FROM student
  WHERE prez <> 'Kolar';
```

matBr	ime	prez	postBr
102	Ana	Horvat	10000

Uvjet usporedbe (*Comparison Condition*) (2)



stanjeSklad

sifArt	minS	maxS	stanje
1	10	50	50
2	20	60	30
3	10	80	5
4	NULL	10	15
5	10	20	NULL

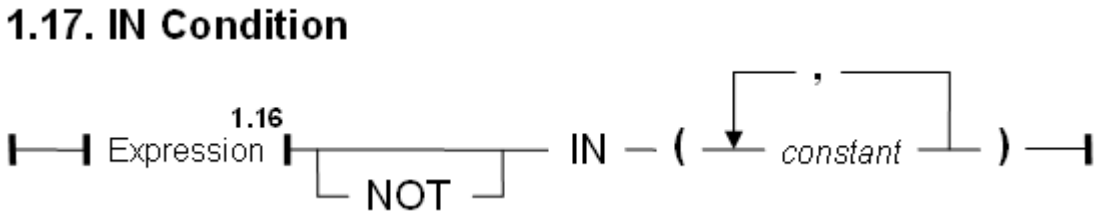
```
SELECT * FROM stanjeSklad
  WHERE stanje BETWEEN minS AND maxS;
```

sifArt	minS	maxS	stanje
1	10	50	50
2	20	60	30

```
SELECT * FROM stanjeSklad
  WHERE stanje NOT BETWEEN minS AND maxS;
```

sifArt	minS	maxS	stanje
3	10	80	5
4	NULL	10	15

Uvjet usporedbe (*Comparison Condition*) (3)



student

matBr	prez
100	Kolar
102	Horvat
103	Novak
105	Horvat
107	NULL
109	Ban

```
SELECT * FROM student
  WHERE prez IN ('Kolar', 'Horvat');
```

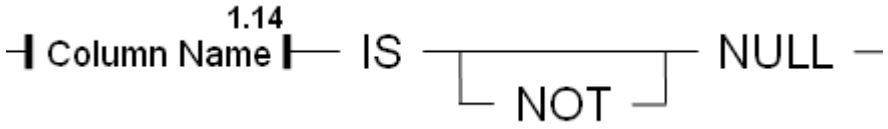
matBr	prez
100	Kolar
102	Horvat
105	Horvat

```
SELECT * FROM student
  WHERE prez NOT IN ('Kolar', 'Horvat');
```

matBr	prez
103	Novak
109	Ban

- ako *Expression* ima vrijednost NULL, tada je rezultat logička vrijednost *unknown*, bez obzira na vrijednosti navedene u skupu

Uvjet usporedbe (*Comparison Condition*) (4)



student

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
105	Novak	NULL
107	Ban	10000

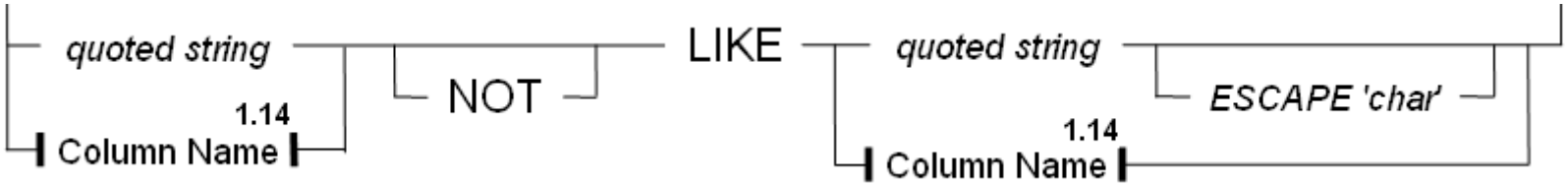
```
SELECT * FROM student
  WHERE postBr IS NULL;
```

matBr	prez	postBr
105	Novak	NULL

```
SELECT * FROM student
  WHERE postBr IS NOT NULL;
```

matBr	prez	postBr
100	Kolar	52000
102	Horvat	10000
107	Ban	10000

Uvjet usporedbe (*Comparison Condition*) (5)



- služi za ispitivanje zadovoljava li (ili ne zadovoljava) vrijednost atributa ili znakovna konstanta zadani uzorak (*pattern*)
- mogu se koristiti sljedeći *wildcard* znakovi:
  - znak % zamjenjuje bilo koju kombinaciju znakova (0 ili više znakova)
  - znak _ zamjenjuje točno jedan znak

Uvjet usporedbe (*Comparison Condition*) (5)

osoba

matBr	ime
1	Matija
2	Metka
3	Matilda
4	Ratkec
5	Marko
6	Ivan

```
SELECT * FROM osoba
  WHERE ime LIKE 'M%';
```

matBr	ime
1	Matija
2	Metka
3	Matilda
5	Marko

```
SELECT * FROM osoba
  WHERE ime LIKE 'Mat%';
```

matBr	ime
1	Matija
3	Matilda

```
SELECT * FROM osoba
  WHERE ime LIKE 'Ma_k%';
```

matBr	ime
5	Marko

char:

```
SELECT * FROM osoba
  WHERE TRIM(ime)
        LIKE '%tk_';
```

varchar:

```
SELECT * FROM osoba
  WHERE ime LIKE '%tk_';
```

matBr	ime
2	Metka

```
SELECT * FROM osoba
  WHERE ime LIKE '%tk%';
```

matBr	ime
2	Metka
4	Ratkec

Uvjet usporedbe (*Comparison Condition*) (5)

tekstovi

rbr	tekst
1	deset %
2	pet % kisika
3	nije pet
4	nije_pet
5	% i _

- znak *char* naveden iza ESCAPE služi za poništavanje specijalnog značenja znakova % ili _ koji su navedeni neposredno iza znaka *char*

```
SELECT * FROM tekstovi
  WHERE tekst LIKE '#%%'
  ESCAPE '#';
```

rbr	tekst
5	% i _

```
SELECT * FROM tekstovi
  WHERE tekst LIKE '%$%'
  ESCAPE '$';
```

rbr	tekst
1	deset %

```
SELECT * FROM tekstovi
  WHERE tekst LIKE '%$%%'
  ESCAPE '$';
```

rbr	tekst
1	deset %
2	pet % kisika
5	% i _

```
SELECT * FROM tekstovi
  WHERE tekst LIKE '%!_pet'
  ESCAPE '!';
```

rbr	tekst
4	nije_pet

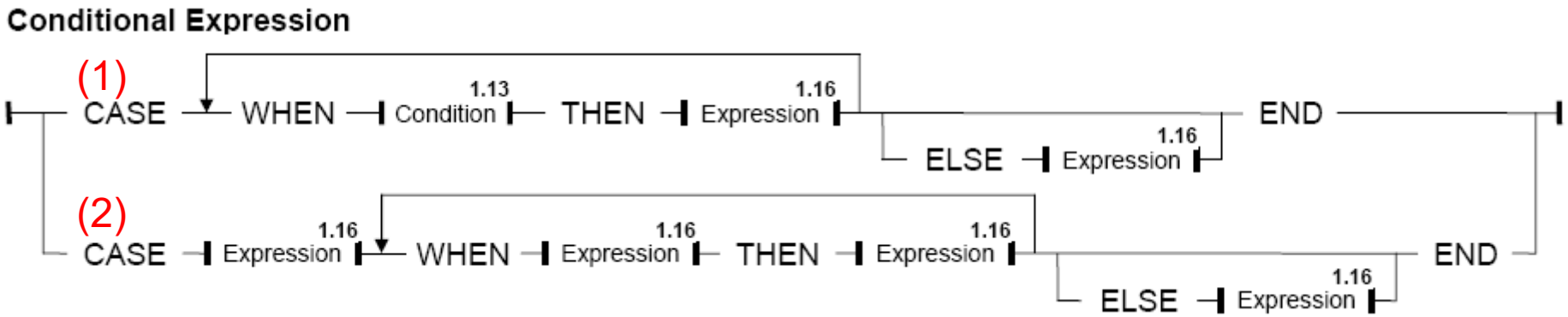
Uvjet usporedbe (*Comparison Condition*) (5)

- x LIKE 'AB% ' — *true* za svaki x koji započinje s AB
- x LIKE '%AB' — *true* za svaki x koji završava* s AB
- x LIKE '%%AB' — *true* za svaki x koji završava* s AB

- x LIKE 'AB%CD' — *true* za svaki x koji započinje s AB i završava* s CD
- x LIKE '%AB%' — *true* za svaki x koji sadrži AB
- x LIKE '_AB' — *true* za svaki x duljine 3 znaka koji završava* s AB
- x LIKE '__AB' — *true* za svaki x duljine 4 znaka koji završava* s AB
- x LIKE 'AB__' — *true* za svaki x duljine* 4 znaka koji započinje s AB
- x LIKE '_AB%' — *true* za svaki x koji započinje bilo kojim znakom, nastavlja se sa znakovima AB, te završava s bilo kojim znakovima

*ako je x varchar, inače trim(x)

Uvjetni izraz (*Conditional Expression*)



- 1. oblik izraza je sličan **`if • else if • else`** naredbi za višestranu selekciju u programskom jeziku C
- 2. oblik izraza je sličan **`switch • case • default`** naredbi za selekciju u programskom jeziku C
- pri čemu postoji bitna razlika:
  - C naredbama "odlučuje se" koje će se naredbe obaviti
  - SQL uvjetnim izrazom "odlučuje se" koja **vrijednost** predstavlja **rezultat** uvjetnog izraza

Uvjetni izraz (*Conditional Expression*) (1)

```
SELECT *
  , CASE
      WHEN ocjena = 5 THEN 'izvrstan'
      WHEN ocjena = 4 THEN 'vrlo dobar'
      WHEN ocjena = 3 THEN 'dobar'
      WHEN ocjena = 2 THEN 'dovoljan'
      WHEN ocjena = 1 THEN 'nedovoljan'
      WHEN ocjena IS NULL THEN 'nepoznato'
      ELSE 'neispravno'
    END AS opis
  FROM ispit;
```

ispit

matBr	ocjena
100	5
102	3
103	1
107	NULL
109	6

matBr	ocjena	opis
100	5	izvrstan
102	3	dobar
103	1	nedovoljan
107	NULL	nepoznato
109	6	neispravno

- ako se više izraza uz WHEN izračuna kao *true*, rezultat izraza je *Expression* naveden uz prvi WHEN čiji se uvjet izračuna kao *true*
- ako se ELSE dio izraza ne navede, a niti jedan uvjet uz WHEN se ne izračuna kao *true*, tada je rezultat izraza NULL vrijednost

Uvjetni izraz (*Conditional Expression*) (2)

```
SELECT *
  , CASE ocjena
      WHEN 5 THEN 'izvrstan'
      WHEN 4 THEN 'vrlo dobar'
      WHEN 3 THEN 'dobar'
      WHEN 2 THEN 'dovoljan'
      WHEN 1 THEN 'nedovoljan'
      ELSE 'neispravno'
    END AS opis
  FROM ispit;
```

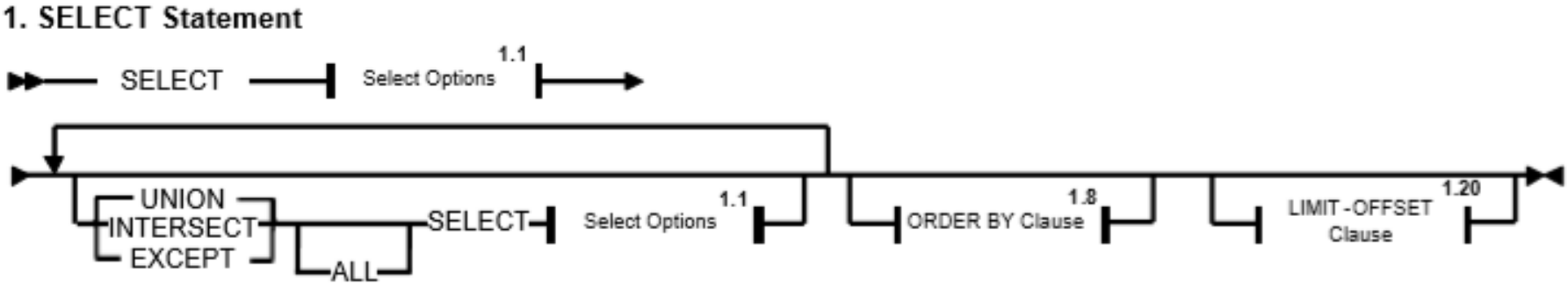
ispit

matBr	ocjena
100	5
102	3
103	1
107	NULL
109	6

matBr	ocjena	opis
100	5	izvrstan
102	3	dobar
103	1	nedovoljan
107	NULL	neispravno
109	6	neispravno

- ako više izraza uz WHEN zadovoljava uvjet jednakosti, rezultat izraza je *Expression* naveden uz prvi WHEN koji zadovoljava uvjet
- ako se ELSE dio izraza ne navede, a niti jedan izraz ne zadovoljava uvjet jednakosti, tada je rezultat izraza NULL vrijednost

Unija, presjek i razlika



- *SELECT Statement* može se graditi od jednog ili više *SELECT dijelova*
- UNION, INTERSECT, EXCEPT - uz izbacivanje duplikata (kopija n-torki)
- UNION ALL, INTERSECT ALL, EXCEPT ALL- bez izbacivanja duplikata (kopija n-torki)
- imena stupaca (atributa rezultantne relacije) određuju se na temelju imena stupaca iz prvog navedenog *SELECT dijela*
- poredak atributa u različitim SELECT dijelovima mora biti jednak
- korespondentni atributi / izrazi moraju odgovarati po tipu podatka i po značenju – unijska kompatibilnost

Unija (*UNION*)

- polozioMat ∪ polozioProg ∪ polozioDiglog

polozioMat

mbr	imeSt	prezSt
100	Ivan	NULL
102	Ana	Novak
105	Rudi	Kolar
111	Jura	Horvat

polozioProg

mbr	ime	prez
100	Ivan	NULL
103	NULL	Ban
105	Rudi	Kolar

polozioDiglog

mbr	ime	prez
102	Ana	Novak
103	NULL	Ban
105	Rudi	Kolar
111	Jura	Horvat

```
SELECT * FROM polozioMat
   UNION
SELECT * FROM polozioProg
   UNION
SELECT * FROM polozioDiglog;
```

mbr	imeSt	prezSt
100	Ivan	NULL
102	Ana	Novak
103	NULL	Ban
105	Rudi	Kolar
111	Jura	Horvat

Unija – multisetovi (*UNION ALL*)

- rezultat sljedeće naredbe nije relacija!

polozioMat

mbr	ime	prez
100	Ivan	NULL
102	Ana	Novak
105	Rudi	Kolar
111	Jura	Horvat

polozioProg

mbr	imeSt	prezSt
100	Ivan	NULL
103	NULL	Ban
105	Rudi	Kolar

polozioDiglog

mbr	imeSt	prezSt
102	Ana	Novak
103	NULL	Ban
105	Rudi	Kolar
111	Jura	Horvat

```
SELECT * FROM polozioMat
   UNION ALL
SELECT * FROM polozioProg
   UNION ALL
SELECT * FROM polozioDiglog;
```

mbr	ime	prez
100	Ivan	NULL
102	Ana	Novak
105	Rudi	Kolar
111	Jura	Horvat
100	Ivan	NULL
103	NULL	Ban
105	Rudi	Kolar
102	Ana	Novak
103	NULL	Ban
105	Rudi	Kolar
111	Jura	Horvat

Unija (*UNION*)

- naredba je ispravna ako su korespondentni atributi istih tipova podataka (INTEGER-INTEGER, CHAR-CHAR, ...), ali odgovornost je korisnika (programera) voditi računa o unijskoj kompatibilnosti
- npr. sljedeća naredba će se obaviti, ali rezultat je besmislen

pecivo

oznaka	naziv
ZE-33	Žemlja s makom
PR-3	Perec sa sezamom

zrakoplov

oznaka	naziv
PR-3	Piper J-3 Cub
B-747	Boeing 747
A-360	Airbus 360

```
SELECT * FROM pecivo
   UNION
SELECT * FROM zrakoplov;
```

oznaka	naziv
ZE-33	Žemlja s makom
PR-3	Perec sa sezamom
PR-3	Piper J-3 Cub
B-747	Boeing 747
A-360	Airbus 360

Piper J-3 Cub

Perec sa sezamom

Presjek (INTERSECT)

polozioMatem

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat

polozioProgr

mbr	ime	prez
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat

studenti koji su položili **i** Matematiku **i** Programiranje

polozioMatem ∩ polozioProgr

```
SELECT * FROM polozioMatem
  INTERSECT
SELECT * FROM polozioProgr;
```

mbr	ime	prez
102	Ana	Novak
107	Jura	Horvat

Presjek – multisetovi (INTERSECT ALL)

polagaoMatem

mbr	ime	prez
100	Ivan	Kolar
100	Ivan	Kolar
102	Ana	Novak
102	Ana	Novak
102	Ana	Novak
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat
107	Jura	Horvat

polagaoProgr

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat
107	Jura	Horvat
107	Jura	Horvat

Ispisati podatke o studentima onoliko puta koliko puta su polagali **i** Matematiku **i** Programiranje

polagaoMatem $\cap_{ALL}$ polagaoProgr

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
102	Ana	Novak
107	Jura	Horvat
107	Jura	Horvat

```
SELECT * FROM polagaoMatem
  INTERSECT ALL
SELECT * FROM polagaoProgr;
```

Razlika (EXCEPT)

polozioMatem

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat

polozioProgr

mbr	ime	prez
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat

studenti koji su položili Matematiku, ali nisu položili Programiranje

polozioMatem \ polozioProgr

```
SELECT * FROM polozioMatem
  EXCEPT
SELECT * FROM polozioProgr
```

mbr	ime	prez
100	Ivan	Kolar
103	Tea	Ban

Razlika – multisetovi (EXCEPT ALL)

polagaoMatem

mbr	ime	prez
100	Ivan	Kolar
100	Ivan	Kolar
102	Ana	Novak
102	Ana	Novak
102	Ana	Novak
102	Ana	Novak
103	Tea	Ban
107	Jura	Horvat
107	Jura	Horvat

polagaoProgr

mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat
107	Jura	Horvat
107	Jura	Horvat

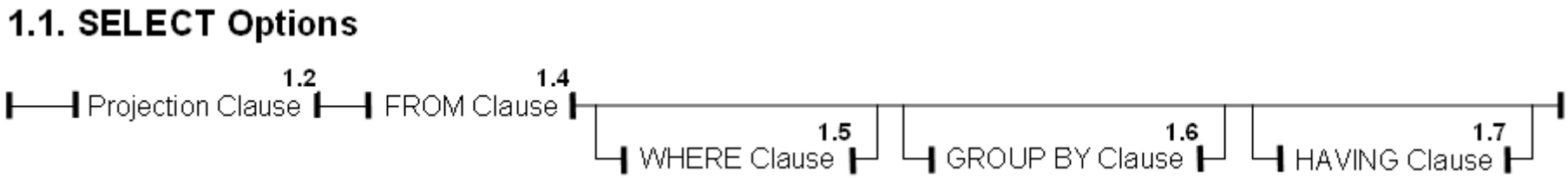
studenti koji su polagali Matematiku više puta nego što su polagali Programiranje

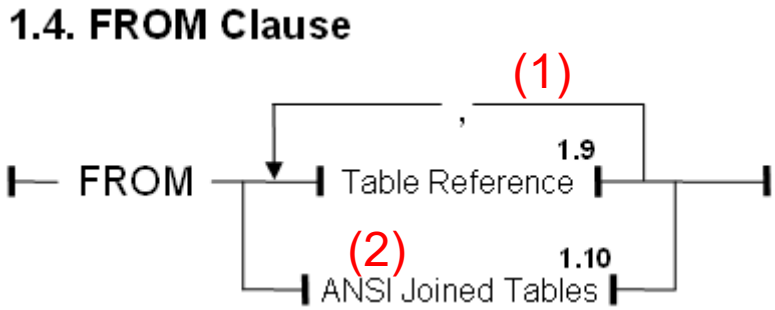
polagaoMatem \ $_{ALL}$ polagaoProgr

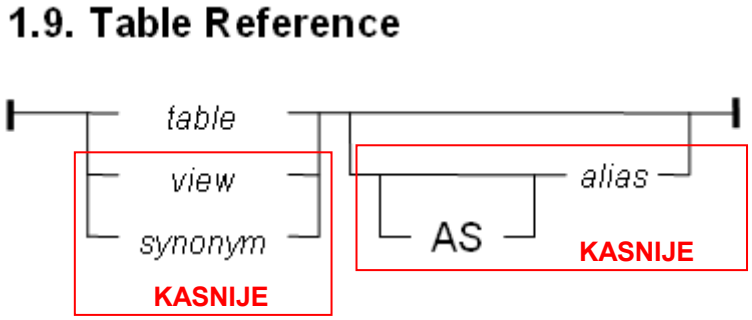
mbr	ime	prez
100	Ivan	Kolar
102	Ana	Novak
102	Ana	Novak
103	Tea	Ban

```
SELECT * FROM polagaoMatem
  EXCEPT ALL
SELECT * FROM polagaoProgr;
```

FROM Clause





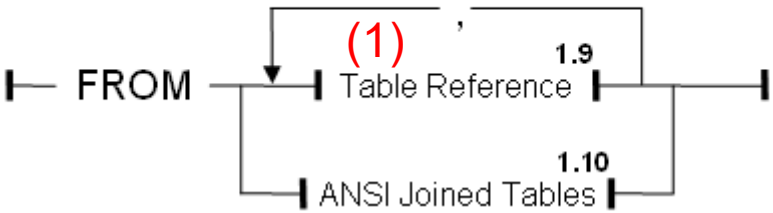


(1) klasična sintaksa (*classical*, *comma-delimited*) za spajanje relacija

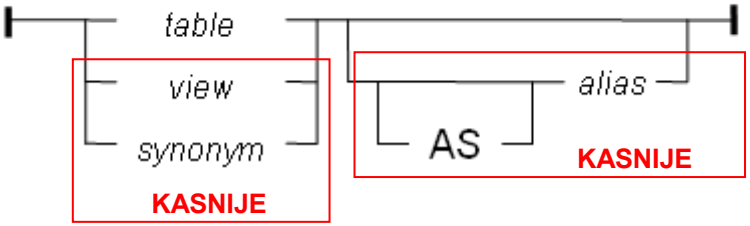
(2) ANSI sintaksa za spajanje relacija

FROM Clause (1)

1.4. FROM Clause



1.9. Table Reference



- klasična sintaksa (*classical*, *comma-delimited*) može se koristiti za obavljanje operacija:
  - Kartezijev produkt
  - spajanje uz uvjet i spajanje s izjednačavanjem
  - prirodno spajanje
- uvjeti spajanja se navode u WHERE dijelu SELECT naredbe, zajedno s eventualnim uvjetima selekcije (uvjeti spajanja i uvjeti selekcije se u tom slučaju povezuju logičkim operatorom AND)

FROM Clause (1)

- Zadane su relacije: r ({ A, B }) s ({ C, D }) t ({ D, E })

$r \times s$

```
SELECT *
  FROM r, s;
```

$r \underset{A=C \wedge B \geq D}{\rhd\lhd} s$

```
SELECT *
  FROM r, s
  WHERE A = C
    AND B >= D;
```

$r \underset{B=C}{\rhd\lhd} s$

```
SELECT *
  FROM r, s
  WHERE B = C;
```

$\sigma_{D>5}(r \underset{B=C}{\rhd\lhd} s)$

```
SELECT *
  FROM r, s
  WHERE B = C
    AND D > 5;
```

FROM Clause (1)

- r ({ A, B }) s ({ C, D }) t ({ D, E })

$(r \times s) \rhd\lhd t$

```
SELECT r.*, s.*, t.E
  FROM r, s, t
  WHERE s.D = t.D;
```

$(r \rhd\lhd s) \rhd\lhd t$

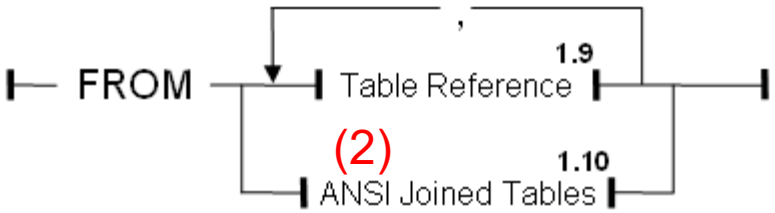
```
SELECT r.*, s.*, t.E
  FROM r, s, t
  WHERE s.D = t.D;
```

$\sigma_{C=100}(s \rhd\lhd t)$

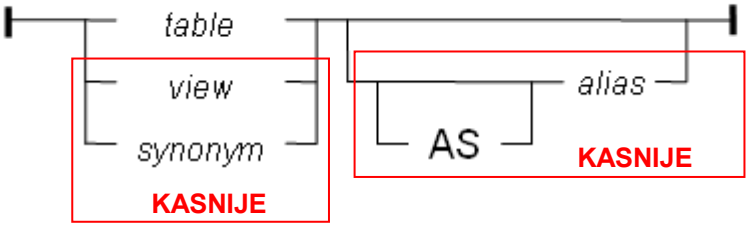
```
SELECT s.*, t.E
  FROM s, t
  WHERE s.D = t.D
    AND C = 100;
```

FROM Clause (2)

1.4. FROM Clause



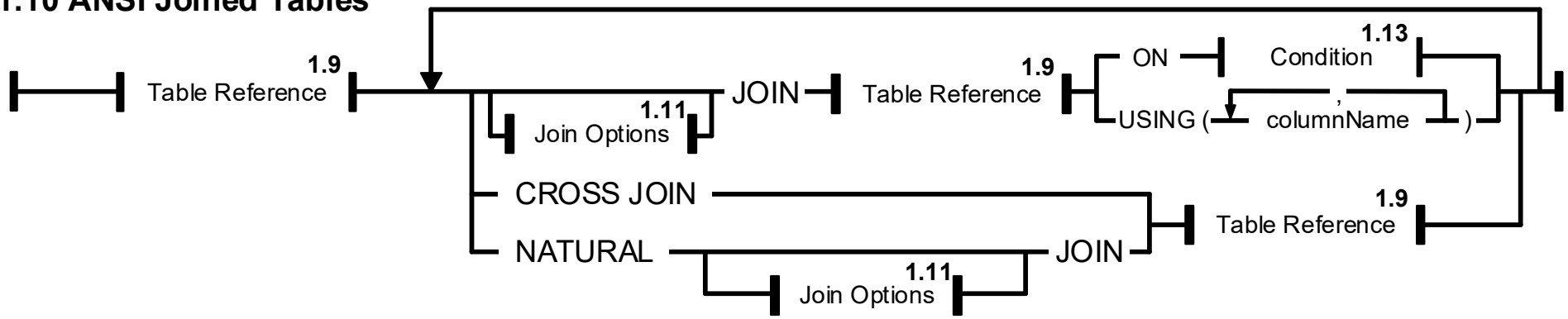
1.9. Table Reference



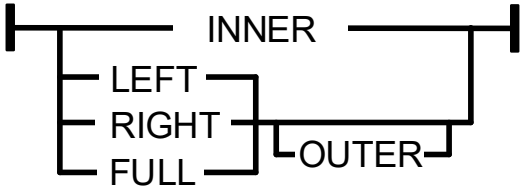
- ANSI sintaksa za spajanje relacija. Može se koristiti za obavljanje operacija:
  - Kartezijev produkt
  - spajanje uz uvjet i spajanje s izjednačavanjem
  - prirodno spajanje
  - **vanjsko spajanje**
    - vanjsko spajanje uz uvjet
    - vanjsko spajanje s izjednačavanjem
    - prirodno vanjsko spajanje

FROM Clause (2)

1.10 ANSI Joined Tables



1.11 Join Options




```
r CROSS JOIN s
r NATURAL JOIN s
r NATURAL LEFT JOIN s
r NATURAL RIGHT JOIN s
r NATURAL FULL JOIN s
r INNER JOIN s ON uvjetSpajanja
r LEFT OUTER JOIN s ON uvjetSpajanja
r RIGHT OUTER JOIN s ON uvjetSpajanja
r FULL OUTER JOIN s ON uvjetSpajanja
```

FROM Clause (2)

- Specifično za PostgreSQL – prirodna spajanja - unutarnja i vanjska:

```
r INNER JOIN s USING (atr1, atr2, …)
r LEFT OUTER JOIN s USING (atr1, atr2, …)
r RIGHT OUTER JOIN s USING (atr1, atr2, …)
r FULL OUTER JOIN s USING (atr1, atr2, …)
```

FROM Clause (2)

- Rezervirane riječi OUTER i INNER se smiju izostaviti:

```
r INNER JOIN s          ≡  r JOIN s
r LEFT OUTER JOIN s     ≡  r LEFT JOIN s
r RIGHT OUTER JOIN s    ≡  r RIGHT JOIN s
r FULL OUTER JOIN s     ≡  r FULL JOIN s
```

FROM Clause (2)

- Zadane su relacije: r ({ A, B }) s ({ C, D }) t ({ D, E })

$r \times s$

```
SELECT *
  FROM r CROSS JOIN s;
```

$r \rhd\lhd_{A=C \wedge B \geq D} s$

```
SELECT *
  FROM r
       INNER JOIN s
         ON A = C AND B >= D;
```

$r \rhd\lhd_{B=C} s$

```
SELECT *
  FROM r
       INNER JOIN s
         ON B = C;
```

$\sigma_{D>5}(r \rhd\lhd_{B=C} s)$

```
SELECT *
  FROM r
       INNER JOIN s
         ON B = C
  WHERE D > 5;
```

FROM Clause (2)

- r ({ A, B }) s ({ C, D }) t ({ D, E }) p ({ E, F })

$(r \times s) \rhd\lhd t$

```
SELECT r.*, s.*, t.E
  FROM r
       CROSS JOIN s
       INNER JOIN t
         ON s.D = t.D;
```

```
SELECT r.*, s.*, t.E
  FROM r
       CROSS JOIN s
       NATURAL JOIN t;
```

$(s \rhd\lhd t) \rhd\lhd p$

```
SELECT s.*, t.E, p.F
  FROM s
       INNER JOIN t
         ON s.D = t.D
       INNER JOIN p
         ON t.E = p.E;
```

```
SELECT s.*, t.E, p.F
  FROM s
    NATURAL JOIN t
       NATURAL JOIN p;
```

$\sigma_{C=100}(s \rhd\lhd t)$

```
SELECT s.*, t.E
  FROM s
       INNER JOIN t
         ON s.D = t.D
  WHERE C = 100;
```

```
SELECT s.*, t.E
  FROM s
       NATURAL JOIN t
  WHERE C = 100;
```

FROM Clause (2)

- r ({ A, B }) s ({ C, D }) t ({ D, E }) p ({ E, F })

$(r \times s)\ {}^{*}\triangleright\triangleleft\ t$

```
SELECT r.*, s.*, t.E
  FROM r
    CROSS JOIN s
    LEFT OUTER JOIN t
      ON s.D = t.D;
```

```
SELECT r.*, s.*, t.E
  FROM r
    CROSS JOIN s
    NATURAL LEFT JOIN t;
```

$(s\ {}^{*}\triangleright\triangleleft^{*}\ t)\ \triangleright\triangleleft^{*}\ p$

```
SELECT s.*, t.D D1, p.*
  FROM s
    FULL OUTER JOIN t
      ON s.D = t.D
    RIGHT OUTER JOIN p
      ON t.E = p.E;
```

```
SELECT s.*, t.D D1, p.*
  FROM s
   NATURAL FULL JOIN t
   NATURAL RIGHT JOIN p;
```

$\sigma_{C=100}(s\ {}^{*}\triangleright\triangleleft\ t)$

```
SELECT s.*, t.E
  FROM s
    LEFT OUTER JOIN t
      ON s.D = t.D
  WHERE C = 100;
```

```
SELECT s.*, t.E
  FROM s
    NATURAL LEFT JOIN t
WHERE C = 100;
```

FROM Clause (2)

- Ako se obavlja operacija spajanja i selekcija, uvjete spajanja treba navesti u ON dijelu, a uvjete selekcije treba navesti u WHERE dijelu SELECT naredbe
  - iako, u slučaju kad nema vanjskog spajanja, rezultat upita ne ovisi o tome je li uvjet selekcije naveden u ON ili WHERE dijelu naredbe

student

matBr	prez	pbrSt
101	Kolar	10000
102	Horvat	21000
103	Novak	NULL

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT *
  FROM student
  JOIN mjesto
    ON pbrSt = pbr
 WHERE prez = 'Kolar';
```

```
SELECT *
  FROM student
  JOIN mjesto
    ON pbrSt = pbr
   AND prez = 'Kolar';
```

$\sigma_{prez = \text{'Kolar'}}(\text{student} \underset{pbrst = pbr}{\triangleright\!\triangleleft} \text{mjesto})$

$\text{student} \underset{pbrst = pbr \wedge prez = \text{'Kolar'}}{\triangleright\!\triangleleft} \text{mjesto}$

- oba upita daju isti rezultat

matBr	prez	pbrSt	pbr	nazMjesto
101	Kolar	10000	10000	Zagreb

FROM Clause (2)

- Ako se koristi vanjsko spajanje, navođenje uvjeta selekcije u ON dijelu umjesto WHERE dijelu može **bitno** utjecati na rezultat

student

matBr	prez	pbrSt
101	Kolar	10000
102	Horvat	21000
103	Novak	NULL

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT *
  FROM student
  LEFT JOIN mjesto
         ON pbrSt = pbr
  WHERE prez = 'Kolar';
```

$$\sigma_{prez = \text{'Kolar'}}(student \overset{pbrst = pbr}{*\triangleright\triangleleft} mjesto)$$

- Tek nakon obavljenog spajanja prema uvjetu navedenom u ON dijelu naredbe, obavlja se selekcija n-torki prema uvjetu navedenom u WHERE dijelu naredbe

matBr	prez	pbrSt	pbr	nazMjesto
101	Kolar	10000	10000	Zagreb

FROM Clause (2)

- Ovdje je prikazan upit sličan prethodnom, ali u kojem je uvjet selekcije napisan na "pogrešnom" mjestu

student

matBr	prez	pbrSt
101	Kolar	10000
102	Horvat	21000
103	Novak	NULL

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT *
  FROM student
  LEFT JOIN mjesto
         ON pbrSt = pbr
         AND prez = 'Kolar';
```

student *⟕ mjesto

pbrst = pbr ∧ prez = ‘Kolar’

- Ovdje će se pojaviti sve n-torke iz relacije student - uz one n-torke relacije student koje ne zadovoljavaju uvjet spajanja (uočite koji je uvjet spajanja ovdje naveden) dodat će se NULL vrijednosti

matBr	prez	pbrSt	pbr	nazMjesto
101	Kolar	10000	10000	Zagreb
102	Horvat	21000	NULL	NULL
103	Novak	NULL	NULL	NULL

FROM Clause (2)

- **Logički promatrano***, kada se u upitu spajaju više od dvije relacije, redoslijed spajanja je slijeva na desno: spajaju se prve dvije relacije, zatim se dobiveni rezultat spaja s trećom navedenom relacijom, zatim se dobiveni rezultat spaja s četvrtom navedenom relacijom, itd.

(*) konačni rezultat će sigurno odgovarati rezultatu koji bi se dobio kada bi se relacije spajale s lijeva na desno. Fizički promatrano, upit će se možda izvesti drugačijim redoslijedom, ali o tome brine dio SUBP-a koji se naziva optimizator upita

FROM Clause (2)

- ako se ne koristi vanjsko spajanje, redoslijed spajanja je ionako irelevantan, jer vrijedi:

$$(r_1 \rhd\lhd r_2) \rhd\lhd r_3 \quad \equiv \quad r_1 \rhd\lhd (r_2 \rhd\lhd r_3)$$

- ako se koristi vanjsko spajanje, redoslijed spajanja jest važan jer:

$$(r_1 \ ^*\rhd\lhd r_2) \rhd\lhd r_3 \quad \neq \quad r_1 \ ^*\rhd\lhd (r_2 \rhd\lhd r_3)$$

student

mBr	prez	pbr
101	Kolar	10000
102	Horvat	21000
103	Novak	NULL

mjesto

pbr	nazMj	sifZup
10000	Zagreb	21
21000	Split	17

zupanija

sifZup	nazZup
21	Grad Zagreb
17	Splitsko-dalmatinska

mBr	prez	pbr	nazMjesto	sifZup	nazZup
101	Kolar	10000	Zagreb	21	Grad Zagreb
102	Horvat	21000	Split	17	Splitsko-dalmatinska

$r_1 \ ^*\rhd\lhd (r_2 \rhd\lhd r_3)$

$(r_1 \ ^*\rhd\lhd r_2) \rhd\lhd r_3$

mBr	prez	pbr	nazMjesto	sifZup	nazZup
101	Kolar	10000	Zagreb	21	Grad Zagreb
102	Horvat	21000	Split	17	Splitsko-dalmatinska
103	Novak	NULL	NULL	NULL	NULL

FROM Clause (2)

stud

mbr	prez	pbrSt
101	Horvat	42000
102	Novak	21000

mjesto

pbr	nazMjesto	sifZupMj
42000	Varaždin	7
21000	Split	NULL

zupanija

sifZup	nazZup
7	Varaždinska
4	Istarska

$$(\text{stud} \overset{*}{\triangleright\!\triangleleft}_{\text{pbrSt=pbr}} \text{mjesto}) \triangleright\!\triangleleft_{\text{sifZupMj=sifZup}} \text{zupanija}$$

```
SELECT stud.*, mjesto.*, zupanija.*
  FROM stud
       LEFT OUTER JOIN mjesto
         ON pbrSt = pbr
       INNER JOIN zupanija
         ON sifZupMj = sifZup;
```

- prvo se spajaju relacije stud i mjesto, a zatim se dobiveni rezultat spaja s relacijom zupanija

mbr	prez	pbrSt	pbr	nazMjesto	sifZupMj	sifZup	nazZup
101	Horvat	42000	42000	Varaždin	7	7	Varaždinska

FROM Clause (2)

$$stud \overset{*}{\triangleright\!\triangleleft}_{pbrSt=pbr} (mjesto \overset{}{\triangleright\!\triangleleft}_{sifZupMj=sifZup} zupanija)$$

da bismo izraz relacijske algebre mogli napisati u obliku SQL naredbe, napisat ćemo ga u drugačijem obliku

$$\equiv (mjesto \overset{}{\triangleright\!\triangleleft}_{sifZupMj=sifZup} zupanija) \overset{*}{\triangleright\!\triangleleft}_{pbrSt=pbr} stud$$

```
SELECT stud.*, mjesto.*, zupanija.*
  FROM mjesto
       INNER JOIN zupanija
         ON sifZupMj = sifZup
       RIGHT OUTER JOIN stud
         ON pbrSt = pbr;
```

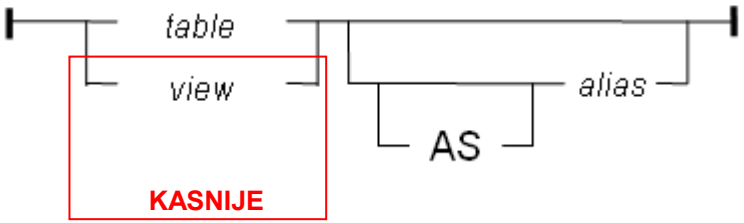
- prvo se spajaju relacije mjesto i zupanija, a zatim se dobiveni rezultat spaja s relacijom stud

mbr	prez	pbrSt	pbr	nazMjesto	sifZupMj	sifZup	nazZup
101	Horvat	42000	42000	Varaždin	7	7	Varaždinska
102	Novak	21000	NULL	NULL	NULL	NULL	NULL

Preimenovanje relacija unutar upita

- relacija se unutar upita može preimenovati u *alias* ime
  - *alias* ime je vidljivo samo unutar upita (ne utječe na stvarno ime relacije u bazi podataka)

1.9. Table Reference



- rezervirana riječ AS se smije ispustiti
- na relaciju koja je u upitu dobila *alias* ime, moguće je referencirati se isključivo preko tog istog *alias* imena

```
SELECT nazMjesto, nazZupanija
  FROM mjesto AS town
     , zupanija AS county
  WHERE town.sifZupanija = county.sifZupanija;
```

```
SELECT nazMjesto, nazZupanija
  FROM mjesto AS town
       JOIN zupanija AS county
         ON town.sifZupanija = county.sifZupanija;
```

Preimenovanje relacija unutar upita

- iako se preimenovanjem relacija može skratiti duljina teksta upita, u praksi se to **ne preporuča** jer upiti postaju manje razumljivi

```
SELECT o.jmbg, prezime, m.pbr, nazMjesto
   FROM osoba AS o
      , mjesto AS m
      , zaposlenje AS z1
      , zupanija AS z2
   WHERE o.jmbg = z1.jmbg
     AND o.pbr = m.pbr
     AND m.sifZup = z2.sifZup
     AND z2.nazZup = 'Varaždinska'
     AND z1.radnoMjesto = 'Dimnjačar'
```

- preimenovanje relacija unutar upita treba se koristiti onda kada se ista relacija pojavljuje u više uloga unutar istog upita

Paralelno spajanje

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
103	Ban	10000	10000

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

- Kako dobiti sljedeći rezultat:

mbr	prez	pbrRod	pbrStan	nazMjestoR
100	Kolar	10000	21000	Zagreb
102	Novak	21000	10000	Split
103	Ban	10000	10000	Zagreb

- To je lako:

```
SELECT student.*, mjesto.nazMjesto AS nazMjestoR
  FROM student
     , mjesto
  WHERE student.pbrRod = mjesto.pbr;
```

Paralelno spajanje

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
103	Ban	10000	10000

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

- Kako dobiti sljedeći rezultat:

mbr	prez	pbrRod	nazMjestoR	pbrStan	nazMjestoS
100	Kolar	10000	Zagreb	21000	Split
102	Novak	21000	Split	10000	Zagreb
103	Ban	10000	Zagreb	10000	Zagreb

Paralelno spajanje

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
103	Ban	10000	10000

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT mbr, prez
     , pbrRod, mjesto.nazMjesto AS nazMjestoR
     , pbrStan, mjesto.nazMjesto AS nazMjestoS
  FROM student
     , mjesto
  WHERE student.pbrRod = mjesto.pbr
    AND student.pbrStan = mjesto.pbr;
```

NEISPRAVNO RJEŠENJE

mbr	prez	pbrRod	nazMjestoR	pbrStan	nazMjestoS
103	Ban	10000	Zagreb	10000	Zagreb

- Upit **nije dobar** jer jednu n-torku iz relacije student pokušavamo spojiti s jednom n-torkom iz relacije mjesto uz sljedeći uvjet spajanja: vrijednost atributa pbrRod, te **istovremeno** i vrijednost atributa pbrStan iz relacije student su jednake vrijednosti atributa pbr iz relacije mjesto

Paralelno spajanje

- Kad bismo načinili dvije kopije relacije mjesto: mjestoR i mjestoS, sa shemama i sadržajem jednakim relaciji mjesto:

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
103	Ban	10000	10000

mjestoR

pbr	nazMjesto
10000	Zagreb
21000	Split

mjestoS

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT mbr, prez
     , pbrRod, mjestoR.nazMjesto AS nazMjestoR
     , pbrStan, mjestoS.nazMjesto AS nazMjestoS
  FROM student, mjestoR, mjestoS
  WHERE student.pbrRod = mjestoR.pbr
    AND student.pbrStan = mjestoS.pbr;
```

- konačni rezultat je ispravan, ali radi se o vrlo lošem rješenju!

Paralelno spajanje

- Ispravno rješenje: za tablicu **mjesto** definiramo dvije različite uloge

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

mjesto AS mjestoR

mjesto AS mjestoS

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
103	Ban	10000	10000

mjestoR

pbr	nazMjesto
10000	Zagreb
21000	Split

mjestoS

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT mbr, prez
     , pbrRod, mjestoR.nazMjesto AS nazMjestoR
     , pbrStan, mjestoS.nazMjesto AS nazMjestoS
  FROM student
     , mjesto AS mjestoR
     , mjesto AS mjestoS
  WHERE student.pbrRod = mjestoR.pbr
    AND student.pbrStan = mjestoS.pbr;
```

- u upitu se **ista** relacija pojavljuje u dvije različite uloge

Paralelno spajanje - ANSI

- Ispravno rješenje:

mjesto

pbr	nazMjesto
10000	Zagreb
21000	Split

mjesto AS mjestoR

mjesto AS mjestoS

student

mbr	prez	pbrRod	pbrStan
100	Kolar	10000	21000
102	Novak	21000	10000
103	Ban	10000	10000

mjestoR

pbr	nazMjesto
10000	Zagreb
21000	Split

mjestoS

pbr	nazMjesto
10000	Zagreb
21000	Split

```
SELECT mbr, prez
     , pbrRod, mjestoR.nazMjesto AS nazMjestoR
     , pbrStan, mjestoS.nazMjesto AS nazMjestoS
  FROM student
       JOIN mjesto AS mjestoR
         ON student.pbrRod = mjestoR.pbr
       JOIN mjesto AS mjestoS
         ON student.pbrStan = mjestoS.pbr;
```

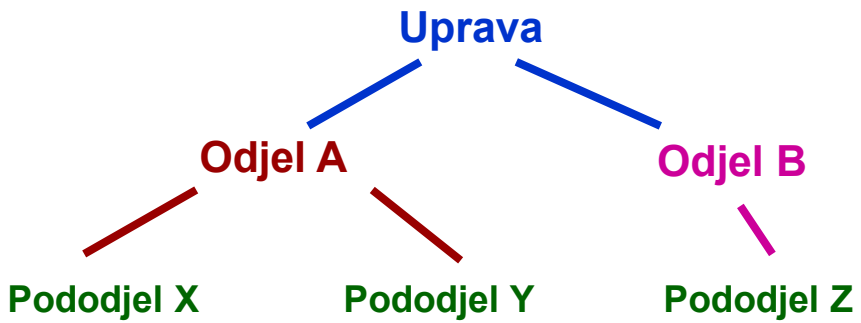
Refleksivno spajanje

- Pojedine n-torke iz relacije povezane su s drugim n-torkama iz **iste** relacije

orgjed

sifOrgjed	nazOrgjed	sifNadorgjed
1	Uprava	**NULL**
2	Odjel A	1
3	Odjel B	1
4	Pododjel X	2
5	Pododjel Y	2
6	Pododjel Z	3

- Uprava nema nadređenu org. jedinicu
- Odjelu A neposredno nadređena jedinica je Uprava
- Odjelu B neposredno nadređena jedinica je Uprava
- Pododjelu X neposredno nadređena jedinica je Odjel A
- itd.



Refleksivno spajanje

- Kako dobiti sljedeći rezultat

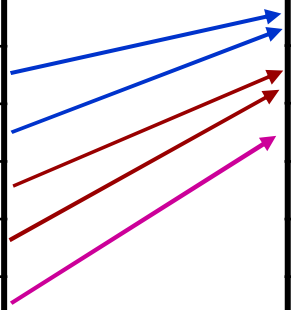
sifOrgjed	nazOrgjed	sifNadorgjed	nazNadorgjed
1	Uprava	**NULL**	**NULL**
2	Odjel A	**1**	**Uprava**
3	Odjel B	**1**	**Uprava**
4	Pododjel X	**2**	**Odjel A**
5	Pododjel Y	**2**	**Odjel A**
6	Pododjel Z	**3**	**Odjel B**

- problem je sličan i slično se rješava kao u slučaju paralelnog spajanja
- radi se o spajanju relacije same sa sobom
- relacija orgjed treba se u upitu pojaviti dva puta, jednom u ulozi organizacijske jedinice, a jednom u ulozi njezine nadređene organizacijske jedinice

Refleksivno spajanje

orgjed

sifOrgjed	nazOrgjed	sifNadorgjed
1	Uprava	**NULL**
2	Odjel A	**1**
3	Odjel B	**1**
4	Pododjel X	**2**
5	Pododjel Y	**2**
6	Pododjel Z	**3**



orgjed (u ulozi nadređene org.jedinice)

sifOrgjed	nazOrgjed	sifNadorgjed
1	Uprava	**NULL**
2	Odjel A	**1**
3	Odjel B	**1**
4	Pododjel X	**2**
5	Pododjel Y	**2**
6	Pododjel Z	**3**

```
SELECT orgjed.sifOrgjed
     , orgjed.nazOrgjed
     , orgjed.sifNadorgjed
     , nadorgjed.nazOrgjed AS nazNadorgjed
  FROM orgjed, orgjed AS nadOrgjed
  WHERE orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;
```

Refleksivno spajanje

```
SELECT orgjed.sifOrgjed
     , orgjed.nazOrgjed
     , orgjed.sifNadorgjed
     , nadorgjed.nazOrgjed AS nazNadorgjed
  FROM orgjed, orgjed AS nadOrgjed
  WHERE orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;
```

sifOrgjed	nazOrgjed	sifNadorgjed	nazNadorgjed
2	Odjel A	1	**Uprava**
3	Odjel B	1	**Uprava**
4	Pododjel X	2	**Odjel A**
5	Pododjel Y	2	**Odjel A**
6	Pododjel Z	3	**Odjel B**

- Nema organizacijske jedinice Uprava? Kako to popraviti?

Refleksivno spajanje

```
SELECT orgjed.sifOrgjed
     , orgjed.nazOrgjed
     , orgjed.sifNadorgjed
     , nadorgjed.nazOrgjed AS nazNadorgjed
  FROM orgjed
       LEFT OUTER JOIN orgjed AS nadOrgjed
         ON orgjed.sifNadorgjed = nadOrgjed.sifOrgjed;
```

sifOrgjed	nazOrgjed	sifNadorgjed	nazNadorgjed
1	Uprava	**NULL**	**NULL**
2	Odjel A	**1**	**Uprava**
3	Odjel B	**1**	**Uprava**
4	Pododjel X	**2**	**Odjel A**
5	Pododjel Y	**2**	**Odjel A**
6	Pododjel Z	**3**	**Odjel B**

Refleksivno spajanje

- Kako dobiti sljedeći rezultat
  - uz svaku organizacijsku jedinicu ispisati nazive neposredno podređenih organizacijskih jedinica
  - ako org. jedinica ima više od jedne podređene org. jedinice, u popisu se pojavljuje više puta
  - u popisu se moraju naći i one organizacijske jedinice koje nemaju niti jednu podređenu organizacijsku jedinicu

sifOrgjed	nazOrgjed	nazPodorgjed
1	Uprava	Odjel A
1	Uprava	Odjel B
2	Odjel A	Pododjel X
2	Odjel A	Pododjel Y
3	Odjel B	Pododjel Z
4	Pododjel X	NULL
5	Pododjel Y	NULL
6	Pododjel Z	NULL

Refleksivno spajanje

orgjed

sifOrgjed	nazOrgjed	sifNadorgjed
1	Uprava	**NULL**
2	Odjel A	**1**
3	Odjel B	**1**
4	Pododjel X	**2**
5	Pododjel Y	**2**
6	Pododjel Z	**3**

orgjed (u ulozi podređene org.jedinice)

sifOrgjed	nazOrgjed	sifNadorgjed
1	Uprava	**NULL**
2	Odjel A	**1**
3	Odjel B	**1**
4	Pododjel X	**2**
5	Pododjel Y	**2**
6	Pododjel Z	**3**

```
SELECT orgjed.sifOrgjed
     , orgjed.nazOrgjed
     , podOrgjed.nazOrgjed AS nazPodorgjed
  FROM orgjed
       LEFT OUTER JOIN orgjed AS podOrgjed
         ON podOrgjed.sifNadorgjed = orgjed.sifOrgjed;
```

Preimenovanje relacija unutar upita

- Još jedan primjer u kojem se koristi preimenovanje relacije
- Ispisati podatke o svim osobama čija je plaća manja od plaće osobe sa šifrom 103

osoba

sifra	ime	prez	placa
100	Ana	Novak	6000
101	Ana	Kolar	5000
102	Ivan	Kolar	3000
103	Ana	Novak	5000
104	Jura	Ban	4000

osoba103

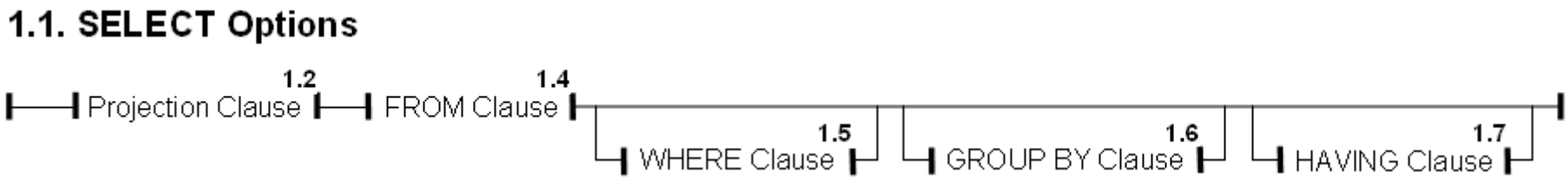
sifra	ime	prez	placa
103	Ana	Novak	5000

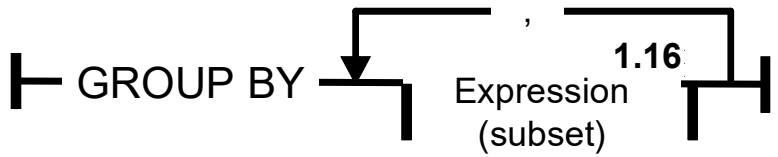
$$\text{osoba} \triangleright\!\triangleleft_{\text{placa} < \text{placa103}} \rho_{\text{osoba103(s,i,p,placa103)}} (\sigma_{\text{sifra} = 103}(\text{osoba}))$$

```
SELECT osoba.*
  FROM osoba
       INNER JOIN osoba AS osoba103
          ON osoba.placa < osoba103.placa
             AND osoba103.sifra = 103;
```

sifra	ime	prez	placa
102	Ivan	Kolar	3000
104	Jura	Ban	4000

GROUP BY Clause





- U GROUP BY dijelu naredbe
  - se navodi jedan ili više **izraza** koji su navedeni u FROM dijelu naredbe
  - nije dozvoljeno koristiti agregatne izraze

GROUP BY Clause

ispit

matBr	nazPredmet	ocjena
100	Matematika	3
100	Programiranje	2
100	Fizika	5
101	Matematika	2
101	Programiranje	2
101	Fizika	3
102	Matematika	4

```
SELECT nazPredmet AS naziv
     , AVG(ocjena) AS prosjek
  FROM ispit
  GROUP BY nazPredmet;
```

naziv	prosjek
Matematika	3
Programiranje	2
Fizika	4

- PostgreSQL dopušta (ali ne i svi ostali SUBP-ovi) u GROUP BY koristiti izraze ili zamjenska imena atributa (*display_label*)

```
SELECT nazPredmet AS naziv
     , AVG(ocjena)
  FROM ispit
  GROUP BY naziv;
```

HAVING Clause

ispit

matBr	nazPredmet	ocjena
100	Matematika	3
100	Programiranje	2
100	Fizika	5
101	Matematika	2
101	Programiranje	2
101	Fizika	3
102	Matematika	4

```
SELECT nazPredmet AS naziv
     , AVG(ocjena) AS prosjek
  FROM ispit
  GROUP BY nazPredmet;
```

naziv	prosjek
Matematika	3
Programiranje	2
Fizika	4

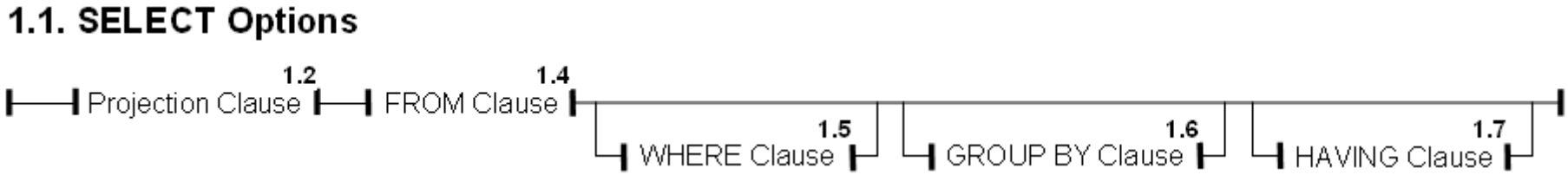
- Kako u rezultatu prikazati samo one grupe koje zadovoljavaju neki uvjet, npr. kako u rezultatu prikazati samo one predmete za koje je prosjek ocjena veći od 2 ?

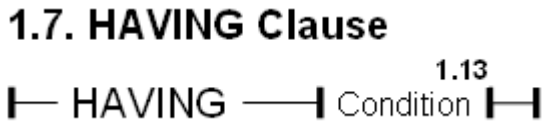
```
SELECT nazPredmet AS naziv
     , AVG(ocjena) AS prosjek
  FROM ispit
  GROUP BY nazPredmet
  HAVING AVG(ocjena) > 2;
```

naziv	prosjek
Matematika	3
Fizika	4

HAVING Clause

- U *Condition* koji se navodi u HAVING dijelu naredbe dopušteno je u izrazima izvan agregatnih funkcija koristiti samo one atribute koji su navedeni u GROUP BY dijelu naredbe






```
SELECT nazPredmet AS naziv
     , AVG(ocjena) AS prosjek
  FROM ispit
  GROUP BY nazPredmet
  HAVING matBr > 104;
```

- U HAVING dijelu SELECT naredbe postavljaju se **uvjeti na grupe** nastale grupiranjem navedenim u GROUP BY dijelu
- za razliku od WHERE dijela u kojem se postavljaju uvjeti na pojedine n-torke

HAVING Clause

- Primjer: ispisati nazive predmeta i njihove prosječne ocjene, ali samo za one predmete u kojima je najveća ikad dobivena ocjena bila manja ili jednaka 4

ispit

matBr	nazPredmet	ocjena
100	Matematika	3
100	Programiranje	2
100	Fizika	5
101	Matematika	2
101	Programiranje	2
101	Fizika	3
102	Matematika	4

```
SELECT nazPredmet AS naziv
     , AVG(ocjena) AS prosjek
  FROM ispit
  GROUP BY nazPredmet
  HAVING MAX(ocjena) <= 4;
```

naziv	prosjek
Matematika	3
Programiranje	2

HAVING Clause

- U rezultatu se pojavljuju one grupe za koje se navedeni uvjet (*Condition*) izračuna kao logička vrijednost *true*. U rezultatu se ne pojavljuju one grupe za koje se navedeni uvjet izračuna kao logička vrijednost *false* ili *unknown*

ispit

matBr	nazPredmet	ocjena
100	Matematika	3
100	Programiranje	2
100	Fizika	NULL
101	Matematika	2
101	Programiranje	2
101	Fizika	NULL
102	Matematika	4

```
SELECT nazPredmet AS naziv
     , AVG(ocjena) AS prosjek
  FROM ispit
  GROUP BY nazPredmet
  HAVING AVG(ocjena) > 2;
```

naziv	prosjek
Matematika	3

GROUP BY, HAVING

Primjer: Za ispite iz studAdmin baze podataka ispisati naziv predmeta, godinu i mjesec ispita (temeljem datuma ispitnog roka) i težinski prosjek ocjena iz položenih ispita ali samo za one godine i mjesece u kojima je težinski prosjek ocjena manji ili jednak 2.50.

ispit

JMBAG	sifPredmet	datumRok	datumIspit	ocjena	sifNastavnik
0555000490	1	13.06.2020	13.06.2020	5	690
…	…	…	…	…	…

predmet

sifPredmet	nazPredmet	ECTSBod	ukBrSatiTjedno
0555000490	Vještine komuniciranja	3.0	2
…	…	…	…

```
SELECT nazPredmet
     , EXTRACT(YEAR FROM datumIspit) godina
     , EXTRACT(MONTH FROM datumIspit) mjesec
     , ROUND(SUM(ocjena*ECTSBod)/SUM(ECTSBod), 2) tezinskiProsjek
FROM ispit
NATURAL JOIN predmet
WHERE ocjena > 1
GROUP BY nazPredmet
       , EXTRACT(YEAR FROM datumIspit)
       , EXTRACT(MONTH FROM datumIspit)
HAVING ROUND(SUM(ocjena*ECTSBod)/SUM(ECTSBod), 2) < 2.50
```

ORDER BY Clause

- Koristi se za sortiranje rezultata upita
- Ispisati podatke o položenim ispitima: poredati ih prema ocjenama, tako da se bliže početku liste nalaze studenti s većim ocjenama. Studente koji imaju međusobno jednake ocjene poredati prema prezimenima, tako da se "manja" prezimena ispisuju prije "većih" prezimena (tj. po abecedi)

poloziliProg

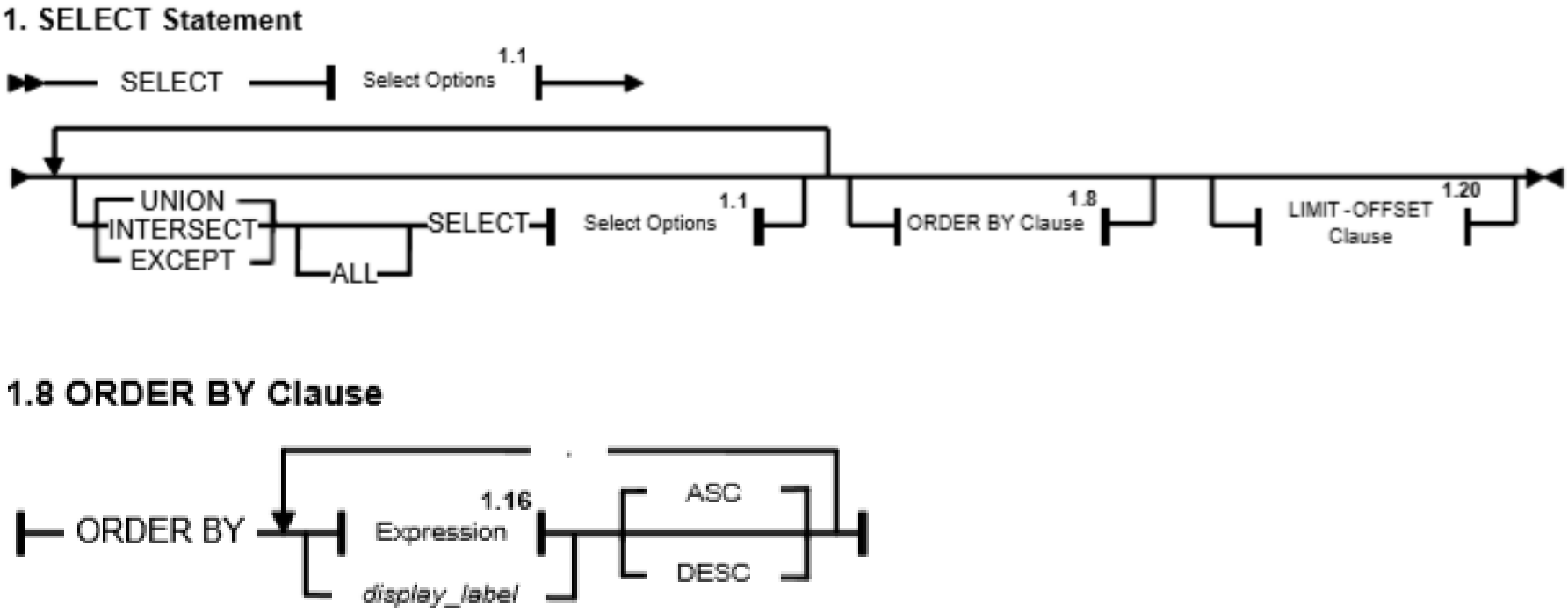
matBr	prez	ocjena
100	Horvat	3
107	Novak	3
102	Horvat	5
101	Kolar	5
103	Kolar	2
104	Horvat	3

```
SELECT *
  FROM poloziliProg
  ORDER BY ocjena DESC
         , prez ASC;
```

matBr	prez	ocjena
102	Horvat	5
101	Kolar	5
104	Horvat	3
100	Horvat	3
107	Novak	3
103	Kolar	2

DESC
silazno (*descending*)
ASC
uzlazno (*ascending*)

ORDER BY Clause



- Ako se smjer sortiranja ne navede, podrazumijeva se uzlazni (ASC) smjer sortiranja

ORDER BY Clause

- U ORDER BY dijelu naredbe mogu se koristiti i izrazi koji nisu navedeni u listi za selekciju
- U jednoj SELECT naredbi može se pojaviti samo jedan ORDER BY dio naredbe
  - ako se u SELECT naredbi koristi UNION, INTERSECT ili EXCEPT (skupovske ili multisetovske operacije), ORDER BY se nalazi iza posljednjeg SELECT dijela naredbe
- SQL standard zahtijeva da se NULL vrijednosti pri sortiranju smatraju ili uvijek manjim ili uvijek većim od svih drugih vrijednosti
  - PostgreSQL NULL vrijednosti pri sortiranju uvijek tretira kao da je veća od svih ostalih vrijednosti

ORDER BY Clause

bodoviMat

mbr	prez	bodLab	bodMI
101	Novak	20	30
103	Horvat	NULL	20
107	Ban	10	80

bodoviProg

mbr	prez	bodLab	bodMI
102	Kolar	12	NULL
104	Novak	30	0

- ispisati podatke o bodovima na lab. vježbama i međuispitu, te ukupnom broju bodova svih studenata, poredati po ukupnom broju bodova: studenti s manjim ukupnim brojem bodova nalaze se bliže početku liste

```
SELECT *, bodLab + bodMI AS ukupno
  FROM bodoviMat
UNION
SELECT *, bodLab + bodMI AS ukupno
  FROM bodoviProg
ORDER BY ukupno;
```

mbr	prez	bodLab	bodMI	ukupno
104	Novak	30	0	30
101	Novak	20	30	50
107	Ban	10	80	90
102	Kolar	12	NULL	NULL
103	Horvat	NULL	20	NULL

"Redoslijed obavljanja" dijelova SELECT naredbe

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. DISTINCT

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. DISTINCT

6. UNION
7. ORDER BY

rezultat

- **Logički promatrano**, tj. konačni rezultat će sigurno odgovarati rezultatu koji bi se dobio kada bi se operacije obavljale navedenim redoslijedom. Fizički promatrano, upit će se možda izvesti drugačijim redoslijedom.