

# Napredni modeli i baze podataka

Predavanja  
listopad 2015.

---

**Napredni SQL**

# Napredni SQL - teme

- 1. Pivotiranje**
- 2. Funkcije za rad s podacima u prozoru**
- 3. Rekurzivni upiti i CTE (Common Table Expression)**

# Napredni SQL - teme

- 1. Pivotiranje**
2. Funkcije za rad s podacima u prozoru
3. Rekurzivni upiti i CTE (Common Table Expression)

# Motivacijski primjer

Odredi broj izlazaka na ispit po godinama i mjesecima u sljedećem obliku:

	siječanj	veljača	ožujak	travanj	svibanj	lipanj	srpanj	kolovoz	rujan	listopad	studen	prosinac
2009	17215	2908	846	728	680	9081	10066	...	...	...	...	...
2010	20400	717	235	196	205	9887	9255	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...

ispit

sifPred	sifStud	datumRok	ocjena
1	1	01.02.2009	1
1	3	01.02.2009	5
1	1	01.03.2009	4
1	2	01.02.2009	5
...	...	...	...

Ovo znamo, ali to nije rješenje zadatka:

```
SELECT EXTRACT(YEAR FROM datumRok) AS godina
      , EXTRACT(MONTH FROM datumRok) AS mjesec
      , COUNT(*) AS brIspita
FROM ispit
GROUP BY godina, mjesec
```

godina	mjesec	brIspita
2009	1	17215
2009	2	2908
...	...	...

Znamo li bolje?

## Pivotiranje – bolje rješenje?

```
SELECT EXTRACT(YEAR FROM datumRok) AS godina
      , (SELECT COUNT(*)
          FROM ispit
          WHERE EXTRACT(MONTH FROM datumRok) = 1) AS siječanj
      , ...
      , (SELECT COUNT(*)
          FROM ispit
          WHERE EXTRACT(MONTH FROM datumRok) = 12) AS prosinac
FROM ispit
GROUP BY godina
ORDER BY godina
```

godina	siječanj	veljača	ožujak	travanj	svibanj	lipanj	srpanj	kolovoz	rujan	listopad	studen	prosina
2009	17215	2908	846	728	680	9081	10066	...	...	...	...	...
2010	20400	717	235	196	205	9887	9255	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...

Jesmo li zadovoljni s rješenjem?

Nespretno i neprimjenjivo u općenitoj situaciji kada je broj stupaca nepoznat.

# Pivotiranje

- Pivotiranje podataka je uobičajena tehnika za predstavljanje podataka u sustavima poslovne inteligencije (BI alati)
- Standardnim SQL-om se podaci (najčešće rezultati primjene agregatnih funkcija) prikazuju u recima, a pivotiranjem se ti isti podatci predočavaju u stupcima

godina	mjesec	brlspita
2009	1	17215
2009	2	2908
...	...	...



	siječanj	veljača	ožujak	travanj	svibanj	lipanj	srpanj	kolovoz	rujan	listopad	studeni	prosinac
2009	17215	2908	846	728	680	9081	10066	...	...	...	...	...
2010	20400	717	235	196	205	9887	9255	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...

- Pivot tablice su sastavni dio aplikacija tipa tablični kalkulator (Excel) :

C	D	E	F	G	H	I	J	K	L	M	N	O	P
sifPred	sifStud	DatumRok	ocjena										
1	1	1.6.2013	1										
1	3	1.6.2013	5										
1	1	1.7.2013	2										
1	2	1.7.2013	2										
1	4	1.7.2013	3										
1	5	1.7.2013	3										
2	1	1.7.2013	4										
2	2	1.7.2013	5										

Count of ocjena	Column Labels					
Row Labels		1	2	3	4	5
1.6.2013		1			1	2
1.7.2013			2	2	1	1
Grand Total		1	2	2	1	2

PivotTable Field List

Choose fields to add to report:

- ☒ DatumRok
- ☒ ocjena

Drag fields between areas below:

Report Filter

Column Labels

- ocjena

Row Labels

- DatumRok

Values

- Count of ocjena

## Pivotiranje u SQL-u

- Nije definirano SQL standardom
- Podržano u
  - Oracle 11g
  - SQL Server 2005+
    - PIVOT
  - PostgreSQL 8.3 +
    - crosstab funkcija

	Rezultat	
crosstab (text sql)	skup n-torki	Proizvodi "pivot tablicu" koja sadrži imena redaka i N imena stupaca. N (tip i broj vrijednosti) je određen upitom.
crosstab (text srcSql, text categorySql)	skup n-torki	Proizvodi "pivot tablicu". Stupci pivot tablice su specificirani drugim upitom (argumentom).

# PostgreSQL: crosstab funkcija

## crosstab (text SQL)

Rezultat SQL naredbe u obliku:

prvaVrijURedu	kategorija	vrijednost
red1	kat1	vrij11
red1	kat2	vrij12
red1	kat3	vrij13
	...	...
red2	kat1	vrij21
red2	kat2	vrij22
red2	kat3	vrij23
		...

crosstab funkcija pretvara u sljedeći oblik

prvaVrijURedu	kat1	kat2	kat3	...
red1	vrij11	vrij12	vrij13	...
red2	vrij21	vrij22	vrij23	...
...	...	...	...	...

SQL naredba mora vratiti trojke s tim da prva vrijednost predstavlja prvu vrijednost u retku, druga kategoriju stupca a treća je vrijednost ćelije.

*crosstab* funkcija vraća skup n-torki i imena stupaca **moraju biti eksplicitno definirana** u FROM dijelu pozivajuće SQL naredbe

```
SELECT * FROM crosstab('...') AS ct (prvaVrijURedu text, kat1 text, kat2 ext,...);
```

Rezultat SQL naredbe (koja je argument *crosstab* funkcije) **mora** biti sortiran prema *prvaVrijURedu*, a zatim prema *kategoriji*.



# PostgreSQL: crosstab funkcija - primjer

Odredi broj izlazaka na ispit po godinama i mjesecima u sljedećem obliku:

godina	siječanj	veljača	ožujak	travanj	svibanj	lipanj	srpanj					
2009	17215	2908	846	728	680	9081	10066	...	...	...	...	...
2010	20400	717	235	196	205	9887	9255	...	...	...	...	...
...	...	...		...	...	...	...	...	...	...		

```
SELECT * FROM crosstab(text SQLStatement)
      AS pivotTable (godina INT, siječanj INT, veljača INT,...)
```

Rezultat izvođenja SQLStatement mora biti skup trojki s tim da

1. vrijednost predstavlja godinu
2. vrijednost redni broj mjeseca u godini
3. ukupan broj ispita u godini i mjesecu

godina	mjesec	brIsпита
2009	1	17215
2009	2	2908
2009	3	846
...	...	...

```
SELECT EXTRACT(YEAR FROM datumRok) AS godina
      , EXTRACT(MONTH FROM datumRok) AS mjesec
      , COUNT(*)                      AS brIsпита
FROM ispit
GROUP BY EXTRACT(YEAR FROM datumRok), EXTRACT(MONTH FROM datumRok)
ORDER BY EXTRACT(YEAR FROM datumRok), EXTRACT(MONTH FROM datumRok)
```

Rezultat gornje SQL naredbe će crosstab funkcija transformirati u pivot tablicu.

# PostgreSQL: crosstab funkcija - primjer

Odredi broj izlazaka na ispit po godinama i mjesecima u sljedećem obliku:

	siječanj	veljača	ožujak	travanj	svibanj	lipanj	srpanj					
2009	17215	2908	846	728	680	9081	10066	...	...	...	...	...
2010	20400	717	235	196	205	9887	9255	...	...	...	...	...
...	...	...		...	...	...	...	...	...	...		

```
SET DateStyle = 'German, DMY';
CREATE EXTENSION tablefunc;
SELECT *
  FROM crosstab ('SELECT EXTRACT(YEAR FROM datumRok) AS godina
                  , EXTRACT(MONTH FROM datumRok) AS mjesec
                  , COUNT(*) AS brIsp
                  FROM ispit
                 GROUP BY godina, mjesec
                 ORDER BY godina, mjesec')
 AS pivotTable (godina INT
                , siječanj INT, veljača INT, ožujak INT, travanj INT, svibanj INT, lipanj INT
                , srpanj INT, kolovoz INT, rujan INT, listopad INT, studeni INT, prosinac INT)
 ORDER BY godina
```

mora biti int

za istu godinu zapisi moraju biti sortirani

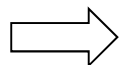
Preporuka: osigurati odgovarajući tip podatka u skladu s definicijom rezultata SQL naredbe

```
SELECT *
  FROM crosstab ('SELECT CAST(EXTRACT(YEAR FROM datumRok) AS int) AS godina
                  , EXTRACT(MONTH FROM datumRok)::int AS mjesec
                  , CAST(COUNT(*) AS int) AS brIsp
                  FROM ispit
                 ...
```

# PostgreSQL: crosstab funkcija – primjer (poboljšano rješenje)

Što ako za neku godinu nije bilo ispita u svakom od 12 mjeseci?

godina	mjesec	brlspita
2006	12	567
2007	2	17215
2007	5	2908
2008	1	9055
...	...	...



	siječanj	veljača	ožujak	travanj	svibanj							
2006	...	...	...	...	...	...	...	...	...	...	...	...
2007	17215	2908										
...	...	...	...	...	...	...	...	...	...	...	...	...

```
CREATE TEMP TABLE mjesec
(rbrMjesec int);
INSERT INTO mjesec VALUES ( 1);
...
INSERT INTO mjesec VALUES ( 12);
```

crosstab (text srcSql, text categorySql)

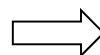
```
SELECT *
FROM crosstab ('SELECT CAST(EXTRACT(YEAR FROM datumRok) AS int) AS godina
                , CAST(EXTRACT(MONTH FROM datumRok) AS int) AS mjesec
                , CAST(COUNT(*) AS int) AS brIsp
                FROM ispit
                GROUP BY godina, mjesec
                ORDER BY godina, mjesec'
, 'SELECT rbrMjesec FROM mjesec ORDER BY rbrMjesec')
AS pivotTable (godina INT
                , siječanj INT, veljača INT, ožujak INT, travanj INT, svibanj INT, lipanj INT
                , srpanj INT, kolovoz INT, rujan INT, listopad INT, studeni INT, prosinac INT)
ORDER BY godina
```

	siječanj	veljača	ožujak	travanj	svibanj							
2006	...	...	...	...	...	...	...	...	...	...	...	...
2007		17215			2908							

# PostgreSQL: Pivotiranje – unaprijed nepoznat broj stupaca?

kupacProizvod

kupac	proizvod	količina
Ana	Mlijeko	45
Pero	Sol	45
Mia	Šećer	35
Ivo	Mlijeko	37
Ana	Brašno	65
Pero	Mlijeko	75
Mia	Mlijeko	35
Ivo	Sol	45
...	...	...



kupac	brašno	mlijeko	sol	šećer	...
Ana	65	45			
Ivo			45		
Mia		35		35	
Pero		75	45		

```
SELECT *  
FROM crosstab ('SELECT kupac, proizvod, kolicina  
                FROM kupacProizvod  
                ORDER BY kupac, proizvod'  
              , 'SELECT DISTINCT proizvod  
                FROM kupacProizvod  
                ORDER BY proizvod')  
AS pivotTable (kupac char(15), Brašno INT, Mlijeko INT, Sol INT, Šećer INT)  
ORDER BY kupac
```

generirati dinamički  
(Dynamic SQL )

- dinamičko kreiranje i izvođenje SQL naredbu moguće samo u PL/pgSQL - SQL Procedural Language
- Dynamic SQL nije predviđen kao tema u okviru predmeta

# PostgreSQL: Pivotiranje - primjer s ispita iz rujna 2014

igrac

siflgrac	ime	prezime
987	Marin	Čilić
875	Novak	Đoković
452	Roger	Federer
564	Kei	Nishikori
...	...	...

turnir

sifTurnir	naziv
54	Australian Open
23	Roland Garros
17	Wimbeldon
98	US Open

turnirIgrac

godina	sifTurnir	siflgrac	bodovi
2013	54	875	2000
2014	98	987	2000
2014	98	564	1500
2014	17	875	2000
2014	17	452	1300
...	...	...	...

Korištenjem mogućnosti PostgreSQL sustava baza podataka i ugrađenih funkcija za pivotiranje u tom sustavu, napišite SQL upit pomoću kojeg će se ispisati ukupan broj bodova koje su igrači osvojili po godinama od 2010 do 2014 u sljedećem obliku:

prezimeIgrac	g2010	g2011	g2012	g2013	g2014
Čilić	12290	18			
Đoković	8670	20			
...	...	...			

```
CREATE TEMP TABLE godina
(godina int);
INSERT INTO godina VALUES ( 2010);
INSERT INTO godina VALUES ( 2011);
INSERT INTO godina VALUES ( 2012);
INSERT INTO godina VALUES ( 2013);
INSERT INTO godina VALUES ( 2014);
```

```
SELECT *
FROM crosstab ('SELECT CAST(igrac.prezime as char(50)) AS prezimeIgrac
, CAST(godina AS int) as godina
, CAST(SUM(osvojioBodova) AS int) AS ukBodova
FROM igrac JOIN turnirIgrac
ON igrac.sifIgrac = turnirIgrac.sifIgrac
GROUP BY prezimeIgrac, godina
ORDER BY prezimeIgrac, godina'
, 'SELECT CAST(godina as INT) FROM godina ORDER BY godina')
AS pivotTable (prezimeIgrac Char(50)
, g2010 INT, g2011 INT, g2012 INT, g2013 INT, g2014 INT)
ORDER BY prezimeIgrac
```

# PostgreSQL: Pivotiranje - primjer s ispita iz rujna 2014

igrac

siflgrac	ime	prezime
----------	-----	---------

turnir

sifTurnir	naziv
-----------	-------

turnirIgrac

godina	sifTurnir	siflgrac	bodovi
--------	-----------	----------	--------

Korištenjem mogućnosti PostgreSQL sustava baza podataka i ugrađenih funkcija za pivotiranje u tom sustavu, napišite SQL upit pomoću kojeg će se ispisati ukupan broj bodova koje su igrači osvojili po godinama od 2010 do 2014 u sljedećem obliku:

prezimeIgrac	g2010	g2011	g2012	g2013	g2014
Čilić	12290	18			
Đoković	8670	20			
...	...	...			

Ovo nije ispravno rješenje:

```
SELECT *
FROM crosstab ('SELECT CAST(igrac.prezime as char(50)) AS prezimeIgrac
                , CAST(godina AS int) as godina
                , CAST(SUM(osvojioBodova) AS int) AS ukBodova
FROM igrac JOIN turnirIgrac
      ON igrac.sifIgrac = turnirIgrac.sifIgrac
GROUP BY prezimeIgrac, godina
ORDER BY prezimeIgrac, godina')
AS pivotTable (prezimeIgrac Char(50)
, g2010 INT, g2011 INT, g2012 INT, g2013 INT, g2014 INT)
ORDER BY prezimeIgrac
```

# Napredni SQL - teme

1. Pivotiranje
- 2. Funkcije za rad s podacima u prozoru**
3. Rekurzivni upiti i CTE (Common Table Expression)

# Motivacijski primjer 1

Za predmet i ispitni rok odrediti kumulativan broj ispita i kumulativan prosjek ocjena. Kumulativan broj ispita, odnosno prosjek ocjena, uključuje sve ispite s tog i prethodnih ispitnih rokova iz istog predmeta.

ispit

sifPred	sifStud	datumRok	ocjena
1	1	01.06.2013	1
1	3	01.06.2013	5
1	1	01.07.2013	4
1	2	01.07.2013	5
1	4	01.07.2013	3
1	5	01.07.2013	2
2	1	01.07.2013	3
2	2	01.07.2013	2

Ovo znamo, ali to nije rješenje zadatka:

```
SELECT sifPred
      , datumRok
      , COUNT(*)      brIspit
      , AVG(ocjena)   prOcj
FROM ispit
GROUP BY sifPred, datumRok
```

sifPred	datumRok	kumBrispit	kumPrOcj
1	01.06.2013	2	3.00
1	01.06.2013	6	3.33
2	01.07.2013	2	2.50

Format 'dd.mm.yyyy' se postiže sa:  
SET dateStyle ='German, DMY';

sifPred	datumRok	brIspit	prOcj
1	01.06.2013	2	3.00
1	01.06.2013	4	3.50
2	01.07.2013	2	2.50



## Motivacijski primjer 2

Odrediti ukupan broj bodova i rang studenta u okviru predmeta.

studProvjera

sifPred	sifStud	kratProvjera	bod
1	1	1DZ	5.00
1	2	1DZ	10.00
1	3	1DZ	5.00
1	1	MI	10.00
1	2	MI	5.00
1	3	MI	20.00
...	...	...	...
2	1	MI	20.00
2	2	MI	25.00
2	3	MI	30.00

Ovo znamo, ali to nije rješenje zadatka:

```
SELECT sifPred
      , sifStud
      , SUM(bod)    ukBod
FROM studProvjera
GROUP BY sifPred, sifStud
--ORDER BY ukBod DESC, sifPred
```

sifPred	sifStud	ukBod	rang
1	3	25.00	1
1	1	15.00	2
1	2	15.00	2
2	3	30.00	1
2	2	25.00	2
2	1	20.00	3

sifPred	sifStud	ukBod
1	3	25.00
1	1	15.00
1	2	15.00
2	3	30.00
2	2	25.00
2	1	20.00

## Motivacijski primjer 3

Odrediti ukupan broj bodova i rang studenta za svaki predmet. Dodatno, odrediti prosjek ukupnog broja bodova koje studenti osvoje na predmetu.

studProvjera

sifPred	sifStud	kratProvjera	bod
1	1	1DZ	5.00
1	2	1DZ	10.00
1	3	1DZ	5.00
1	1	MI	10.00
1	2	MI	5.00
1	3	MI	20.00
...	...	...	...
2	1	MI	20.00
2	2	MI	25.00
2	3	MI	30.00

sifPred	sifStud	ukBod	rang	prosUkBod
1	3	25.00	1	18.333
1	1	15.00	2	18.333
1	2	15.00	2	18.333
2	3	30.00	1	25.000
2	2	25.00	2	25.000
2	1	20.00	3	25.000

Ukupan broj bodova studenta na predmetu:

```
SELECT sifPred
      , SUM(bod) ukBod
FROM studProvjera
GROUP BY sifPred, sifStud
```

Prosjek ukupnog broja bodova za studenta na predmetu?

**sintaksa ne dozvoljava**

```
SELECT sifPred
      , AVG(SUM(bod))
      prosUkBod
FROM studProvjera
GROUP BY ????
```

## Što nam treba?

- Mogućnost podjele n-torki relacije na prozore (particije/okvire)
  - prozor n-torke se određuje dinamički obzirom na njenu okolinu
- Mogućnost računanja agregatnih vrijednosti na razini prozora (particije/okvira)
  - u istom SQL upitu različite agregatne funkcije izračunati temeljem različito definiranih prozora
  - izbjeći „gubitak” n-torke koje se nužno dogodi pri GROUP BY
- Dodatne funkcije čiji rezultat se evaluiira temeljem n-torki sadržanih u prozoru (particiji/okviru) (npr. rang)

## Prozori i funkcije za rad s prozorima

- Definirane standardom ISO SQL:2003, detaljnije razrađene standardom ISO SQL:2008
- Podržane u Oracle, SQL Server, Sybase, DB2, **PostgreSQL**, Firebird
- Primjenjuju se najčešće u OLAP ali i u OLTP sustavima za upite koji uključuju
  - rangiranje
  - ugniježdene, kumulativne i "pomične" agregacije
  - ...
- Sintaksa i primjeri koji slijede: PostgreSQL

## SQL agregatna funkcija: GROUP BY ↔ nad prozorom

SQL agregatna funkcija:

```
SELECT sifPred
      , datumRok
      , COUNT(*)    brIspit
      , AVG(ocjena) prOcj
FROM ispit
GROUP BY sifPred, datumRok
```

ispit

sifPred	sifStud	datumRok	ocjena
1	1	01.06.2013	1
1	3	01.06.2013	5
1	1	01.07.2013	4
1	2	01.07.2013	5
1	4	01.07.2013	3
1	5	01.07.2013	2
2	1	01.07.2013	3
2	2	01.07.2013	2

sifPred	datumRok	brIspit	prOcj
1	01.06.2013	2	3.00
1	01.07.2013	4	3.50
2	01.07.2013	2	2.50

Funkcija nad podacima u prozoru:

```
SELECT sifPred
      , sifStud
      , ocjena
      , COUNT(*) OVER
      (PARTITION BY
        sifPred, datumRok) brIsp
      , AVG(ocjena) OVER
      (PARTITION BY
        sifPred, datumRok) prOcj
FROM ispit
```

sifPred	sifStud	datumRok	ocjena		sifPred	sifStud	ocjena	brIsp	prOcj
1	1	01.06.2013	1		1	1	1	2	3.00
1	3	01.06.2013	5		1	3	5	2	3.00
1	1	01.07.2013	4		1	1	4	4	3.50
1	2	01.07.2013	5		1	2	5	4	3.50
1	4	01.07.2013	3		1	4	3	4	3.50
1	5	01.07.2013	2		1	5	2	4	3.50
2	1	01.07.2013	3		2	1	3	2	2.50
2	2	01.07.2013	2		2	2	2	2	2.50

# GROUP BY + agregatne funkcije ↔ Funkcije za rad s prozorima

GROUP BY $atr_1, \dots, atr_n$	Funkcije za rad s prozorima
Jedna n-torka na izlazu za svaku grupu	Jedna n-torka na izlazu za svaku u prozor ulaznu n-torku
Vrijednosti agregatnih funkcija se računaju za cijelu grupu	Vrijednosti agregatnih funkcija se računaju za prozore (particije/okvire)
Samo jedan način grupiranja u jednoj SELECT naredbi	Agregati u istoj SELECT naredbi mogu biti izračunati temeljem n-torki sadržanih u različitim prozorima (particijama/okvirima)

## Prozor (Window)

Tranzijentni skup n-torki pomoću kojeg se definiraju:

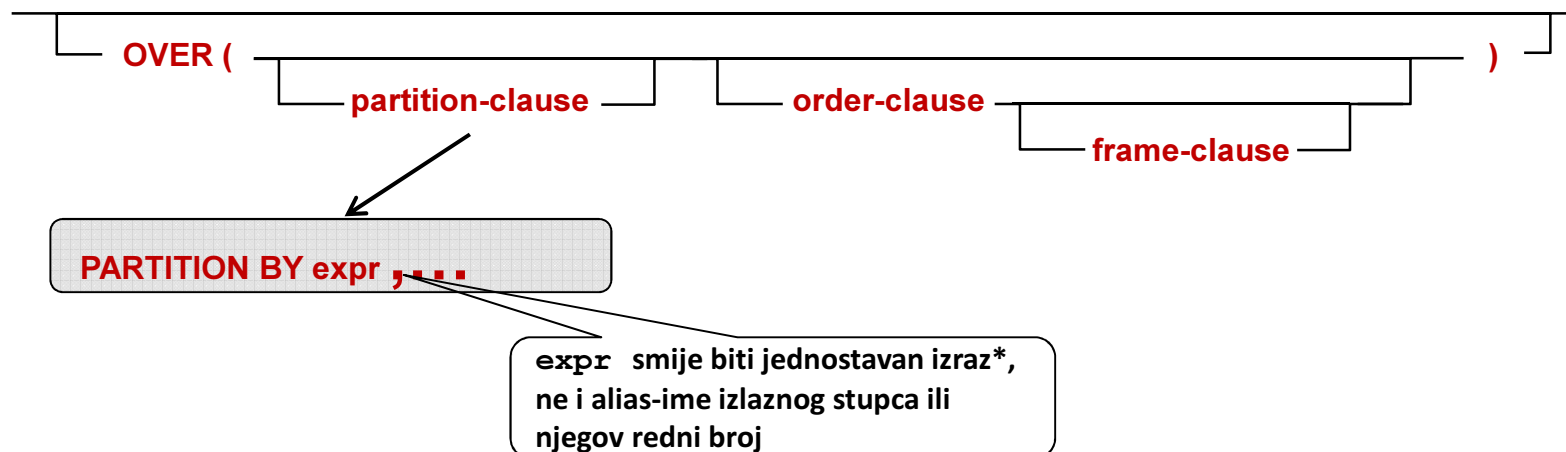
- Particija (partition)
  - Definira se pomoću PARTITION BY dijela
  - Ne može biti pomična
  - Može sadržavati okvir(e)
- Okvir (frame)
  - Definira se
    - pomoću frame i ORDER BY dijelova
    - uvijek relativno u odnosu na tekuću n-torku
  - Pomiče se unutar particije
  - Ne može zahvatiti više particija
- **U istoj SELECT naredbi moguće je neke funkcije izračunavati temeljem n-torki u particiji a neke temeljem n-torki u okviru.**

# PostgreSQL: Particija

- Definira se pomoću PARTITION BY dijela u OVER ()
- Omogućuje podjelu n-torki relacije, slično kao GROUP BY
- Svaka n-torka pripada jednoj particiji
- Bez PARTITION BY cijela relacija je jedna particija

```
SELECT sifPred
      , sifStud
      , ocjena
      , AVG(ocjena) OVER (PARTITION BY sifPred, datumRok)
FROM ispit
```

Function  
(args)



\*jednostavni izraz uključuje agregatnu funkciju primijenjenu nad nekim izrazom – npr. AVG(ocjena) ili SUM(bodovi)



# PostgreSQL: Okvir (frame)

- Definira se pomoću frame i order dijelova u OVER ()
- Svaka n-torka pripada okviru, okvir pripada particiji
- Bez frame i ORDER dijela, cijela particija predstavlja jedan okvir
- Okvir se može pomicati kako se pomičemo po djelatnom skupu

Function  
(args)



order-clause

**ORDER BY** *expr* [ASC|DESC] [NULS FIRST|LAST] ,...

*expr* smije biti jednostavan izraz,  
ne i alias-ime izlaznog stupca ili  
njegov redni broj

frame-clause

**ROWS** **RANGE** **BETWEEN** **endpoint-spec** **AND** **endpoint-spec**

endpoint-spec

**UNBOUNDED PRECEDING**

**unsigned-value** **PRECEDING**

**CURRENT ROW**

**unsigned-value** **FOLLOWING**

**UNBOUNDED FOLLOWING**

## Okvir (frame)

Dvije vrste okvira:

- ROWS okviri
  - n-torka pripada okviru tekuće n-torke ako je u specificiranom rasponu – navodi se broj n-torki koje prethode ili slijede tekuću
  - poredak ne mora biti definiran (bez ORDER BY)
- RANGE okviri
  - n-torka pripada okviru tekuće n-torke ako je vrijednost odgovarajućih atributa (navedenih u ORDER BY) u specificiranom rasponu vrijednosti
  - Važan je poredak n-torki u particiji (ORDER BY) jer on definira n-torke koje prethode/slijede tekuću.

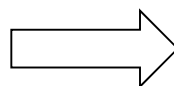
# PostgreSQL: Definiranje okvira

- [ RANGE | ROWS ] BETWEEN FrameStart AND FrameEnd
  - FrameStart i FrameEnd mogu biti :
    - UNBOUNDED PRECEDING
    - n PRECEDING (za sada samo za ROWS)\*
    - CURRENT ROW
    - n FOLLOWING (za sada samo za ROWS)\*
    - UNBOUNDED FOLLOWING
  - skraćeno:  
[ RANGE | ROWS ] FrameStart  
znači  
[ RANGE | ROWS ] BETWEEN FrameStart  
AND CURRENT ROW
- U PostgreSQL-u se razlika između ROWS i RANGE okvira ne može uočiti zbog ograničenja (\*)

# ORACLE: Definiranje okvira pomoću RANGE - primjer

dohodakDjel

sifDjelatnik	datumDohodak	iznos
1	01.06.2013	5000.00
1	15.06.2013	1000.00
1	01.07.2013	5000.00
1	01.08.2013	6000.00
2	01.07.2013	7000.00
2	01.09.2013	5000.00



sifDjelatnik	datumDohodak	mjesec	iznos	iznos3M
1	01.06.2013	6	5000.00	11000.00
1	15.06.2013	6	1000.00	11000.00
1	01.07.2013	7	5000.00	17000.00
1	01.08.2013	8	6000.00	11000.00
2	01.07.2013	7	7000.00	7000.00
2	01.09.2013	9	5000.00	5000.00

```
SELECT sifDjelatnik, datumDohodak,  
       EXTRACT (MONTH FROM datumDohodak) mjesec, iznos  
       SUM(iznos) OVER (PARTITION BY sifDjelatnik  
                        ORDER BY EXTRACT (MONTH FROM datumDohodak)  
                        RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING) iznos3M  
FROM dohodakDjel  
ORDER BY sifDjelatnik, EXTRACT (MONTH FROM datumDohodak)
```

Mjesec je cjelobrojna vrijednost pa dodavanje i oduzimanje 1 u odnosu na mjesec tekuće n-torke stvara okvir od 3 mjeseca. Osim u slučaju kada djelatnik nije imao dohodak u svakom mjesecu.

## PostgreSQL: Primjeri okvira

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW	sve n-torke od početka particije do tekuće n-torke
ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING	prethodna n-torka
ROWS BETWEEN 1 FOLLOWING AND 1 FOLLOWING	sljedeća n-torka
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING	Prethodna, trenutna i sljedeća n- torka

# PostgreSQL: Okvir (frame) – primjeri 1

ispit

sifPred	sifStud	datumRok	ocjena
1	1	01.06.2013	1
1	3	01.06.2013	5
1	1	01.07.2013	4
1	2	01.07.2013	5
1	4	01.07.2013	3
1	5	01.07.2013	2
2	1	01.07.2013	3
2	2	01.07.2013	2

```
SELECT AVG(ocjena) OVER ()  
FROM ispit;
```

Cijela relacija je jedna particija.  
Okvir je cijela particija (relacija)

avg
3.125
3.125
3.125
3.125
3.125
3.125
3.125
3.125

```
SELECT sifPred, sifStud, datumRok  
      , ocjena  
      , AVG(ocjena) OVER(PARTITION BY sifPred)  
FROM ispit  
ORDER BY sifPred;
```

Particiji pripadaju n-torke s jednakom sifPred.  
Okvir je cijela particija.

sifPred	sifStud	datumRok	ocjena	avg
1	1	01.06.2013	1	3.333
1	3	01.06.2013	5	3.333
1	1	01.07.2013	4	3.333
1	2	01.07.2013	5	3.333
1	4	01.07.2013	3	3.333
1	5	01.07.2013	2	3.333
2	1	01.07.2013	3	2.500
2	2	01.07.2013	2	2.500

## PostgreSQL: Okvir (frame) – primjeri 2

```
SELECT *,
  AVG(ocjena) OVER (PARTITION BY sifPred
                    ORDER BY datumRok, ocjena)
FROM ispit
ORDER BY sifPred, datumRok, ocjena;
```

```
SELECT *,
  AVG(ocjena) OVER (PARTITION BY sifPred
                    ORDER BY ocjena, datumRok)
FROM ispit
ORDER BY sifPred, ocjena, datumRok;
```

Jednak rezultat dobije se i sljedećom sintaksom:

```
OVER (PARTITION BY sifPred ORDER BY datumRok, ocjena
      ROWS UNBOUNDED PRECEDING)
--ili ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
--ili RANGE UNBOUNDED PRECEDING)
--ili RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
```

Particiji pripadaju n-torke s jednakom sifPred.

Okviru pripadaju n-torke iz particije od početka do trenutne (CURRENT ROW) - važan je ORDER BY. Bez ORDER BY sve n-torke particije pripadale bi istom okviru.

sifPred	sifStud	datumRok	ocjena	avg
1	1	01.06.2013	1	1.000
1	3	01.06.2013	5	3.000
1	5	01.07.2013	2	2.666
1	4	01.07.2013	3	2.750
1	1	01.07.2013	4	3.000
1	2	01.07.2013	5	3.333
2	2	01.07.2013	2	2.000
2	1	01.07.2013	3	2.500

sifPred	sifStud	datumRok	ocjena	avg
1	1	01.06.2013	1	1.000
1	5	01.07.2013	2	1.500
1	4	01.07.2013	3	2.000
1	1	01.07.2013	4	2.500
1	3	01.06.2013	5	3.000
1	2	01.07.2013	5	3.333
2	2	01.07.2013	2	2.000
2	1	01.07.2013	3	2.500

## PostgreSQL: Definiranje prozora – dodatna mogućnost

- Prozor je moguće definirati i imenovati na jednom mjestu
- Može se koristiti isključivo u SELECT i ORDER BY dijelovima
  - ne u npr. WHERE, GROUP BY i HAVING dijelovima

```
SELECT targetList,... wFunc(args) OVER w
```

```
FROM tableList,...
```

```
WHERE qualList,...
```

```
GROUP BY groupKeyList,...
```

```
HAVING groupQualList,...
```

```
WINDOW w AS — ( —  
                  partition-clause order-clause  
                  frame-clause ) —
```

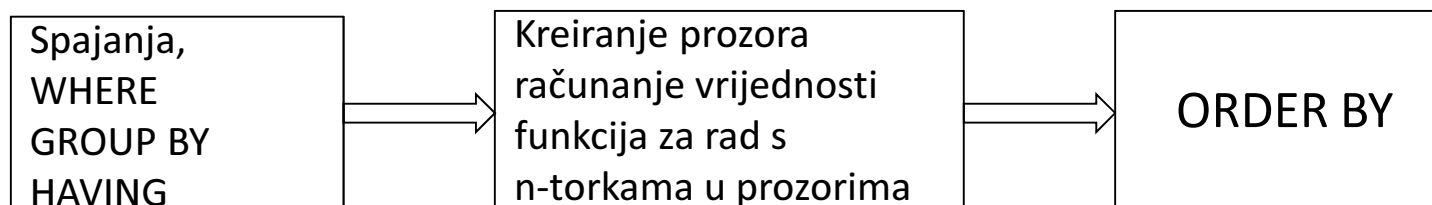
```
ORDER BY targetList,...
```

Zašto?

```
SELECT sifPred, sifStud, datumRok  
      , ocjena  
      , AVG(ocjena) OVER w  
FROM ispit  
WINDOW w AS (PARTITION BY sifPred  
              ORDER BY datumRok, ocjena  
              RANGE UNBOUNDED PRECEDING)  
ORDER BY sifPred, datumRok, AVG(ocjena) OVER w;
```



## Redoslijed aktivnosti pri obavljanju upita



Funkcije za rad s prozorima mogu koristiti izračunate vrijednosti agregatnih funkcija (COUNT, AVG, SUM,...) nakon procesiranja FROM, WHERE, GROUP BY i HAVING dijelova SELECT naredbe

Posljedica (i to odlična): mogućnost obavljanja ugniježđenih agregacija:

- **AVG (SUM (bod) )**
- **MIN (SUM (bod) )**
- ...

Sada je jasno zbog čega se izrazi s „prozorima” mogu koristiti isključivo u SELECT i ORDER BY dijelovima upita.

# PostgreSQL: Ugniježdene agregacije – primjer 1

Za svakog zaposlenika ispisati ukupan prihod u prvih 6 mjeseci 2013. godine. Ispisati i prosjek ukupnog prihoda svih zaposlenika tvrtke te prosjek ukupnog prihoda zaposlenih u istoj organizacijskoj jedinici u prvih 6 mjeseci 2013. godine.

djelatnik

sifDjel	ime	prezime	sifOrgJed
1	Ana	Hlad	1
2	Petar	Milić	1
3	Mia	Horvat	2
4	Ivan	Kralj	2

orgJed

sifOrgJed	naziv
1	Kadrovska
2	Tajništvo

djellsplata

sifDjel	datumIsplata	iznos
1	01.01.2013	4500.00
1	01.02.2013	4530.00
2	01.01.2013	3540.00
2	01.01.2013	3700.00
3	01.01.2013	6500.00
3	01.02.2013	7550.00
4	01.01.2013	3500.00
4	01.02.2013	4570.00

```
SELECT ime, prezime, naziv
      , SUM (iznos) ukIznos
      , AVG (SUM(iznos)) OVER () prosT
      , AVG (SUM(iznos)) OVER
        (PARTITION BY djelatnik.sifOrgJed)
        prosOJ
FROM djelatnik, orgjed, djelIsplata
WHERE djelatnik.sifOrgJed = orgjed.sifOrgJed
      AND djelatnik.sifDjel= djelIsplata.sifDjel
      AND datumIsplata BETWEEN '01.01.2013'
                                AND '30.06.2013'
GROUP BY djelatnik.sifDjel, ime, prezime
        , djelatnik.sifOrgJed, naziv
```

ime	prezime	naziv	ukIznos	prosT	prosOJ
Ana	Hlad	Kadrovska	9030.00	9597.50	8135.00
Petar	Milić	Kadrovska	7240.00	9597.50	8135.00
Mia	Horvat	Tajništvo	14050.00	9597.50	11060.00
Ivan	Kralj	Tajništvo	8070.00	9597.50	11060.00

## PostgreSQL: Ugniježdene agregacije – primjer 2

Za predmet i ispitni rok odrediti kumulativan broj ispita koji uključuje sve ispite s tog i neposredno prethodnog ispitnog roka iz istog predmeta.

ispit

sifPred	sifStud	datumRok	ocjena
1	1	01.06.2013	1
1	3	01.06.2013	5
1	1	01.07.2013	4
1	2	01.07.2013	5
1	4	01.07.2013	3
1	5	01.07.2013	2
1	6	01.09.2013	3
1	7	01.09.2013	4
2	1	01.07.2013	3
2	2	01.07.2013	2

sifPred	datumRok	kumBrispit
1	01.06.2013	2
1	01.07.2013	6
1	01.09.2013	6
2	01.07.2013	2

```
SELECT sifPred, datumRok
      , SUM(COUNT(*)) OVER (PARTITION BY sifPred ORDER BY datumRok
                           ROWS 1 PRECEDING ) kumBrIspit
FROM ispit
GROUP BY sifPred, datumRok
ORDER BY sifPred, datumRok
```

## Ugrađene funkcije za rad s prozorima (Window functions)

Prema SQL standardu funkcija za rad s prozorima može biti:

1. funkcija za rangiranje
2. funkcija za distribuciju
3. funkcija za određivanje n-torke temeljem pozicije (row number)
4. agregatna funkcija na razini prozora

## PostgreSQL: Ugrađene funkcije za rad s prozorima

- Funkcije za rangiranje:
  - `row_number()`
  - `rank()`
  - `dense_rank()`
  - `percent_rank()`
- Funkcije za distribuciju:
  - `cume_dist()`
  - `ntile()`
- Funkcija za određivanje n-torke temeljem pozicije
  - `lag()`
  - `lead()`
  - `first_value()`
  - `last_value()`
  - `nth_value()`

## PostgreSQL: Funkcije za rangiranje

<code>row_number ()</code>	Redni broj n-torke u particiji, počinje od 1 za svaku particiju
<code>rank ()</code>	Rang n-torke u particiji, za jednake vrijednosti vraća jednak rang
<code>dense_rank ()</code>	Jednako kao <i>rank ()</i> ali bez "rupa" u slučaju jednako rangiranih n-torki
<code>percent_rank()</code>	relativni rang n-torke: $(rang-1)/(ukupno\ n-torki - 1)$

# PostgreSQL: Funkcije za rangiranje - primjeri

ispit

sifPred	sifStud	datumRok	ocjena
1	1	01.06.2013	1
1	3	01.06.2013	5
1	1	01.07.2013	4
1	2	01.07.2013	5
1	4	01.07.2013	3
1	5	01.07.2013	2
2	1	01.07.2013	3
2	2	01.07.2013	2

```
SELECT sifPred, sifStud, datumRok, ocjena
, rank() OVER (ORDER BY ocjena DESC) as rang
, dense_rank() OVER (ORDER BY ocjena DESC) as dRang
, row_number() OVER (ORDER BY ocjena DESC) as rowNum
FROM ispit
ORDER BY ocjena DESC
```

sifPred	sifStud	datumRok	ocjena	rang	dRang	rowNum
1	3	01.06.2013	5	1	1	1
1	2	01.07.2013	5	1	1	2
1	1	01.07.2013	4	3	2	3
2	1	01.07.2013	3	4	3	4
1	4	01.07.2013	3	4	3	5
2	2	01.07.2013	2	6	4	6
1	5	01.07.2013	2	6	4	7
1	1	01.06.2013	1	8	5	8

```
SELECT sifPred, sifStud, datumRok, ocjena
, rank() OVER (PARTITION BY datumRok
ORDER BY ocjena DESC) as rang
, dense_rank() OVER (PARTITION BY datumRok
ORDER BY ocjena DESC) as dRang
, row_number() OVER (PARTITION BY datumRok
ORDER BY ocjena DESC) as rowNum
FROM ispit
ORDER BY datumRok, ocjena DESC
```

sifPred	sifStud	datumRok	ocjena	rang	dRang	rowNum
1	3	01.06.2013	5	1	1	1
1	1	01.06.2013	1	2	2	2
1	2	01.07.2013	5	1	1	1
1	1	01.07.2013	4	2	2	2
1	4	01.07.2013	3	3	3	3
2	1	01.07.2013	3	3	3	4
2	2	01.07.2013	2	5	4	5
1	5	01.07.2013	2	5	4	6

## PostgreSQL: Rangiranje temeljem agregacije - primjer

Obzirom na ukupno osvojene bodove odrediti

- rang studenta na predmetu i
- ukupan rang bez obzira na predmet.

Zapise poredati prema šifri predmeta i rangu unutar predmeta.

studProvjera

sifPred	sifStud	kratProvjera	bod
1	1	1DZ	5.00
1	2	1DZ	10.00
1	3	1DZ	5.00
1	1	MI	10.00
1	2	MI	5.00
1	3	MI	20.00
2	1	MI	20.00
2	2	MI	25.00
2	3	MI	30.00

```
SELECT sifPred, sifStud, SUM(Bod) ukBod
      , rank() OVER (PARTITION BY sifPred
                     ORDER BY SUM(bod) DESC) rangPred
      , rank() OVER (ORDER BY SUM(bod) DESC) rangGlob
FROM studProvjera
GROUP BY sifPred, sifStud
ORDER BY sifPred, SUM(bod) DESC
--ORDER BY sifPred, rank() OVER (PARTITION BY sifPred
--                                ORDER BY SUM(bod) DESC)
--ORDER BY sifPred, rangPred
```

sifPred	sifStud	ukBod	rangPred	rangGlob
1	3	25.00	1	2
1	1	15.00	2	5
1	2	15.00	2	5
2	3	30.00	1	1
2	2	25.00	2	2
2	1	20.00	3	4



# PostgreSQL: Primjer s ispita iz rujna 2014

igrac

siflgrac	ime	prezime
987	Marin	Čilić
875	Novak	Đoković
452	Roger	Federer
564	Kei	Nishikori
...	...	...

turnir

sifTurnir	naziv
54	Australian Open
23	Roland Garros
17	Wimbeldon
98	US Open

turnirIgrac

godina	sifTurnir	siflgrac	bodovi
2013	54	875	2000
2014	98	987	2000
2014	98	564	1500
2014	17	875	2000
2014	17	452	1300
...	...	...	...

Potrebno je ispisati rang listu igrača (kao u tablici desno) pri čemu se rang odredi temeljem ukupnog osvojenog broja bodova na Grand Slam turnirima. Dodatno, ispisati i ukupan broj Grand Slam turnira na kojima je igrač sudjelovao u cijeloj karijeri (ukTurnira).

rang	igrac	ukBodova	ukTurnira
1	Đoković, Novak	12290	18
2	Nadal, Rafael	8670	20
...	...	...	...

## 1. pokušaj

```
SELECT rank() OVER (ORDER BY ukBodova DESC) rang
, igrac.ime || ', ' || igrac.prezime
, (SELECT SUM(bodovi)
   FROM turnirIgrac
   WHERE turnirIgrac.sifIgrac = igrac.sifIgrac) AS ukBodova
, (SELECT COUNT(*)
   FROM turnirIgrac
   WHERE turnirIgrac.sifIgrac = igrac.sifIgrac) AS ukTurnira
FROM igrac
ORDER BY rang
```

expr smije biti jednostavan izraz, ne i alias-ime izlaznog stupca ili njegov redni broj

```
ERROR: column „ukbodova" does not exist
LINE 1: SELECT rank() OVER (ORDER BY ukbodova DESC) rang
```

# PostgreSQL: Primjer s ispita iz rujna 2014

igrac

sifIgrac	ime	prezime
----------	-----	---------

turnir

sifTurnir	naziv
-----------	-------

turnirIgrac

godina	sifTurnir	sifIgrac	bodovi
--------	-----------	----------	--------

Potrebno je ispisati rang listu igrača pri čemu se rang odredi temeljem ukupnog osvojenog broja bodova na Grand Slam turnirima. Dodatno, ispisati i ukupan broj Grand Slam turnira na kojima je igrač sudjelovao u cijeloj karijeri (ukTurnira).

rang	igrac	ukBodova	ukTurnira
1	Đoković, Novak	12290	18
2	Nadal, Rafael	8670	20
...	...	...	...

## 2. pokušaj

```
SELECT rank() OVER (ORDER BY SUM(osvojioBodova) DESC) rang
      , igrac.ime || ', ' || igrac.prezime
      , SUM(osvojioBodova)
      , COUNT(*) OVER (PARTITION BY turnirIgrac.sifIgrac) ukTurnira
FROM turnirIgrac JOIN igrac
  ON turnirIgrac.sifIgrac = igrac.sifIgrac
GROUP BY igrac.ime, igrac.prezime
ORDER BY rang
```

Vrijednosti nad n-torkama u prozoru se obavljaju kad se obave:

- spajanja,
- WHERE
- GROUP BY
- HAVING

turnirIgrac.sifIgrac ne postoji u međurezultatu pa se ni prozori ne mogu napraviti obzirom na taj atribut

```
ERROR: column "turnirigrac.sifigrac" must appear in the GROUP BY clause or be used in an
aggregate function
LINE 4:      , COUNT(*) OVER (PARTITION BY turnirIgrac.sifIgrac) ukT...
```

# PostgreSQL: Primjer s ispita iz rujna 2014

igrac

sifIgrac	ime	prezime
----------	-----	---------

turnir

sifTurnir	naziv
-----------	-------

turnirIgrac

godina	sifTurnir	sifIgrac	bodovi
--------	-----------	----------	--------

Potrebno je ispisati rang listu igrača pri čemu se rang odredi temeljem ukupnog osvojenog broja bodova na Grand Slam turnirima. Dodatno, ispisati i ukupan broj Grand Slam turnira na kojima je igrač sudjelovao u cijeloj karijeri (ukTurnira).

rang	Igrac	ukBodova	ukTurnira
1	Đoković, Novak	12290	18
2	Nadal, Rafael	8670	20
...	...	...	...

Točno rješenje:

```
SELECT rank() OVER (ORDER BY SUM(osvojioBodova) DESC) rang
      , igrac.Ime || ', ' || igrac.prezime
      , SUM(osvojioBodova)
      , COUNT(*) OVER (PARTITION BY igrac.Ime, igrac.prezime) ukTurnira
FROM turnirIgrac JOIN igrac
    ON turnirIgrac.sifIgrac = igrac.sifIgrac
GROUP BY igrac.Ime, igrac.prezime
ORDER BY rang
--ORDER BY rank() OVER (ORDER BY SUM(osvojioBodova) DESC) --isto ok
```

```
SELECT rank() OVER (ORDER BY SUM(osvojioBodova) DESC) rang
      , igrac.Ime || ', ' || igrac.prezime
      , SUM(osvojioBodova)
      , COUNT(*) OVER (PARTITION BY igrac.sifIgrac) ukTurnira
FROM turnirIgrac JOIN igrac
    ON turnirIgrac.sifIgrac = igrac.sifIgrac
GROUP BY igrac.sifIgrac, igrac.Ime, igrac.prezime
ORDER BY rang
```

ili

# PostgreSQL: Funkcije za distribuciju

cume_dist ()	Relativni rang n-torke: (broj n-torki koje joj prethode ili su "vršnjaci,")/(ukupan broj n-torki u prozoru)
ntile (num_buckets integer)	Rezultat je cijeli broj s vrijednošću od 1 do vrijednosti argumenta kojim se prozor dijeli na jednake dijelove (koliko je to moguće)

```
SELECT sifPred, sifStud, SUM(Bod) ukBod
      , rank()      OVER (PARTITION BY sifPred
                          ORDER BY SUM(bod) DESC) rPred
      , rank()      OVER (ORDER BY SUM(bod) DESC) rGlob
      , cume_dist() OVER (PARTITION BY sifPred
                          ORDER BY SUM(bod) DESC) cdPred
      , cume_dist() OVER (ORDER BY SUM(bod) DESC) cdGlob
FROM studProvjera
GROUP BY sifPred, sifStud
ORDER BY sifPred, SUM(bod) DESC
--ORDER BY sifPred, ukBod DESC
```

sifPred	sifStud	ukBod	rPred	rGlob	crPred	cdGlob
1	3	25.00	1	2	0.333	0.5
1	1	15.00	2	5	1	1
1	2	15.00	2	5	1	1
2	3	30.00	1	1	0.333	0.166
2	2	25.00	2	2	0.666	0.5
2	1	20.00	3	4	1	0.666

studProvjera

sifPred	sifStud	kratProvjera	bod
1	1	1DZ	5.00
1	2	1DZ	10.00
1	3	1DZ	5.00
1	1	MI	10.00
1	2	MI	5.00
1	3	MI	20.00
...	...	...	...
2	1	MI	20.00
2	2	MI	25.00
2	3	MI	30.00

## PostgreSQL: Funkcije za određivanje n-torke temeljem pozicije

<code>lag(value any [,offset integer [,default any ]])</code>	<ul style="list-style-type: none"><li>• Vraća vrijednost koja je izračunata za n-torku koja se nalazi u istoj particiji <i>offset</i> n-torki <b>prije</b> tekuće</li><li>• Ako takva n-torka ne postoji, vraća <i>default</i> vrijednost</li><li>• I <i>offset</i> i <i>default</i> se određuju u odnosu na tekuću n-torku.</li><li>• Predodređene vrijednosti su 1 za <i>offset</i> i <i>NULL</i> za <i>default</i></li></ul>
<code>lead(value any [,offset integer [,default any ]])</code>	Jednako kao lag osim što n-torku traži <i>offset</i> n-torki <b>nakon</b> tekuće
<code>first_value(value any)</code>	Vraća vrijednost koja je izračunata za <b>prvu</b> n-torku u okviru
<code>last_value(value any)</code>	Vraća vrijednost koja je izračunata za <b>zadnju</b> n-torku u okviru
<code>nth_value(value any, nth integer)</code>	<ul style="list-style-type: none"><li>• Vraća vrijednost koja je izračunata za n-torku koja je na <i>nth</i>-oj poziciji u okviru (počinje od 1)</li><li>• Ako takva n-torka ne postoji, vraća <i>NULL</i> vrijednost</li></ul>

# PostgreSQL: Određivanje n-torke temeljem pozicije - primjer

Na temelju dnevnika pogrešaka donosi se odluka o prioritetu popravljavanja pogrešaka.

Da bi se pogreška našla na listi prioriteta, mora se istom korisniku pojaviti barem 3 puta u periodu od 15 minuta. Odrediti treba li pogrešku 75 staviti na listu prioriteta.

dnevnikGreska

sifDnevnik	sifGreska	vrijemeGreska	korisnik
10566	75	25.07.2013 10:18:00	pjuric
10567	75	25.07.2013 10:18:20	tkralj
10568	75	25.07.2013 10:18:50	tkralj
10569	75	25.07.2013 10:23:00	tkralj
10570	75	25.07.2013 10:28:30	tkralj
10571	75	25.07.2013 10:31:00	tkralj
10572	75	25.07.2013 10:33:40	tkralj

Svakom zapisu dnevnika s greškom 75 pronaći zapis koji je, obzirom na vrijeme greške, udaljen od njega za 2 pozicije (zajedno s promatranim zapisom radi se o točno 3 greške). Zapis mora pripadati istom korisniku.

```
SELECT sifDnevnik, sifGreska, korisnik, vrijemeGreska
      , LAG(vrijemeGreska, 2) OVER
          (PARTITION BY korisnik
            ORDER BY vrijemeGreska ASC) AS istaGreska2pozPrije
FROM dnevnikGreska
WHERE sifGreska = 75;
```

sifDnevnik	sifGreska	korisnik	vrijemeGreska	istaGreska2pozPrije
10566	75	pjuric	25.07.2013 10:18:00	
10567	75	tkralj	25.07.2013 10:18:20	
10568	75	tkralj	25.07.2013 10:18:50	
10569	75	tkralj	25.07.2013 10:23:00	25.07.2013 10:18:20
10570	75	tkralj	25.07.2013 10:28:30	25.07.2013 10:18:50
10571	75	tkralj	25.07.2013 10:31:00	25.07.2013 10:23:00
10572	75	tkralj	25.07.2013 10:33:40	25.07.2013 10:28:30

# PostgreSQL: Određivanje n-torke temeljem pozicije - primjer

sifDnevnik	sifGreska	korisnik	vrijemeGreska	istaGreska2pozPrije
10566	75	pjuric	25.07.2013 10:18:00	
10567	75	tkralj	25.07.2013 10:18:20	
10568	75	tkralj	25.07.2013 10:18:50	
10569	75	tkralj	25.07.2013 10:23:00	25.07.2013 10:18:20
10570	75	tkralj	25.07.2013 10:28:30	25.07.2013 10:18:50
10571	75	tkralj	25.07.2013 10:31:00	25.07.2013 10:23:00
10572	75	tkralj	25.07.2013 10:33:40	25.07.2013 10:28:30

Postoje li zapisi za koje vrijedi da se u manje od 15 minuta ista greška istom korisniku pojavila još 2 puta?

```
SELECT *
FROM
  (SELECT sifDnevnik, sifGreska, korisnik, vrijemeGreska
    , LAG(vrijemeGreska, 2) OVER
      (PARTITION BY korisnik
        ORDER BY vrijemeGreska ASC) AS istaGreska2PozPrije
    FROM dnevnikGreska
    WHERE sifGreska = 75) AS greska75
WHERE greska75.istaGreska2PozPrije IS NOT NULL
AND greska75.vrijemeGreska - greska75.istaGreska2PozPrije < '15 minutes';
```

sifDnevnik	sifGreska	korisnik	vrijemeGreska	istaGreska2pozPrije
10570	75	tkralj	2013-07-25 10:28:30	2013-07-25 10:18:00
10571	75	tkralj	2013-07-25 10:31:00	2013-07-25 10:18:50
10572	75	tkralj	2013-07-25 10:33:40	2013-07-25 10:23:00

## Napredni SQL - teme

1. Pivotiranje
2. Funkcije za rad s podacima u prozoru
3. **Rekurzivni upiti i CTE (Common Table Expression)**



# Motivacijski primjer 1

Pronađi sve osobe s kojima je **ana123@hotmail.com** povezana direktno ili preko prijatelja. Uzeti u obzir samo prijateljstva u kojima je **ana123@hotmail.com** (odnosno njeni prijatelji i td.) inicijator (osoba1).

foaf

Friend of a friend

osoba1	osoba2
...	...
ana123@hotmail.com	zecG@gmail.com
zecG@gmail.com	iva.malic@fer.hr
klaraB@gmail.com	ana123@hotmail.com
zecG@gmail.com	jezV@hotmail.com
Iva.malic@fer.hr	jezV@hotmail.com
...	...

Prijatelji

Prijatelji  
prijatelja

Prijatelji  
prijatelja  
prijatelja

```
SELECT osoba1, osoba2
  FROM foaf foaf1
 WHERE foaf1.osoba1 = 'ana123@hotmail.com'
 UNION
 SELECT foaf2.osoba1, foaf2.osoba2
  FROM foaf foaf1, foaf foaf2
 WHERE foaf1.osoba2 =foaf2.osoba1
       AND foaf1.osoba1 = 'ana123@hotmail.com'
 UNION
 SELECT foaf3.osoba1, foaf3.osoba2
  FROM foaf foaf1, foaf foaf2, foaf foaf3
 WHERE foaf1.osoba2 =foaf2.osoba1
       AND foaf2.osoba2 =foaf3.osoba1
       AND foaf1.osoba1 = 'ana123@hotmail.com'
 UNION ...
```

SQL upitom bismo znali pronaći sve prijatelje osobe **ana123@hotmail.com**, sve prijatelje prijatelja itd.

- Znali bismo napisati (nespretno, nepregledano) upit za unaprijed zadanu "dubinu" veze.
- Za nepoznatu/promjenjivu "dubinu" veze problem se svodi na rekurzivnu pretragu relacije *foaf*.

## Motivacijski primjer 2

Odredi ukupna primanja zaposlenih na projektu sa šifrom 234. Smatra se da na projektu radi djelatnik nadležan za projekt ali i svi djelatnici kojima je on nadređen.

djelatnik

sifDjel	ime	prezime	dohodak	sifNadrDjel
1	Ana	Hlad	5750.00	
2	Petar	Milić	4843.00	1
3	Mia	Horvat	7540.00	1
4	Ivan	Kralj	12050.00	2
5	Marko	Car	6890.00	2
6	Hrvoje	Kolar	4980.00	3

projekt

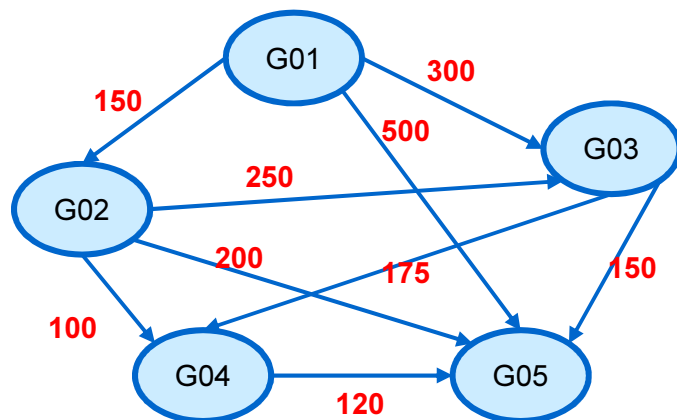
sifprojekt	...	sifNadIDjel
...	...	...
234	...	1
764	...	2
...	...	...

Znali bismo napisati upit za unaprijed zadanu "dubinu" hijerarhijske veze djelatnik-nadređeni djelatnik.

Za nepoznatu dubinu hijerarhijske veze djelatnik-nadređeni djelatnik – rekurzivna pretraga relacije ***djelatnik*** .

## Motivacijski primjer 3

Pronađi najjeftiniji let između polazišta A i odredišta B.



let

polaziste	odrediste	cijena
G01	G02	150
G01	G03	300
G01	G05	500
G02	G03	250
G02	G04	100
G02	G05	200
G03	G04	175
G03	G05	150
G04	G05	120

Uzevši u obzir moguća presjedanja problem se svodi na rekurzivnu pretragu relacije **let**.

# Common Table Expressions (CTE)

- SQL:1999 (SQL3) standard definira WITH naredbu:

```
WITH R1 AS (query1),  
     R2 AS (query2),  
     ...,  
     Rn AS (queryn),  
<upit koji uključuje R1, R2,...,Rn (i druge relacije)>
```

- Predviđena je i sljedeća sintaksa

```
WITH R1(A1, A2, ..., Am) AS (query1), ...
```

Ideja:

1. Odrediti sadržaj relacija R1, R2,..., Rn i pohraniti ga u privremene relacije (TEMP TABLE)
2. Procesirati upit koji uključuje R1, R2,..., Rn i druge relacije
3. Ukloniti R1, R2,..., Rn

Ovakvi izrazi se nazivaju Common Table Expressions (CTE)

# Rekurzivni upiti

- WITH odnosno CTE omogućuju implementaciju rekurzivnih upita:

```
WITH RECURSIVE
  R1 AS (query1),
  R2 AS (query2),
  ...,
  Rn AS (queryn),
<upit koji uključuje R1, R2,...,Rn (i druge relacije)>
```

- R1, R2,..., Rn mogu biti rekurzivni ili međusobno rekurzivni:
- rekurzija se obično postiže pomoću **UNION** temeljnih relacija i ključne riječi **RECURSIVE**

```
WITH RECURSIVE
  R AS (upit
        UNION
        rekurzivni upit)
<upit koji uključuje R (i druge relacije)>
```

## Primjer 1 - foaf

Pronađi sve osobe s kojima je osoba **ana123@hotmail.com** povezana direktno ili preko prijatelja. Uzeti u obzir samo prijateljstva u kojima je **ana123@hotmail.com** (odnosno njeni prijatelji i td.) inicijator (osoba1).

foaf

osoba1	osoba2
...	...
ana123@hotmail.com	zecG@gmail.com
zecG@gmail.com	iva.malic@fer.hr
klaraB@gmail.com	ana123@hotmail.com
zecG@gmail.com	jezV@hotmail.com
Iva.malic@fer.hr	jezV@hotmail.com
...	...

Neposredni prijatelji:

osoba1	osoba2
ana123@hotmail.com	zecG@gmail.com

Prijatelji prijatelja :

osoba1	osoba2	
ana123@hotmail.com	iva.malic@fer.hr	preko zecG@gmail.com
ana123@hotmail.com	jezV@hotmail.com	preko zecG@gmail.com
ana123@hotmail.com	jezV@hotmail.com	preko zecG@gmail.com, iva.malic@fer.hr

# PostgreSQL: Primjer 1 - foaf

```
WITH RECURSIVE prijatelji(prijatelj1, prijatelj2)
AS
(
    SELECT osoba1, osoba2
      FROM foaf
)
SELECT * FROM prijatelji
```

s UNION

```
WITH RECURSIVE prijatelji(prijatelj1, prijatelj2)
AS
(
    SELECT osoba1, osoba2
      FROM foaf

    UNION

    SELECT prijatelji.prijatelj1
      , foaf.osoba2 as prijatelj2
      FROM prijatelji, foaf
    WHERE prijatelji.prijatelj2 = foaf.osoba1
)
SELECT * FROM prijatelji
```

Rezultat (jednak sadržaju relacije foaf):

prijatelj1	prijatelj2
ana123@hotmail.com	zecG@gmail.com
zecG@gmail.com	iva.malic@fer.hr
klaraB@gmail.com	ana123@hotmail.com
zecG@gmail.com	jezV@hotmail.com
Iva.malic@fer.hr	jezV@hotmail.com

1. prolaz


2. prolaz

3. prolaz

prijatelj1	prijatelj2
ana123@hotmail.com	zecG@gmail.com
zecG@gmail.com	iva.malic@fer.hr
klaraB@gmail.com	ana123@hotmail.com
zecG@gmail.com	jezV@hotmail.com
Iva.malic@fer.hr	jezV@hotmail.com
ana123@hotmail.com	iva.malic@fer.hr
ana123@hotmail.com	jezV@hotmail.com
zecG@gmail.com	jezV@hotmail.com
klaraB@gmail.com	zecG@gmail.com
klaraB@gmail.com	iva.malic@fer.hr
klaraB@gmail.com	jezV@hotmail.com

# PostgreSQL: Primjer 1 - foaf

```
WITH RECURSIVE prijatelji(prijatelj1, prijatelj2)
AS
(
    SELECT osoba1, osoba2
      FROM foaf

    UNION 

    SELECT prijatelji.prijatelj1
          , foaf.osoba2  as prijatelj2
      FROM prijatelji, foaf
     WHERE prijatelji.prijatelj2 = foaf.osoba1
)
SELECT * FROM prijatelji
 WHERE prijatelj1 = 'ana123@hotmail.com'
```

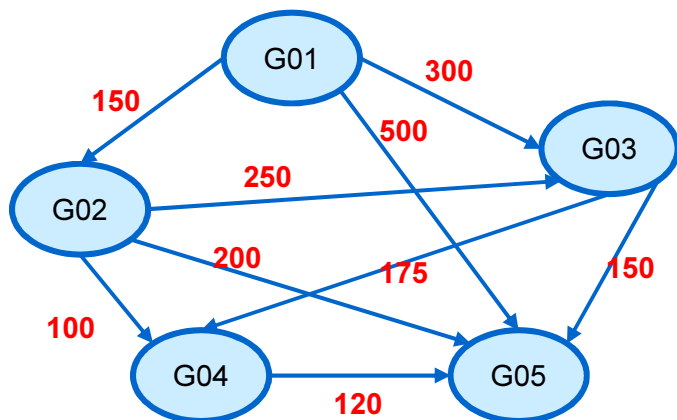
prijatelj1	prijatelj2
ana123@hotmail.com	zecG@gmail.com
ana123@hotmail.com	iva.malic@fer.hr
ana123@hotmail.com	jezV@hotmail.com

Za vježbu: ispisati i osobe s kojima je **ana123@hotmail.com** povezana na njihovu inicijativu te njihove prijatelje rekurzivno (npr. klaraB@gmail.com)



## Primjer 2 – najjeftiniji plan leta

Pronađi sve moguće planove letenja i pripadnu cijenu između polazišta G01 i odredišta G05. Prikaži i broj presjedanja.



Rezultat (mogući):

ruta	cijena	brPresjedanja
G01-->G05	500	0
G01-->G02-->G03-->G04-->G05	695	3
G01-->G02-->G03-->G05	550	2
G01-->G02-->G04-->G05	370	2
G01-->G02-->G05	350	1
G01-->G03-->G04-->G05	595	2
G01-->G03-->G05	450	1

let

polaziste	odrediste	cijena
G01	G02	150
G01	G03	300
G01	G05	500
G02	G03	250
G02	G04	100
G02	G05	200
G03	G04	175
G03	G05	150
G04	G05	120

## Primjer 2 – najjeftiniji plan leta

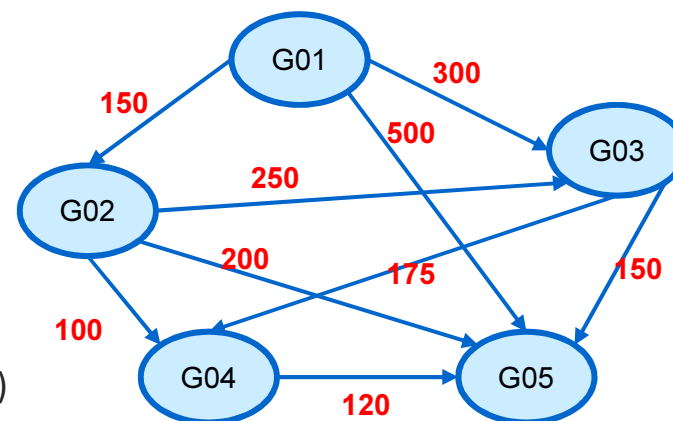
### Ideja:

Pomoću rekurzivnog upita stvoriti privremenu relaciju

`sviPlanoviLetenja` (`polaziste`,  
`odrediste`,  
`planLeta`,  
`ukCijena`,  
`brPresj`)

gdje je

- **planLeta** znakovni niz oblika G01-->G02-->G03-->G04-->G05, a
- **ukCijena** ukupna cijena danog plana leta (695 za gornji plan)



### 1. korak

U `sviPlanoviLetenja` insertirati n-torke sljedećeg oblika:

polaziste	odrediste	planLeta	ukCijena	brPresj
	G01	G01	0	-1
	G02	G02	0	-1
	G03	G03	0	-1
	G04	G04	0	-1

### 3. korak

U `sviPlanoviLetenja` insertirati n-torke sljedećeg oblika:

polaziste	odrediste	planLeta	ukCijena	brPresj
G02	G03	G01-->G02-->G03	400	1
G02	G04	G01-->G02-->G04	250	1
G02	G05	G01-->G02-->G05	350	1
...	...	...	...	1

### 2. korak

U `sviPlanoviLetenja` insertirati n-torke sljedećeg oblika:

polaziste	odrediste	planLeta	ukCijena	brPresj
G01	G02	G01-->G02	150	0
G01	G03	G01-->G03	300	0
G01	G05	G01-->G05	500	0
G02	G03	G02-->G03	250	0
G02	G04	G02-->G04	100	0
G02	G05	G02-->G05	200	0
G03	G04	G03-->G04	175	0
G03	G05	G03-->G05	150	0
G04	G05	G04-->G05	120	0

### 4. korak

U `sviPlanoviLetenja` insertirati n-torke ...

## PostgreSQL: Primjer 2 –najjeftiniji plan leta

Pronađi sve moguće planove letenja i pripadnu cijenu između polazišta G01 i odredišta G05.  
Prikaži i broj presjedanja.

```
WITH RECURSIVE sviPlanoviLetenja
  (polaziste, odrediste, planLeta, ukCijena, brPresjedanja)
AS
  ( (SELECT DISTINCT
      '',
      polaziste,
      CAST (polaziste AS VARCHAR(600)),
      0,
      -1
    FROM let
  )
  UNION ALL
  (SELECT child.polaziste,
    child.odrediste,
    CAST (parent.planLeta || '-->' || CAST (SUBSTR(child.odrediste, 1,3) AS CHAR(3))
      AS VARCHAR(600)),
    parent.ukCijena + child.Cijena,
    parent.brPresjedanja + 1
    FROM sviPlanoviLetenja parent, let child
    WHERE parent.odrediste = child.polaziste)
  )
SELECT planLeta, ukCijena, brPresjedanja
  FROM sviPlanoviLetenja
 WHERE sviPlanoviLetenja.brPresjedanja >= 0
   AND sviPlanoviLetenja.planLeta LIKE 'G01%'
   AND sviPlanoviLetenja.planLeta LIKE '%G05'
 ORDER BY ukCijena
```

The diagram illustrates a network of flight paths between five airports: G01, G02, G03, G04, and G05. Each airport is represented by a blue oval. The flight paths are represented by blue arrows, and the cost of each flight is indicated by a red number next to the arrow. The costs are as follows:

- G01 to G02: 150
- G01 to G03: 300
- G01 to G04: 500
- G02 to G03: 250
- G02 to G04: 200
- G02 to G05: 175
- G03 to G05: 150
- G04 to G05: 120

# Beskonačna rekurzija

foaf

osoba1	osoba2
ana123@hotmail.com	zecG@gmail.com
zecG@gmail.com	iva.malic@fer.hr
klaraB@gmail.com	ana123@hotmail.com
zecG@gmail.com	jezV@hotmail.com
iva.malic@fer.hr	jezV@hotmail.com
ana123@hotmail.com	ana123@hotmail.com

← Uzrokovat će beskonačnu petlju

## PostgreSQL

```
WITH RECURSIVE prijatelji
  (prijatelj1, prijatelj2) AS
  ( (SELECT osoba1, osoba2 FROM foaf)
    UNION
    (SELECT prijatelj1, osoba2
     FROM foaf, prijatelji
     WHERE prijatelji.prijatelj2 = foaf.osoba1)
  )
SELECT * FROM prijatelji
WHERE prijatelj1 = 'ana123@hotmail.com'
```

Prepozna beskonačnu rekurziju i uspije je razriješiti.

Dobije se sljedeći rezultat:

prijatelj1	prijatelj2
ana123@hotmail.com	zecG@gmail.com
ana123@hotmail.com	iva.malic@fer.hr
ana123@hotmail.com	jezV@hotmail.com
ana123@hotmail.com	ana123@hotmail.com

## SQL Server

```
WITH prijatelji
  (prijatelj1, prijatelj2) AS
  ( (SELECT osoba1, osoba2 FROM foaf)
    UNION ALL
    (SELECT prijatelj1, osoba2
     FROM foaf, prijatelji
     WHERE prijatelji.prijatelj2 = foaf.osoba1)
  )
SELECT * FROM prijatelji
WHERE prijatelj1 = 'ana123@hotmail.com'
```

Msg 530, Level 16, State 1, Line 1 The statement terminated.  
The maximum recursion 100 has been exhausted before  
statement completion.



## Rekurzivni upiti u relacijskim sustavima za upravljanje BP

- Microsoft SQL Server 2005 +
- Oracle 11g
- IBM DB2 v2
- PostgreSQL 8.4+
- Firebird 2.1