



Poglavje V

Relacijsko poizvedovanje

Napredni SQL

- Stične operacije
- Delo z množicami
- Gnezdenje poizvedb
- Skupinske poizvedbe

Računanje stika...



- Stik predstavlja podmnožico kartezičnega produkta
- Če pri navedbi dveh tabel A in B v FROM sklopu ne navedemo pogoja v WHERE sklopu, dobimo kartezični produkt med A in B.
- ISO standard ponuja tudi posebno obliko zapisa za kartezični produkt:

```
SELECT [DISTINCT | ALL] {* | columnList}  
FROM Table1 CROSS JOIN Table2
```

Računanje stika



- V splošnem si lahko postopek za generiranje rezultata SELECT stavka s stikom predstavljamo takole:
 - Sestavi kartezični produkt tabel, ki so naštete v sklopu FROM
 - Če obstaja WHERE sklop, upoštevaj vse pogoje in iz kartezičnega produkta izberi samo tiste vrstice, ki pogojem ustrezajo (selekcija); pogoj v WHERE sklopu je lahko zelo obširen
 - Iz izbranih vrstic kartezičnega produkta izberi samo tiste stolpce, ki ustrezajo naboru v SELECT sklopu (projekcija)
 - Če je bil uporabljen DISTINCT operator, eliminiraj dvojnike (del projekcije)
 - Če obstaja ORDER BY sklop, ustrezno razvrsti vrstice

Primer stika s sestavljenim pogojem



- Izpiši oznake in imena jadralcev, ki so rezervirali rdeče čolne na dan 10. 10. 2006.

```
SELECT j.jid, j.ime
FROM jadralec j, rezervacija r, coln c
WHERE  j.jid=r.jid AND r.cid=c.cid
      AND c.barva='rdeca'
      AND r.dan=DATE'2006-10-10';
```

```
+-----+-----+
|  jid  |  ime   |
+-----+-----+
|   22  | Darko  |
+-----+-----+
```

Stik več tabel v FROM vrstici



- Kartezični produkt:
 - CROSS JOIN
- Pogojni stik:
 - Splošno: JOIN ali INNER JOIN ... ON (pogoj)
 - Ekvistik: JOIN ali INNER JOIN ... USING (atribut, atribut, ...)
 - Naravni stik: NATURAL JOIN
- Zunanji stiki:
 - Levi zunanji stik: LEFT OUTER JOIN
 - Desni zunanji stik: RIGHT OUTER JOIN
 - Polni zunanji stik: FULL OUTER JOIN (isto kot $R \bowtie S \cup R \ltimes S$)

Stik več tabel v FROM vrstici



- SQL z operatorjem INNER JOIN omogoča alternativne načine stika med več tabelami:
 - FROM rezervacija r INNER JOIN jadralec j ON (r.jid=j.jid)
 - FROM rezervacija INNER JOIN jadralec USING (jid)
 - FROM rezervacija NATURAL JOIN jadralec
- INNER lahko izpustimo
- Zgornji zapisi nadomestijo sklopa FROM in WHERE
- V prvem primeru rezultat vsebuje dva identična stolpca jid, v drugem in tretjem pa ne
- Pozor: operatorji JOIN pogosto niso v celoti implementirani

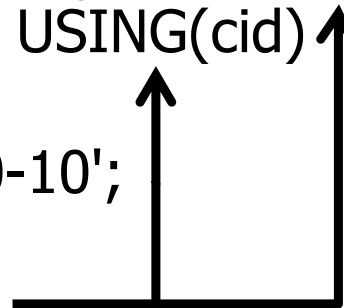
Primer poizvedbe po več tabelah

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

- Izpiši oznake in imena jadralcev, ki so rezervirali rdeče čolne na dan 10. 10. 2006.

```
SELECT j.jid, j.ime  
FROM   jadralec j JOIN rezervacija r USING (jid)  
      JOIN coln c USING(cid)  
WHERE  c.barva='rdeca' AND  
      r.dan=DATE'2006-10-10';
```

Zakaj ne NATURAL JOIN?



jid	ime
22	Darko

Zunanji stik...

- S pomočjo zunanjega stika dobimo v rezultat tudi vrstice, ki nimajo stične vrednosti v drugi tabeli.

R=OSEBA

ID	Priimek in ime	PTT
1	Kante Janez	5270
2	Tratnik Jože	5000
3	Mali Mihael	
4	Brecelj Jana	1000

S=KRAJ

PTT	Naziv
1000	Ljubljana
5000	Nova Gorica
5270	Ajdovščina

$(\Pi_{\text{Priimek in ime, PTT, Naziv zač.preb.}}(R)) \bowtie S$

Priimek in ime	PTT	Naziv zač. preb.
Kante Janez	5270	Ajdovščina
Tratnik Jože	5000	Nova Gorica
Mali Mihael		
Brecelj Jana	1000	Ljubljana

Levi zunanji stik (left outer join)

- Za zapis SELECT stavka, ki vsebuje zunanji stik med dvema tabelama, uporabimo naslednjo sintakso:

```
SELECT  DISTINCT ime, cid
FROM    jadralec LEFT OUTER JOIN rezervacija
        USING (jid);
```

- OUTER lahko izpustimo Kaj izpiše zgornja poizvedba?

ime	cid
Darko	101
Darko	102
Darko	103
Darko	104
Borut	NULL
Lojze	102
Lojze	103
Lojze	104
Andrej	NULL
Rajko	NULL
Henrik	101
Henrik	102
Zdravko	NULL
Henrik	103
Anze	NULL
Bine	NULL

- | ime | cid |
|---------|------|
| Darko | 101 |
| Darko | 102 |
| Darko | 103 |
| Darko | 104 |
| Borut | NULL |
| Lojze | 102 |
| Lojze | 103 |
| Lojze | 104 |
| Andrej | NULL |
| Rajko | NULL |
| Henrik | 101 |
| Henrik | 102 |
| Zdravko | NULL |
| Henrik | 103 |
| Anze | NULL |
| Bine | NULL |

```
SELECT DISTINCT ime, cid
FROM jadralec FULL OUTER JOIN rezervacija
      USING (jid);
```

Gnezdenje poizvedb...



- Nekateri SELECT stavki vsebujejo tudi vgnezdene SELECT stavke.
- Rezultat vgnezdenega SELECT stavka se lahko uporabi kot množica (z operatorji IN, EXISTS, ALL, ANY) v FROM, WHERE ali HAVING sklopih drugega SELECT stavka (subselect)
- Vgnezdeni SELECT stavki se lahko pojavijo tudi v INSERT, UPDATE in DELETE stavkih.

Primer vgnezdenega SELECT stavka

- Izpiši imena jadralcev, ki so rezervirali čoln modre barve.

```
SELECT j.ime  
FROM jadralec j  
WHERE j.jid IN  
  (SELECT jid  
   FROM coln c, rezervacija r  
   WHERE c.cid=r.cid AND  
         c.barva='modra');
```

množica šifer jadralcev,
ki so rezervirali čoln
modre barve

+	-----	+
	ime	
+	-----	+
	Darko	
	Henrik	
+	-----	+

Pravila gnezdenja SELECT stavkov...



- Načeloma naj (zaradi učinkovitosti) vgnezdjeni SELECT stavki ne uporabljajo ORDER BY ali DISTINCT (tudi nima smisla, saj se rezultat obravnava kot množica).
- Če je mogoče, naj (zaradi učinkovitosti primerjav) SELECT sklop vgnezdenega SELECT stavka zajema samo en stolpec (najpogostejše PK),
 - razen v primeru uporabe ukaza EXISTS.
 - Pogoste izjeme
- Večkratna primerjava (nestandardne implementacije):
WHERE (A, B) IN (SELECT A, B FROM ...)

Pravila gnezdenja SELECT stavkov...



- Imena stolpcev v vgnezenem SELECT stavku se privzeto nanašajo na tabele iz vgnezenega ali zunanjega SELECT stavka (v tem vrstnem redu)
- Priporočljiva (praktično nujna) je eksplicitna uporaba sinonimov za tabele (aliasov) pri vseh atributih

Pravila gnezdenja SELECT stavkov



- Ko je vgnezen SELECT stavek operand v primerjavi, se mora nahajati na desni strani enačbe: `WHERE x = (SELECT ...);`
- Vgnezeni SELECT stavek ne more biti operand v izrazu (po standardu; PostgreSQL in MariaDB to dopuščata).

`WHERE x = 2 * (SELECT ...) + 1;`

Množice: kvantifikacija v SQL



- Kvantifikacija logičnega pogoja (eksplicitna z ALL ali ANY)
- Kvantifikacija izpolnjevanja logičnega pogoja v WHERE sklopu:
 - vedno eksistenčna
 - univerzalno kvantifikacijo implementiramo s pomočjo dvojne negacije

Kvantifikatorja ANY in ALL



- V vgnezenih SELECT stavkih, ki vračajo en sam stolpec, lahko uporabljamo operatorja ANY in ALL, da preverimo veljavnost logičnega pogoja na celotni množici rezultatov (kvantifikacija).
- Z uporabo ALL bo pogoj izpolnjen samo, če bo veljal za vse vrednosti, ki ji vrača poizvedba (\forall).
- Z uporabo ANY, bo pogoj izpolnjen, če bo veljal za vsaj eno od vrednosti, ki ji poizvedba vrača (\exists).
- Če je rezultat poizvedbe prazen, bo ALL vrnil true, ANY pa false.
- Namesto ANY lahko uporabljamo tudi SOME.

Primer uporabe ANY

- Izpiši imena čolnov, ki so daljši od vsaj enega rdečega čolna.

```
SELECT ime
FROM coln
WHERE dolzina > ANY( SELECT dolzina
                     FROM coln
                     WHERE barva='rdeca' );
```

```
+-----+
| ime      |
+-----+
| Sun Odyssey |
| Bavaria   |
+-----+
```

Primer uporabe ALL

- Izpiši imena čolnov, ki so daljši od vseh zelenih čolnov.

```
SELECT ime
FROM coln
WHERE dolzina > ALL
      (SELECT dolzina
       FROM coln
       WHERE barva='zelena');
```

```
+-----+
| ime    |
+-----+
| Bavaria|
+-----+
```

Primer

- Z ALL, ANY in gnezdenjem:
 - Izpišite šifro jadralca z najvišjim ratingom
 - Izpišite šifro jadralca z ne najnižjim ratingom

```
SELECT jid
FROM jadralec
WHERE rating >= ALL
      (SELECT rating
       FROM jadralec);
```

```
+-----+
| jid |
+-----+
|  58 |
|  71 |
+-----+
```

```
SELECT jid
FROM jadralec
WHERE rating > ANY
      (SELECT rating
       FROM jadralec);
```

```
+-----+
| jid |
+-----+
|  22 |
|  31 |
|  32 |
|  58 |
|  64 |
|  71 |
|  74 |
|  85 |
|  95 |
+-----+
```

Uporaba EXISTS in NOT EXISTS



- EXISTS in NOT EXISTS lahko uporabljamo le v vgnezenih poizvedbah.
- Vračata logičen rezultat true/false.
- EXISTS (...)
 - true dobimo, če obstaja vsaj ena vrstica v tabeli, ki je rezultat vgnezdene poizvedbe.
 - false dobimo, če vgnezdena poizvedba vrača prazno množico.
- NOT EXISTS je negacija rezultata EXISTS.

Uporaba EXISTS in NOT EXISTS



- (NOT) EXISTS preveri samo, če v rezultatu vgnezdene poizvedbe (ne) obstajajo vrstice
- Število stolpcev v SELECT sklopu vgnezdene poizvedbe je zato nepomembno, projekcija nepotrebna
- Navadno uporabimo sintakso:
... EXISTS (SELECT * FROM ...)

Primer uporabe EXISTS

- Izpiši vse jadralce, ki so kadarkoli rezervirali čoln Bavaria.

```
SELECT *  
FROM jadralec j  
WHERE EXISTS
```

```
( SELECT *
```

```
FROM rezervacija r, coln c
```

```
WHERE
```

```
r.jid = j.jid AND
```

```
r.cid = c.cid AND
```

```
c.ime = 'Bavaria' );
```

jid	ime	rating	starost
22	Darko	7	45
31	Lojze	8	55.5

Kaj se zgodi, če ta
pogoj izpustimo?

Korelirana
vgnezdena
poizvedba.

Primer uporabe EXISTS

- Izpiši vse jadralce, ki so kadarkoli rezervirali čoln Bavaria.

```
SELECT *  
FROM jadralec j  
WHERE EXISTS  
  ( SELECT *  
    FROM rezervacija r, coln c  
    WHERE r.jid = j.jid AND  
          r.cid = c.cid AND  
          c.ime = 'Bavaria' );
```

jid	ime	rating	starost
22	Darko	7	45
29	Borut	1	33
31	Lojze	8	55.5
32	Andrej	8	25.5
58	Rajko	10	35
64	Henrik	7	35
71	Zdravko	10	16
74	Henrik	9	35
85	Anze	3	25.5
95	Bine	3	63.5

Izpiše podatke o jadralcu, če obstaja vsaj ena rezervacija čolna Bavaria (ne nujno od istega jadralca)

Namesto EXISTS lahko tu uporabimo stik



```
SELECT DISTINCT j.*  
FROM jadralec j, rezervacija r, coln c  
WHERE j.jid = r.jid AND r.cid = c.cid AND  
      c.ime = 'Bavaria';
```

+	-----	+	-----	+	-----	+	-----	+
	jid		ime		rating		starost	
+	-----	+	-----	+	-----	+	-----	+
	22		Darko		7		45	
	31		Lojze		8		55.5	
+	-----	+	-----	+	-----	+	-----	+

- Korelirane gnezdene poizvedbe zelo obremenjujejo SUPB in zato se jim želimo izogniti.
- Običajno (ne pa vedno) je to možno.

Uporaba operacij nad množicami...



- Rezultate dveh ali več poizvedb lahko kot množice združujemo z operatorji:
 - UNION (unija),
 - INTERSECT (presek)
 - EXCEPT ali MINUS (razlika)
EXCEPT (PostgreSQL, SQL Server, MariaDB), MINUS
- Da lahko izvajamo naštete operacije, morajo delni rezultati biti skladni (domene atributov morajo biti enake oz. primerljive).

Primer (nesmiselne) unije

- Izpiši ratinge jadralcev in dolžine čolnov.

```
(SELECT rating
FROM jadralec)
UNION
(SELECT dolzina
FROM coln);
```

Naziv stolpca rezultata
se povzame po prvem
SELECT stavku.
Tipa delnih rezultatov
morata biti primerljiva.

rating
7
1
8
10
9
3
34
37
50

Primer unije

- Izpiši imena vseh čolnov, ki jih je rezerviral jadralec Darko ali pa jadralec Lojze

```
SELECT c.ime    -- DISTINCT?
FROM coln c, rezervacija r, jadralec j
WHERE c.cid = r.cid AND r.jid = j.jid
      AND ( j.ime = 'Darko'
            OR j.ime='Lojze' );
```

```
+-----+
| ime    |
+-----+
| Elan   |
| Elan   |
| Elan   |
| Sun Odyssey |
| Sun Odyssey |
| Bavaria |
| Bavaria |
+-----+
```

Primer unije



- Izpiši oznake vseh čolnov, ki jih je rezerviral jadralec Darko ali pa jadralec Lojze (brez ponavljanja).

```
(SELECT c.cid
```

```
FROM coln c, rezervacija r, jadralec j
```

```
WHERE c.cid =r.cid AND r.jid = j.jid  
      AND j.ime = 'Darko')
```

```
UNION
```

```
(SELECT c.cid
```

```
FROM coln c, rezervacija r, jadralec j
```

```
WHERE c.cid =r.cid AND r.jid = j.jid AND j.ime = 'Lojze');
```

```
+-----+  
|  cid  |  
+-----+  
|  101  |  
|  102  |  
|  103  |  
|  104  |  
+-----+
```

Primer unije s ponovitvami

- Izpiši oznake vseh čolnov, ki jih je rezerviral jadralec Darko ali Lojze.

```
(SELECT c.cid  
FROM coln c, rezervacija r, jadralec j  
WHERE c.cid = r.cid AND r.jid = j.jid  
      AND j.ime = 'Darko')
```

UNION ALL

```
(SELECT c.cid  
FROM coln c, rezervacija r, jadralec j  
WHERE c.cid = r.cid AND r.jid = j.jid AND j.ime = 'Lojze');
```

```
+-----+  
|  cid  |  
+-----+  
|  101  |  
|  102  |  
|  103  |  
|  104  |  
|  102  |  
|  103  |  
|  104  |  
+-----+
```

Primer preseka

- Izpiši imena vseh čolnov, ki sta jih rezerviral jadralca Darko in Lojze.

```
SELECT c.ime  
FROM coln c,
```

rezervacija r,

jadralec j

```
WHERE c.cid = r.cid AND r.jid = j.jid
```

```
AND (j.ime = 'Darko' AND j.ime = 'Lojze');
```

		Darko			Lojze		
+	-----+		+	-----+		+	-----+
	jid	ime		rating		starost	
+	-----+		+	-----+		+	-----+
	22	Darko		7		45	
	29	Borut		1		33	
	31	Lojze		8		55.5	
	32	Andrej		8		25.5	
	58	Rajko		10		35	
	64	Henrik		7		35	
	71	Zdravko		10		16	
	74	Henrik		9		35	
	85	Anze		3		25.5	
	95	Bine		3		63.5	
+	-----+		+	-----+		+	-----+

Napačna raba konjunkcije:
jadralec ne more **istočasno**
biti Darko in Lojze!
Rezultat: prazna množica.

Primer preseka



- Izpiši oznake vseh čolnov, ki sta jih rezervirala jadralca Darko in Lojze.

```
(SELECT c.cid
```

```
FROM coln c, rezervacija r, jadralec j
```

```
WHERE c.cid =r.cid AND r.jid = j.jid
```

```
AND j.ime = 'Darko')
```

```
INTERSECT
```

```
(SELECT c.cid
```

```
FROM coln c, rezervacija r, jadralec j
```

```
WHERE c.cid =r.cid AND r.jid = j.jid AND j.ime = 'Lojze');
```

```
+-----+  
| cid |  
+-----+  
| 102 |  
| 103 |  
| 104 |  
+-----+
```


Presek v brez INTERSECT

- MySQL včasih ni poznal ukaza INTERSECT
- Nadomestimo z definicijo preseka:

$$x \in A \cap B \Rightarrow x \in A \wedge x \in B$$

A generiramo
B preverimo

```
SELECT DISTINCT c.cid
FROM coln c, rezervacija r, jadralec j
WHERE c.cid = r.cid AND r.jid = j.jid AND j.ime = 'Darko'
AND c.cid IN
(
  SELECT c.cid
  FROM coln c, rezervacija r, jadralec j
  WHERE c.cid = r.cid AND r.jid = j.jid AND j.ime = 'Lojze');
```

Namesto ukaza INTERSECT

```
+-----+
|  cid  |
+-----+
|  102  |
|  103  |
|  104  |
+-----+
```

Uporaba razlike: EXCEPT ali MINUS

- Izpiši oznake jadralcev, ki niso rezervirali nobenega čolna.

```
(SELECT jid
FROM jadralec)
EXCEPT
(SELECT jid
FROM rezervacija);
```

```
+-----+
| jid |
+-----+
|  29 |
|  32 |
|  58 |
|  71 |
|  85 |
|  95 |
+-----+
```

Razlika brez EXCEPT ali MINUS

- MySQL včasih ni poznal ukaza EXCEPT

- Nadomestimo z definicijo razlike:

$$x \in A - B \Rightarrow x \in A \wedge x \notin B$$

A {
SELECT jid
FROM jadralec
WHERE jid NOT IN
B {
(SELECT jid
FROM rezervacija);

```
+-----+  
|  jid  |  
+-----+  
|  29   |  
|  32   |  
|  58   |  
|  71   |  
|  85   |  
|  95   |  
+-----+
```

Namesto ukaza EXCEPT

Delna uporaba operacij nad množicami



- Sintaksa (ISO standard, redko podprto):
op [ALL] [CORRESPONDING [BY {column1 [, ...]}]]
- Če uporabimo CORRESPONDING BY, se operacija primerjanja izvede samo nad naštetimi stolpci
- Če uporabimo samo CORRESPONDING brez BY člena, se operacija izvede samo nad skupnimi stolpci.
- *Microsoft SQL* server podpira, *PostgreSQL* in *MySQL* pa ne podpirata CORRESPONDING BY
- Če uporabimo ALL, lahko rezultat vključuje tudi dvojnike (UNION, EXCEPT)

Delna uporaba operacij nad množicami

- Izpiši vse podatke jadralcev, ki niso rezervirali nobenega čolna.

```
(SELECT *  
FROM jadralec)  
EXCEPT    -- ????  
(SELECT *  
FROM rezervacija);
```


Tako ne bo šlo. Ideje?

```
(SELECT *  
FROM jadralec)  
EXCEPT CORRESPONDING BY jid  
(SELECT jid  
FROM rezervacija);
```

Kako bi to naredili brez
CORRESPONDING BY?

Kvantifikacija izpolnjevanja logičnega pogoja v WHERE _P_ sklopu:

- Nad tabelo preverjamo logični pogoj P
- Vedno eksistenčna kvantifikacija glede na množico vrstic v rezultatu (kdaj bo neprazen).



```
SELECT DISTINCT r.jid
FROM rezervacija r, coln c
WHERE r.cid = c.cid
AND c.barva = 'rdeca';
```

\exists vrstica: P(vrstica)

npr. „vsaj enkrat rezervira rdeč čoln“

- Univerzalna kvantifikacija s pomočjo dvojne negacije (pogoj in negirana množica)

\forall vrstica: P(vrstica) =

npr. „vedno rezervira rdeč čoln“

$\neg \exists$ vrstica: \neg P(vrstica) = {vse vrstice} – {vrstice: \neg P(vrstica)}

Kvantifikacija izpolnjevanja logičnega pogoja v WHERE _P_ sklopu:

„vsaj enkrat rezervira rdeč čoln“

\exists vrstica: P(vrstica)

```
SELECT DISTINCT r.jid  
FROM rezervacija r, coln c  
WHERE r.cid = c.cid  
AND c.barva = 'rdeca';
```

\forall vrstica: P(vrstica) =

„vedno rezervira rdeč čoln“

$\neg \exists$ vrstica: \neg P(vrstica) = {vse vrstice} - {vrstice: \neg P(vrstica)}

```
SELECT jid  
FROM jadralec  
-- Ali ???  
FROM rezervacija
```

-

```
SELECT r.jid  
FROM rezervacija r, coln c  
WHERE r.cid = c.cid  
AND c.barva <> 'rdeca';
```

Primer večkrat gnezdene poizvedbe

- Poišči imena jadralcev ki niso nikoli rezervirali nobenega rdečega čoln.

```
SELECT ime
FROM jadralec
WHERE jid NOT IN
  (SELECT jid
   FROM rezervacija
   WHERE cid IN
     (SELECT cid
      FROM coln
      WHERE barva='rdeca'));
```

{Vsi} - {Tisti, ki so že kdaj rezervirali kakšen rdeč čoln}

Množica rezervacij
rdečih čolnov

Množica rdečih
čolnov

Množica šifer
jadralcev, ki so že
kdaj rezervirali
katerega od
rdečih čolnov

ime
Borut
Andrej
Rajko
Zdravko
Henrik
Anze
Bine

Skupinske operacije - agregiranje podatkov



- Razširitev konceptov relacijske algebre
- Izračun po eni skupini sestavljeni iz več vrstic (npr. cela tabela)
 - Posplošitev: sočasen izračun po več skupinah
- ISO standard definira pet agregatnih operacij
 - COUNT vrne število vrednosti v določenem stolpcu
 - SUM vrne seštevek vrednosti v določenem stolpcu
 - AVG vrne povprečje vrednosti v določenem stolpcu
 - MIN vrne najmanjšo vrednost v določenem stolpcu
 - MAX vrne največjo vrednost v določenem stolpcu
- Vse operacije delujejo na enem atributu (stolpcu tabele) in vračajo eno samo vrednost, npr. AVG(starost)

Agregiranje podatkov...



- COUNT, MIN in MAX se uporabljajo za numerične in ne-numerične (primerljive) vrednosti, SUM in AVG zahtevata numerične vrednosti.
- Vse operacije razen COUNT(*) najprej odstranijo vrstice z NULL oznako v stolpcu, po katerem agregiramo.
- COUNT(*) prešteje vse vrstice, ne glede na NULL oznake ali duplikate.

Agregiranje podatkov...



- Če se želimo znebiti duplikatov, uporabimo DISTINCT pred imenom stolpca (znotraj oklepajev): `COUNT(DISTINCT EMSO)`
- DISTINCT (znotraj oklepajev) nima učinka na MIN/MAX, lahko pa vpliva na COUNT, SUM ali AVG (pri the pravzaprav nima smisla) .
- Agregatne operacije lahko uporabimo le v SELECT ali HAVING sklopu

Agregatni (skupinski) operatorji in atributi



- Skupinski atributi: veljajo za CELO skupino (npr. minimum, maksimum, število elementov, ...)
- Rezultat agregatne operacije je skupinski atribut
- Lahko nastopajo v SELECT, GROUP BY, HAVING ali ORDER BY vrstici
- V SELECT vrstici se v dani poizvedbi lahko nahajajo samo navadni ali samo skupinski atributi (ne smemo jih mešati)
- Če potrebujemo obe vrsti atributov, uporabimo gnezdene poizvedbe

Agregiranje podatkov

- Navadnih in skupinskih atributov NE SMEMO istočasno uporabljati v SELECT sklopu, razen če navadne attribute s pomočjo GROUP BY promoviramo v skupinske

~~SELECT jid, COUNT(cid)
FROM rezervacija;~~

↓
Napačna raba agregacije

Pravilna raba agregacije

↓
SELECT COUNT(cid)
FROM rezervacija;

Uporaba COUNT

- Koliko rezervacij čolnov je bilo narejenih v mesecu oktobru 2006?

```
SELECT COUNT(*) AS "St.rezervacij"  
FROM rezervacija  
WHERE dan BETWEEN DATE'2006-10-01'  
AND DATE'2006-10-31';
```

Tu je lahko karkoli:
*, jid, cid, dan

+-----+
St.rezervacij
+-----+
4
+-----+

Datumski interval. Intervalno iskanje je lahko počasno. Ali ga na tukaj zares potrebujemo?

Uporaba COUNT

- Koliko rezervacij čolnov je bilo narejenih v mesecu oktobru 2006?

```
SELECT COUNT(*)  
AS "St.rezervacij"  
FROM rezervacija  
WHERE EXTRACT(MONTH FROM dan)=10  
      AND EXTRACT(YEAR FROM dan)=2006;
```

+-----+	
St.rezervacij	
+-----+	
	4
+-----+	

Izbiranje želenih datumskih
gradnikov (DAY, MONTH, YEAR, ...)

Uporaba COUNT (nespametna alternativa)

- Koliko rezervacij čolnov je bilo narejenih v mesecu oktobru 2006?

```
SELECT COUNT(jid)
AS "St.rezervacij"
FROM rezervacija
WHERE EXTRACT(MONTH FROM dan)=10
      AND EXTRACT(YEAR FROM dan)=2006;
```

```
+-----+
| St.rezervacij |
+-----+
|                4 |
+-----+
```


Uporaba več agregatov istočasno

- Izpiši povprečno, minimalno in maksimalno starost jadralcev z ratingom 8 ali več.

```
SELECT AVG(starost), MIN(starost), MAX(starost)
FROM jadralec
WHERE rating >= 8;
```

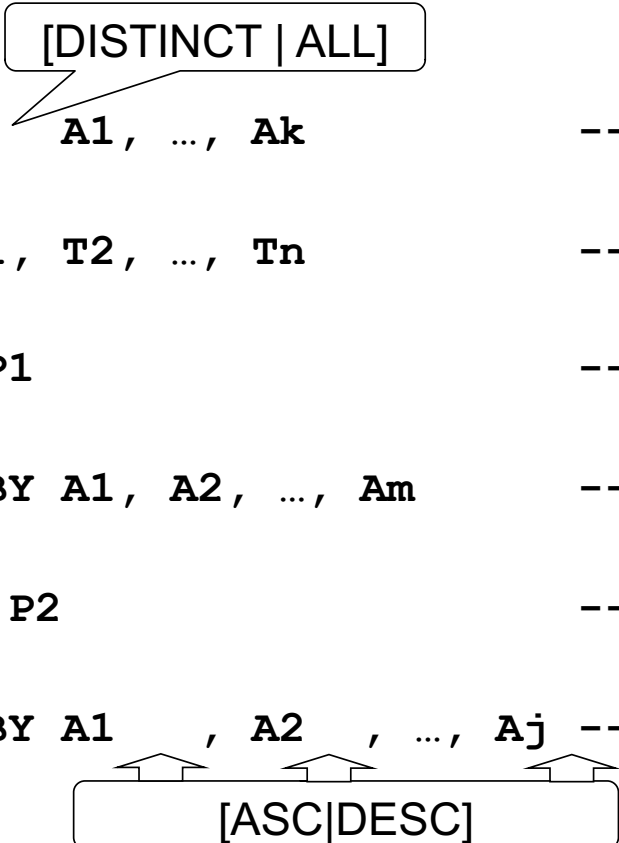

+	-----+	-----+	-----+
	AVG(starost)	MIN(starost)	MAX(starost)
+	-----+	-----+	-----+
	33.4	16	55.5
+	-----+	-----+	-----+

Združevanje (grupiranje) podatkov...



- Agregatne funkcije v osnovi delujejo na celotni množici rezultatov
- Pogosto želimo izračunati agregate na več podmnožicah celotne množice rezultatov – skupinah.
- Sklop GROUP BY uporabimo za združevanje vrstic rezultata v skupine glede na enake vrednosti navedenih atributov

Razširjeni SELECT stavek


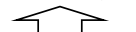
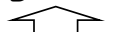

SELECT  A1, ..., Ak -- projekcija

FROM T1, T2, ..., Tn -- kartezični produkt

WHERE P1 -- selekcija po vrsticah

GROUP BY A1, A2, ..., Am -- grupiranje po atributih

HAVING P2 -- selekcija po skupinah

ORDER BY A1 , A2 , ..., Aj -- urejanje po atributih
   [ASC|DESC]

Združevanje podatkov v skupine (GROUP BY)



- Skupine nastanejo tako, da združimo vse vrstice z enako vrednostjo atributa, po katerem grupiramo.
- Atributi, po katerih grupiramo, imajo isto vrednost za celo skupino in tako postanejo skupinski atributi!
- SELECT in GROUP BY sta tesno povezana
 - vsak element v SELECT vrstici ima enako vrednost za vse elemente (vrstice) v skupini (torej je skupinski atribut)
 - SELECT sklop lahko vsebuje le:
 - imena stolpcev po katerih grupiramo
 - agregatne operacije
 - konstante ali
 - izraze, ki so sestavljeni iz kombinacije naštetih elementov.

Združevanje podatkov...



- Vsi stolpci, ki so navedeni v GROUP BY sklopu, se morajo nahajati tudi v SELECT sklopu in obratno: vsi stolpci, ki so navedeni v SELECT sklopu, se morajo nahajati tudi v GROUP BY sklopu.
 - Izjema: tisti atributi, ki nastopajo samo znotraj agregatnih operacij.
- Če uporabljamo WHERE sklop v kombinaciji z GROUP BY, se WHERE upošteva najprej, združevanje pa se izvede na preostalih vrsticah.
- ISO standard jemlje oznake NULL kot enake, ko gre za združevanje.

Primer združevanja

- Za vsak rating izpiši število jadralcev.

```
SELECT rating, COUNT(*)  
FROM jadralec  
GROUP BY rating;
```

Za vsako skupino ratingov
vrne število jadralcev

rating	COUNT(*)
1	1
3	2
7	2
8	2
9	1
10	2

Izziv: v poizvedbi nekateri ratingi manjkajo. Zakaj? Smiselno dopolnite poizvedbo, da bodo zastopani vsi!

Primer združevanja po več atributih

- Za vsak rating in starost izpiši število jadralcev.

```
SELECT rating, starost, COUNT(*)  
FROM jadralec  
GROUP BY rating, starost;
```

↓
Za vsako skupino
ratingov in starosti
vrne število jadralcev

rating	starost	COUNT(*)
1	33	1
3	25.5	1
3	63.5	1
7	35	1
7	45	1
8	25.5	1
8	55.5	1
9	35	1
10	16	1
10	35	1

Omejitev skupin



- HAVING sklop je namenjen uporabi v kombinaciji z GROUP BY kot omejitev skupin, ki se lahko pojavijo v rezultatu.
- Deluje podobno kot WHERE
 - WHERE filtrira posamezne vrstice
 - HAVING filtrira skupine.
- Stolpci, ki so navedeni v HAVING sklopu, morajo biti skupinski – torej tudi v SELECT sklopu ali v agregatih.

Uporaba sklopa HAVING

- Izpiši število čolnov vsake barve, vendar le za tiste barve, kjer imamo več kot en čoln.

```
SELECT barva, COUNT(*)  
FROM coln  
GROUP BY barva  
HAVING COUNT(*) > 1;
```

```
+-----+-----+  
| barva | COUNT(*) |  
+-----+-----+  
| rdeca |         2 |  
+-----+-----+
```

GROUP_CONCAT(...)

- Izpiši število in imena čolnov vsake barve, vendar le za tiste barve, kjer imamo več kot en čoln.

Kaj če bi bilo tukaj samo ime?
SELECT barva, COUNT(*), ime ...

```
SELECT barva, COUNT(*), GROUP_CONCAT(ime) as imena
FROM coln
GROUP BY barva
HAVING COUNT(*) > 1;
```

barva	COUNT(*)	imena
rdeca	2	Elan,Bavaria

- Funkcija združi vrednosti atributa v skupini v niz znakov.
- Uporabno predvsem pri razvoju, redkeje v produkciji.

Sledenje izvajanja grupiranja



- Za vsak rating v tabeli jadralcev izpiši starost najmlajšega polnoletnega jadralca s tem ratingom, vendar samo za tiste ratinge, ki jih imata vsaj dva polnoletna jadralca!

```
SELECT rating, MIN(starost) AS najmlajsi
FROM jadralec
WHERE starost >= 18
GROUP BY rating
HAVING COUNT(*) > 1
ORDER BY rating;
```

rating	najmlajsi
3	25.5
7	35
8	25.5

Kako deluje ta poizvedba (1)?



jid	ime	rating	starost
22	Darko	7	45
29	Borut	1	33
31	Lojze	8	55.5
32	Andrej	8	25.5
58	Rajko	10	35
64	Henrik	7	35
71	Zdravko	10	16
74	Henrik	9	35
85	Anze	3	25.5
95	Bine	3	63.5

1. korak:
vsi jadralci

Kako deluje ta poizvedba (2)?



jid	ime	rating	starost
22	Darko	7	45
29	Borut	1	33
31	Lojze	8	55.5
32	Andrej	8	25.5
58	Rajko	10	35
64	Henrik	7	35
74	Henrik	9	35
85	Anze	3	25.5
95	Bine	3	63.5

2. korak: selekcija
WHERE starost>=18

Kako deluje ta poizvedba (3)?



+-----+-----+	
rating starost	
+-----+-----+	
7 45	
1 33	
8 55.5	
8 25.5	
10 35	
7 35	
9 35	
3 25.5	
3 63.5	
+-----+-----+	

3. korak: eliminacija nepotrebnih atributov
samo atributi, ki se nahajajo v SELECT,
WHERE, GROUP BY in HAVING sklopih so
potrebni za nadaljnje delo

Kako deluje ta poizvedba (4)?



rating	starost
1	33
3	25.5
3	63.5
7	45
7	35
8	55.5
8	25.5
9	35
10	35

4. korak: grupiranje po vrednosti atributa rating

Kako deluje ta poizvedba (5)?



+-----+-----+	
rating starost	
+-----+-----+	
3 25.5	
3 63.5	
+-----+-----+	
7 45	
7 35	
+-----+-----+	
8 55.5	
8 25.5	
+-----+-----+	

5. korak: eliminacija odvečnih skupin.
Ohranimo samo tiste, za katere velja
HAVING COUNT(*) > 1

Kako deluje ta poizvedba (6)?



```
+-----+-----+
| rating | starost |
+-----+-----+
|      3 |    25.5 |
+-----+-----+
|      7 |     35 |
+-----+-----+
|      8 |    25.5 |
+-----+-----+
```

6. korak: izvajanje skupinskega operatorja
(v naše primeru MIN) na vsaki
posamezni skupini

Gnezdene poizvedbe v FROM sklopu



- Rezultat poizvedbe je začasna tabela
- Zakaj ga ne bi mogli uporabiti v **FROM** vrstici kot vsebino začasne tabele?

SELECT *

FROM (poizvedba1) AS sinonim1,
 (poizvedba2) AS sinonim2,
WHERE sinonim1.x = sinonim2.y;

- Obvezna uporaba sinonimov!

Pogosti tabelarični izrazi

- CTE (Common Table Expressions), ISO SQL:1999

- Oracle, Microsoft, PostgreSQL podpirajo
- MariaDB (nad ver 10.2), MySQL (nad ver 8) podpirata
- olajšajo pisanje DDL in tudi DML stavkov
- nadomešča začasne tabele

poizvedba1 je CTE1
poizvedba2 je CTE2
...

WITH sinonim1 AS (poizvedba1),
 sinonim2 AS (poizvedba2), ...

SELECT *

FROM sinonim1 JOIN sinonim2 ON (...)

WHERE ... -- poizvedba, ki (večkrat) uporabi sinonime

Pogosti tabelarični izrazi



- CTE ponuja različne možnosti za optimizacijo
 - Uporaba rezultata CTE kot začasne tabele
 - Kombiniranje definicije CTE in poizvedbe (kot pri uporabi pogledov – VIEW)
- Rezultat neodvisen od izvedbe
- Izvedba močno odvisna od implementacije optimizatorja poizvedb SQL

Primerjava SQL in relacijske algebre



- Tabele niso vedno relacije: mogoče podvajanje
- Pogledi in tabele pogosto niso enako obravnavani (čeprav bi bili lahko, npr. pri vnosu vrstic)
- SQL ne preverja minimalnosti ključev (primarni ključ je lahko poljuben nadključ)
- Trivrednostna logika (T, F, NULL)
- Skupinske operacije kot razširitev rel. algebre

Domača naloga (brez oddaje)



- Oglejte si in preizkusite datumske funkcije ki jih ponuja MariaDB ali MySQL (npr. DATE, DATEDIFF, CURDATE, EXTRACT, ...)
 - <https://mariadb.com/kb/en/date-time-functions/>
- Oglejte si celotno knjižico vgrajenih funkcij MariaDB ali MySQL, npr. znakovne, datumske, aritmetične, agregatne, nadzor toka, de(kodiranje), enkripcija, zgoščevanje (hash), JSON, ...
 - <https://mariadb.com/kb/en/built-in-functions/>
- Od vas pričakujem, da vgrajene funkcije poznate in jih boste znali uporabiti na izpitu!