

8. Týždeň

Vnorené dopyty. CASE výraz. ROLLUP a CUBE. Extrémy s ALL.

1) Typy výsledkov dopytu

2) Vnorené dopyty (VD) <http://dev.mysql.com/doc/refman/5.0/en/subqueries.html>

- a) Pravidlá zápisu VD
- b) Miesta VD v **SELECT**e s príkladmi
- c) Typy VD
- d) Príklady – Poliklinika

3) CASE výraz <http://dev.mysql.com/doc/refman/5.5/en/case.html>

4) GROUP BY s ROLLUP

5) LIMIT

6) Extrémy a vnorené dopyty

- a) Extrémy iba s ALL
- b) Prechodné tabuľky a premenné

1) Typy výsledkov dopytu

- tabuľka (viac riadkov, viac stĺpcov)
- jeden stĺpec/zoznam
- jediná/skalárna hodnota
- nič

SELECT môže byť všade, kde je

- skalárna hodnota
- **zoznam**
- tabuľka

Napr. kde môže byť **skalárna hodnota** či **zoznam**:

```
SELECT select_zoznam
FROM from_zoznam
WHERE ... sk.hodnota ... IN (in_zoznam)
...
```

2) Vnorené dopyty

Vnorený dopyt je dopyt (SELECT), ktorý je vnorený do (vonkajších) príkazov
SELECT, INSERT, UPDATE, DELETE
alebo ďalšieho vnoreného dopytu.

- Vonkajší dopyt
Outer query
- Vnorený dopyt ⇔ Vnútorý dopyt ⇔ Poddopyt
Subquery, Inner query

Vnorený dopyt sa snáď najčastejšie vyskytuje vo WHERE ... IN klauzule vonkajšieho SELECTu, ale nie len tam, veď SELECT môže byť všade, kde je zoznam (pozri vyššie) a ak napr. VD vracia skalárnu hodnotu, potom môže byť všade, kde môže byť výraz.

Väčšina príkazov s vnorenými dopytmi môže byť sformulovaný ako JOIN. Ktoré riešenie je efektívnejšie, závisí od takých faktorov, ako je napr. veľkosť tabuliek, prítomnosť indexov. Podľa dokumentácie riešenie vnorených dopytov pomocou JOIN je štandardne efektívnejšie: *“The optimizer is more mature for joins than for subqueries, so in many cases a statement that uses a subquery can be executed more efficiently if you rewrite it as a join.”*. <http://dev.mysql.com/doc/refman/5.7/en/subquery-restrictions.html>

Rozdiel medzi VD a JOIN:

- vnorený dopyt je úplný SELECT príkaz, JOIN je iba súčasťou SELECT príkazu
- vnorený dopyt sa môže nachádzať na rôznych miestach SELECTu, JOIN sa nachádza za FROM.

a) Pravidlá zápisu vnorených dopytov

- VD sa vždy píše do zátvoriek
SELECT ... (SELECT ...) ...
- ak sa nachádza vo from_zozname, potom treba vypísať i alias (pomocou AS alebo bez neho):

```
SELECT * FROM (SELECT * FROM Lekari L1 ) v1,  
              (SELECT * FROM Lekari L2 ) AS v2
```

- maximalný počet vnorených dopytov je zhora ohraničený (32 SQL SERVER, MySQL ?)

Poznámka: síce tu ešte nevidíme význam aliasov L1, L2, ale ich použitie sa odporúča aby zápis bol prehľadnejší.

b) Miesta vnorených dopytov v SELECTe s príkladmi

Napr.

- hneď za SELECT v select_zozname
- + v in_zozname v rámci WHERE;
- + za FROM vo from_zozname
- príklad, kde SELECT vystupuje ako skalárna hod. vo WHERE klauzule, ukážeme nižšie pri Extrémoch.

```
SELECT 111 FROM Lekari; # Co bude vysledkom?  
-- v select_zozname:  
SELECT (SELECT krstne FROM Lekari WHERE idL=1)  
       FROM (SELECT * FROM Lekari WHERE idL < 4 ) AS s1;  
  
-- + v in_zozname:  
SELECT * FROM Lekari L  
       WHERE L.idL*2 IN( SELECT idL FROM Lekari);  
  
- + vo from_zozname:  
SELECT (SELECT krstne FROM Lekari WHERE idL=1)  
       FROM (SELECT * FROM Lekari WHERE idL  
             IN (SELECT idL FROM Lekari WHERE idL <4)  
             ) AS s1;
```

- WHERE vyraz IN (VD) ⇔ WHERE vyraz = ANY (VD)
- WHERE vyraz NOT IN (VD) ⇔ WHERE vyraz <> ALL (VD)
- WHERE [NOT] EXISTS (VD)
- GROUP BY, HAVING ...

c) Typy vnorených dopytov

- nezávislý VD / VD bez parametra
- korelovaný VD / VD s parametrom (correlated subquery http://en.wikipedia.org/wiki/Correlated_subquery)
 - postup vykonania!!!
 - počet navštívených riadkov
 - prepojenie tabuliek, kľúče
- self query

d) Príklady

V nasledujúcich príkladoch sa opierame o databázu **Poliklinika**:

Pacienti			Navstevy				
idP	krstne	mesPrijem	idN	idP	idL	den	poplatok
1	Adam	10000	1	1	2	2008-05-05 00:00:00.000	NULL
2	Stefan	9500	2	2	3	2008-05-05 00:00:00.000	NULL
3	Slavo	8500	3	6	3	2008-05-05 00:00:00.000	NULL
4	Klara	9000	4	4	1	2008-06-05 00:00:00.000	200
5	Zuzana	35000	5	5	4	2008-06-05 00:00:00.000	500
6	Tana	20000	6	7	1	2008-06-05 00:00:00.000	200
7	Mato	28000	7	6	1	2008-06-05 00:00:00.000	500
8	Zoli	32000	8	8	3	2008-07-05 00:00:00.000	900
9	Misko	NULL	9	2	1	2008-07-05 00:00:00.000	200
10	Janka	NULL	10	3	3	2008-07-05 00:00:00.000	100
			11	6	2	2008-08-05 00:00:00.000	700
			12	7	2	2008-08-05 00:00:00.000	500
			13	6	4	2008-08-05 00:00:00.000	800
			14	2	1	2008-09-05 00:00:00.000	NULL
			15	3	1	2008-09-05 00:00:00.000	200
			16	8	1	2008-09-05 00:00:00.000	200
			17	9	5	2008-09-05 00:00:00.000	NULL
			18	7	1	2008-10-05 00:00:00.000	300
			19	8	4	2008-10-05 00:00:00.000	800
			20	10	5	2008-10-05 00:00:00.000	300
			21	1	1	2008-11-05 00:00:00.000	350
			22	6	5	2008-11-05 00:00:00.000	400

Lekari			
idL	krstne	spec	datNar
1	Oto	Ocny	1960-05-05 00:00:00.000
2	Zoli	Zubny	1961-11-14 00:00:00.000
3	Klara	Kardiolog	1980-02-15 00:00:00.000
4	Zuzka	Zubny	1970-04-02 00:00:00.000
5	Imro	Interny	1956-11-09 00:00:00.000

1) Zistite **id** pacientov, ktorí už boli u kardiológa, teda u lekára s idL 3.

```
SELECT idP FROM Navstevy
      WHERE idL = 3;
```

... u všetkých kardiológov => JOIN:

```
SELECT idP FROM Navstevy N JOIN Lekari L ON N.idL = L.idL
      WHERE L.spec="Kardiolog";
```

Kontrolujte úlohu pre zubného/ých lekára/ov.

Pokračovanie:

2) Zistite **id** a **mena pacientov**, ktorí už boli u kardiologa s idL 3.

---- 3 riesenia:

---- 3. riesenie je bez NULL, NULL

---- 1. Riesenie: VnoDop a nie JOIN (su tu nevyhnutne aliasy P a L?)
 ---- idP JE V ZOZNAME/VD <=> ROVNA SA PRVKU ZOZNAMU/VD:

```
SELECT P.idP, P.krstne
FROM Pacienti P
WHERE P.idP IN( SELECT N.idP -- zoznam pac. u 3
                FROM Navstevy N
                WHERE N.idL = 3 );
```

idP	krstne
2	Stefan
3	Slavo
6	Tana
8	Zoli
NULL	NULL

---- - nezávisly VD
 ---- - Su nutne P a N?
 ---- - Moze byt: SELECT N.idP ...

---- <=>:

---- 2. riesenie:

---- idP SA ROVNA NIEKTOREMU PRVKU /EXISTUJE TAKY PRVOK VO/ VD:

```
SELECT P.idP, P.krstne
FROM Pacienti P
WHERE P.idP = Any( SELECT N.idP
                  FROM Navstevy N
                  WHERE N.idL = 3 );
```

---- <=>:

---- 3. riesenie pomocou JOIN (na základe nasl. príkladu, lahko napíšete ďalšie verzie dopytu):

```
SELECT P.idP, P.krstne FROM Pacienti P
                        JOIN Navstevy N On P.idP = N.idP
WHERE idL = 3;
```

3) Zistite id a mená pacientov, ktorí sú aj lekármi (pacient a lekár s tým istým menom).

---- 6 rieseni:

---- - 1, 2) nezávisle VD - IN, ANY
 ---- - 3) korelovany VD
 ---- - 4) zoznam tabuliek
 ---- - 5, 6) CROSS a INNER JOIN
 ---- 5, 6 su bez NULL, NULL

---- 1): IN

```
SELECT idP, krstne
FROM Pacienti
WHERE krstne IN(SELECT krstne
                FROM Lekari);
```

idP	krstne
4	Klara
8	Zoli
NULL	NULL

-- <=>

```
SELECT P.idP, P.krstne
FROM Pacienti P
WHERE P.krstne IN( SELECT L.krstne
                  FROM Lekari L );
```

---- <=> 2): ANY

```
SELECT P.idP, P.krstne
FROM Pacienti P
WHERE P.krstne = ANY( SELECT L.krstne
                     FROM Lekari L );
```

```

---- <=> 3) Korelovany dopyt (tu aliasy L, N su potrebne)
----      EXISTS / Existuje taky riadok vo VD ...:
SELECT P.idP, P.krstne
      FROM Pacienti P
      WHERE EXISTS ( SELECT * FROM Lekari L
                    WHERE L.krstne = P.Krstne );

```

```

---- <=> 4): Kart.Sucin
SELECT P.idP, P.krstne
      FROM Pacienti P, Lekari L # zoznam tabuliek
      WHERE L.krstne = P.Krstne;

```

```

---- <=> 5): CROSS JOIN
SELECT P.idP, P.krstne
      FROM Pacienti P CROSS JOIN Lekari L
      WHERE L.krstne = P.Krstne;

```

idP	krstne
4	Klara
8	Zoli

```

---- <=> 7): INNER JOIN
SELECT P.idP, P.krstne
      FROM Pacienti P JOIN Lekari L
      On L.krstne = P.Krstne;

```

Reštrikcie <https://dev.mysql.com/doc/refman/5.0/en/subquery-restrictions.html>

3) CASE výraz

CASE vyhodnotí zoznam podmienok a vráti jeden z viacerých možných výrazov.

CASE môžeme použiť dvojako:

a) CASE vyraz WHEN c1 THEN ... - jednoduchý CASE vyraz

b) CASE WHEN vyraz=c1 THEN ... - vyhľadávaný CASE vyraz

- v príkazoch SELECT, UPDATE, DELETE, SET a

- v klauzulách select_zoznam, IN, WHERE, ORDER BY a HAVING

Syntax: <http://dev.mysql.com/doc/refman/5.0/en/case.html>

```

CASE case_value
  WHEN when_value THEN statement_list
  [WHEN when_value THEN statement_list] ...
  [ELSE statement_list]
END CASE

```

```

CASE
  WHEN search_condition THEN statement_list
  [WHEN search_condition THEN statement_list] ...
  [ELSE statement_list]
END CASE

```

V jednoduchom CASE výraze používame iba rovnosti. CASE môže vrátiť aj NULL.

Príklady

1) Vráťte dátum_narodenia, rok a jeho párnosť študenta v DB Skola (resp. osoba_vztah s tab. Osoba a so stĺpcom dat_nar)!

```
USE Skola; # vyhľadávány CASE
SELECT datum_narodenia, YEAR(datum_narodenia) rok
      , CASE
          WHEN YEAR(datum_narodenia)%2 = 0 THEN 'Parny'
          WHEN YEAR(datum_narodenia)%2 = 1 THEN 'Neparny'
          ELSE '?'
        END as 'parnost'
FROM ziak;
```

Ekvivalentné riešenie - jednoduchý CASE:

```
SELECT datum_narodenia, YEAR(datum_narodenia) rok
      , CASE YEAR(datum_narodenia)%2
          WHEN 0 THEN 'Parny'
          WHEN 1 THEN 'Neparny'
          ELSE '?'
        END as 'parnost'
FROM ziak;
```

datum_narodenia	rok	parnost
1987-07-12 00:00:00	1987	Neparny
1984-02-01 00:00:00	1984	Parny
1980-01-22 00:00:00	1980	Parny
1984-03-03 00:00:00	1984	Parny
1982-04-14 00:00:00	1982	Parny
1979-07-16 00:00:00	1979	Neparny
1977-09-21 00:00:00	1977	Neparny
1977-09-21 00:00:00	1977	Neparny
NULL	NULL	?
1987-12-22 00:00:00	1987	Neparny
1983-06-06 00:00:00	1983	Neparny
1982-10-07 00:00:00	1982	Parny
1981-09-23 00:00:00	1981	Neparny

2) Vráťte datum_narodenia študenta, mesiac a kvartál!

Najprv ukážeme riešenia pomocou IN a BETWEEN:

```
SELECT datum_narodenia, MONTH(datum_narodenia) mes
      , CASE
          WHEN Month(datum_narodenia) IN(1,2,3) THEN 1
          WHEN Month(datum_narodenia) IN (4,5,6) THEN 2
          WHEN Month(datum_narodenia) IN (7,8,9) THEN 3
          WHEN Month(datum_narodenia) IN (10,11,12) THEN 4
          # ELSE datum_narodenia
        END as 'kvart'
FROM ziak;
```

alebo

```
SELECT datum_narodenia, MONTH(datum_narodenia) mes
      , CASE
          WHEN Month(datum_narodenia) BETWEEN 1 and 3 THEN 1
          WHEN Month(datum_narodenia) BETWEEN 4 and 6 THEN 2
          WHEN Month(datum_narodenia) in(7,8,9) THEN 3
          WHEN Month(datum_narodenia) in(10,11,12) THEN 4
          # ELSE datum_narodenia
        END as 'kvart'
FROM ziak;
```

Pomocou celočíselného delenia DIV obdržime kratšie riešenie:

```
SELECT datum_narodenia, MONTH(datum_narodenia) mes,
      ((MONTH(datum_narodenia)-1) DIV 3)+1 kvart
from ziak;
```

datum_narodenia	mes	kvart
1987-07-12 00:00:00	7	3
1984-02-01 00:00:00	2	1
1980-01-22 00:00:00	1	1
1984-03-03 00:00:00	3	1
1982-04-14 00:00:00	4	2
1979-07-16 00:00:00	7	3
1977-09-21 00:00:00	9	3
1977-09-21 00:00:00	9	3
NULL	NULL	NULL
1987-12-22 00:00:00	12	4
1983-06-06 00:00:00	6	2
1982-10-07 00:00:00	10	4
1981-09-23 00:00:00	9	3

4) GROUP BY s ROLLUP

ROLLUP a **CUBE** sa využívajú pri vypočítavaní sumárnych veličín.

- ROLLUP generuje agregačné hodnoty pre **hierarchické** hodnoty vo vybraných stĺpcoch
- CUBE generuje agregačné hodnoty pre všetky kombinácie hodnôt vo vybraných stĺpcoch (MySQL zatiaľ nepodporuje).

4a) Zistite sumárne mesačné poplatky pre jednotlivé poplatky – teda sumárny poplatok **pre každý mesiac** aj **pre každý poplatok**.

(vieme, že v select zozname vymenované neagregačné stĺpce musia byť vymenované aj v GROUP BY klauzule)

USE Poliklinika;

```
SELECT month(den) mes, poplatok, SUM(poplatok) suma
FROM Navstevy
GROUP BY month(den), poplatok
ORDER BY suma;
```

	mes	poplatok	suma
1	5	NULL	NULL
2	9	NULL	NULL
3	7	100	100
4	7	200	200
5	11	350	350
6	11	400	400
7	9	200	400
8	6	200	400
9	8	500	500
10	10	300	600
11	8	700	700
12	8	800	800
13	10	800	800
14	7	900	900
15	6	500	1000

4b) Zistite sumárne poplatky **pre každý mesiac v druhom polroku** (pre jednotlivé poplatky nie).

```
SELECT month(den) mes, SUM(poplatok) suma
FROM Navstevy
-- WHERE month(den) >= 7
-- WHERE mes >= 7 -- not
GROUP BY month(den)
-- GROUP BY mes -- not SS - SQL Server
-- HAVING month(den) >= 7 -- SS, not MySQL
HAVING mes >= 7 -- not SS
-- ORDER BY suma;
-- ORDER BY SUM(poplatok);
ORDER BY mes;
-- ORDER BY month(den);
```

	mes	suma
1	9	400
2	11	750
3	7	1200
4	10	1400
5	8	2000

5a) Zistite sumárne poplatky **pre každý poplatok** z daných troch poplatkov {200, 500, 800}.

-- Sumárne poplatky pre 200,500 a 800

```
SELECT N.poplatok, sum(N.Poplatok) suma
FROM Navstevy N
Where N.Poplatok IN(200,500,800)
Group by poplatok;
```

	poplatok	suma
1	200	1000
2	500	1500
3	800	1600

5b) Zistite sumárne poplatky **pre každý poplatok** z daných troch poplatkov {200, 500, 800} a **pre každého špecialistu**.

-- Sumárne poplatky pre 200,500 a 800 u jednotliv. špecialistov

```
SELECT L.Spec, N.poplatok, sum(N.Poplatok) suma
FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
Where N.Poplatok IN(200,500,800)
Group by spec, poplatok;
```

	Spec	poplatok	suma
1	Ocny	200	1000
2	Ocny	500	500
3	Zubny	500	1000
4	Zubny	800	1600

```
-- Sumárne poplatky aj podľa jednotliv. špecialistov
SELECT L.Spec, N.poplatok, sum(N.Poplatok) suma
  FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
  Where N.Poplatok IN(200,500,800)
  Group by spec, poplatok
With Rollup;
```

	Spec	poplatok	suma
1	Ocny	200	1000
2	Ocny	500	500
3	Ocny	NULL	1500
4	Zubny	500	1000
5	Zubny	800	1600
6	Zubny	NULL	2600
7	NULL	NULL	4100

Prepíšme NULLy na ZVsetci a SumPop.

```
SELECT CASE WHEN L.Spec IS NULL THEN 'ZVsetci'
  Else L.Spec End Spec,
  CASE WHEN N.poplatok IS NULL THEN 'SumPop'
  Else cast(N.poplatok as char(10)) End Popl,
  -- Else cast(N.poplatok as Varchar(10)) End Popl,-- SS
  sum(N.Poplatok) Suma
  FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
  Where N.Poplatok IN(200,500,800)
  Group by spec, poplatok
  With Rollup;
```

	Spec	Popl	Suma
1	Ocny	200	1000
2	Ocny	500	500
3	Ocny	SumPop	1500
4	Zubny	500	1000
5	Zubny	800	1600
6	Zubny	SumPop	2600
7	ZVsetci	SumPop	4100

```
-- Sumárne poplatky aj podľa jedn. spec. aj podľa poplatkov
### Cube - riesenie iba v MS SQL SERVER
SELECT CASE WHEN L.Spec IS NULL THEN 'ZVsetci'
  Else L.Spec End Spec,
  CASE WHEN N.poplatok IS NULL THEN 'SumPop'
  Else cast(N.poplatok as char(10)) End Popl,
  sum(N.Poplatok) Suma
  FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
  Where N.Poplatok IN(200,500,800)
  Group by spec, poplatok
  With Cube Order By Spec, Poplatok Desc;
```

	Spec	Popl	Suma
1	Ocny	500	500
2	Ocny	200	1000
3	Ocny	SumPop	1500
4	Zubny	800	1600
5	Zubny	500	1000
6	Zubny	SumPop	2600
7	ZVsetci	800	1600
8	ZVsetci	500	1500
9	ZVsetci	200	1000
10	ZVsetci	SumPop	4100

Kým štandardný GROUP BY poskytuje jednorozmerné sumárne hodnoty jednej premennej, **kontingenčné tabuľky (pivot tabuľky)** ponúkajú dvojrozmerný pohľad na sumárne veličiny dvoch premenných (pozri neskoršie prednášky).

5) Limit LIMIT m[, n]

LIMIT n vráti iba prvých n riadkov

LIMIT m, n – po vynechaní m riadkov vráti prvých n riadkov

```
SELECT * FROM TT LIMIT 3, 2; -- vrati 4. a 5. riadok tabuľky TT
```

LIMIT sa často používa spolu s ORDER BY.

6) Extrémy a vnorené dopyty

```
- MAX, MIN
- ALL
- ... [ORDER BY] ... LIMIT m[, n]
```

Budeme hľadať najmladších lekárov.

```
USE Poliklinika;
SELECT * FROM Lekari;
```

0) Usporiadajte lekárov podľa veku zostupne:

```
SELECT datNar FROM Lekari
  ORDER BY datNar DESC; -- 5r: 1980//1970//1961//1960//1956
```

idL	krstne	spec	datNar
1	Oto	Ocny	1960-05-05 00:00:00
2	Zoli	Zubny	1961-11-14 00:00:00
3	Klara	Kardiolog	1980-02-15 00:00:00
4	Zuzka	Zubny	1970-04-02 00:00:00
5	Imro	Interny	1956-11-09 00:00:00
NULL	NULL	NULL	NULL

1a) Najdite datum narodenia najmladsieho/ej lekara/ky (maximalny datum narodenia):
SELECT MAX(datNar) najmladsi FROM Lekari;

1b) Vypiste aj jeho/jej krstne meno a specializaciu:
Nasledujúci dopyt

```
SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari;
```

je problematický, nerieši úlohu.

- V štandard SQL sa hlási error, lebo napr. krstné by mal byť BUĎ v AGR.FUNK. ALEBO V GROUP BY.

- V MySQL môže prejsť, ale vráti nesprávny výsledok - Pozor Oto!? See [MySQL Handling of GROUP BY](#)

```
SET sql_mode = ''; # Not SQL Standard - default nastavenie!?  
SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari; ## NO -  
chybny vysledok
```

```
SET sql_mode = 'ONLY_FULL_GROUP_BY'; # SQL Standard  
SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari; ## OK varuje
```

-- NIE - 5 riadkov:

```
# SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari GROUP BY  
krstne, spec;
```

OK 1b) (najdite datum narodenia, krstne a specializaciu najmladsieho/ej lekara/ky)

```
SELECT krstne, spec, L2.datNar FROM Lekari L2  
WHERE L2.datNar =  
( SELECT MAX(L1.datNar) FROM Lekari L1 );
```

krstne	spec	datNar
Klara	Kardiolog	1980-02-15 00:00:00

1c) Riesme ulohu bez pouzitia MAX:

```
SELECT krstne, spec, L2.datNar FROM Lekari L2  
WHERE L2.datNar >= ALL -- podmienka pre KAZDU/ALL DVOJICU!  
( SELECT L1.datNar FROM Lekari L1 );
```

2a) Najdite datum narodenia druhého najmladsieho/u lekara/ky – použitie Max dovolené:

```
SELECT MAX(L2.datNar) 'Druhy najmladsi' FROM Lekari L2  
WHERE L2.datNar <  
( SELECT MAX(L1.datNar) FROM Lekari L1 );
```

2b) Najdite udaje o druhom/ej najmladsom/ej lekarovi/ke – použitie Max dovolené:

```
SELECT krstne, spec, L2.datNar FROM Lekari L2  
WHERE L2.datNar=(  
SELECT MAX(L2.datNar) 'Druhy najmladsi' FROM Lekari L2  
WHERE L2.datNar <  
( SELECT MAX(L1.datNar) FROM Lekari L1 )  
);
```

3) Najdite udaje o tretom/tej najmladsom/ej lekarovi/ke – použitie Max dovolené:

```
select krstne, datnar najmlad from lekari  
where datnar=(  
select MAX(datnar) najml3 from lekari where datnar <  
(select MAX(datnar) najml2 from lekari where datnar <  
(select MAX(datnar) najml1 from lekari))  
);
```

a) Extrémy iba s ALL

Zistite **druhý** najmenší príjem pomocou A) MIN a B) ALL.

Úloha minimálneho príjmu a) s MIN a b) s ALL je ľahký, ako aj 2. najmenšieho príjmu s MIN, pozri 11), 12) a 21).

```
use poliklinika;
select mesPrijem from Pacienti order by mesPrijem;
```

11) 1. MIN pomocou a) MIN

```
select MIN(mesPrijem) from Pacienti;
```

<=>

12) b) Riadky, v ktorých prij <= prij v ALL

```
SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem <= ALL
( SELECT P1.mesPrijem FROM Pacienti P1
WHERE mesPrijem is not NULL);
```

	mesPrijem
▶	8500

A) 21) 2. MIN <=> pomocou a) MIN > MIN

```
SELECT MIN(mesPrijem) FROM Pacienti P2
WHERE P2.mesPrijem >
( SELECT MIN(P1.mesPrijem) FROM Pacienti P1 );
```

	MIN(mesPrijem)
▶	9000

B) Algoritmus s All

Pre 2. najmenší príjem s ALL popíšeme algoritmické kroky 22) 23) 24):

22) V 21) vynechaj prve slovo MIN <=> riadky, v ktorých prij > min

23) V 22) nahraď SELECT s druhým MIN s 12)

24) b) Posledný dopyt 23) je dvojnásobným vstupom pre FROM v dopyte 12), teda zdrojom pre dopyt 12) nie je pôvodná tabuľka Pacienti, ale jej oklieštená verzia, z ktorej už chýbjú pacienti s najmenším príjmom.

22) V 21) vynechaj prve slovo MIN <=> riadky, v ktorých prij > min

```
SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem >
( SELECT MIN(P1.mesPrijem) FROM Pacienti P1 );
```

23) V 22) nahraď vnorený druhý SELECT s 12)

```
SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem >
( SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem <= ALL
( SELECT P1.mesPrijem FROM Pacienti P1
WHERE mesPrijem is not NULL));
```

	mesPrijem
▶	10000
	9500
	9000
	35000
	20000
	28000
	32000

24) b) Posledný dopyt 23) je dvojnásobným vstupom pre FROM v dopyte 12), teda zdrojom pre dopyt 12) nie je pôvodná tabuľka Pacienti, ale jej oklieštená verzia, z ktorej už chýbjú pacienti s najmenším príjmom.

```
SELECT mesPrijem FROM (SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem >
( SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem <= ALL
( SELECT P1.mesPrijem FROM Pacienti P1
WHERE mesPrijem is not NULL) )) P2
WHERE P2.mesPrijem <= ALL
```

```
( SELECT P1.mesPrijem FROM (SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem >
( SELECT mesPrijem FROM Pacienti P2
WHERE P2.mesPrijem <= ALL
( SELECT P1.mesPrijem FROM Pacienti P1
WHERE mesPrijem is not NULL)
)) P1 WHERE mesPrijem is not NULL);
```

MIN(mesPrijem)
9000

C1) Nájdite krstné meno pacienta s druhým najmenším mesačným príjmom s použitím ALL (bez použitia MIN, MAX) a vypíšte aj mesPrijem.

a) Všetky mesačné prijmy – mesPrijem:

```
use Poliklinika;
SELECT mesPrijem FROM Pacienti order by mesPrijem;
```

mesPrijem
NULL
NULL
8500
9000
9500
10000
20000
28000
32000
35000

b) Dva najmenšie mesačné prijmy:

```
SELECT p4.mesPrijem FROM Pacienti p4 -- 8500, 9000
WHERE p4.mesPrijem <= ALL
( SELECT p3.mesPrijem FROM Pacienti p3 -- 7 riadkov
WHERE p3.mesPrijem >
( SELECT p2.mesPrijem FROM Pacienti p2 -- 8500
WHERE p2.mesPrijem <= ALL
( SELECT p1.mesPrijem FROM Pacienti p1 -- 8 riadkov
WHERE p1.mesPrijem IS NOT NULL
)
)
);
```

mesPrijem
8500
9000

c) Meno lekára s druhým najmenším mesačným príjmom:

```
SELECT T.krstne, T.mesPrijem FROM -- Klara, 9000
( SELECT p3.krstne, p3.mesPrijem FROM Pacienti p3
WHERE p3.mesPrijem >
( SELECT p2.mesPrijem FROM Pacienti p2 -- min
WHERE p2.mesPrijem <= ALL
( SELECT p1.mesPrijem FROM Pacienti p1
WHERE p1.mesPrijem IS NOT NULL
)
)
) T
WHERE T.mesPrijem <= ALL
( SELECT p3.mesPrijem FROM Pacienti p3 -- 7
WHERE p3.mesPrijem >
( SELECT p2.mesPrijem FROM Pacienti p2 -- 1
WHERE p2.mesPrijem <= ALL
( SELECT p1.mesPrijem FROM Pacienti p1 -- 8
WHERE p1.mesPrijem IS NOT NULL
)
)
);
```

krstne	mesPrijem
Klara	9000

Vnorený dopyt **T** sa líši od dolného, druhého VD iba s p3.krstne. Nižšie ukážeme že použitie prechodných tabuiek sprehľadní riešenie.

b) Prechodné tabuľky a premenné + C2) riešenie

Vytvorenie prechodnej tabuľky (temporary table for session) PrTa.

```
USE sys;
DROP DATABASE IF EXISTS dbmaz;
CREATE DATABASE dbmaz;
```

```
USE dbmaz;
DROP TABLE IF EXISTS T;
CREATE TABLE T (x int, y int);
INSERT T VALUES (1,11), (2,12), (3,13);
```

```
DROP TABLE IF EXISTS PrTa;
CREATE TEMPORARY TABLE PrTa (SELECT x, y FROM T WHERE x<=2);
SELECT * from PrTa;
```

Vytvorenie premennej @x.

```
SET @k := 11;
SELECT @k;
```

C2) Nájdite krstné meno pacienta s druhým najmenším mesačným príjmom s použitím ALL (bez použitia MIN, MAX) a vypíšte aj mesPrijem – pomocou **prechodnej tabuľky** a premennej @x: ktorá bude obsahovať najmenší mesačný príjem.

```
USE poliklinika;
DROP TABLE IF EXISTS JAJ;
CREATE TEMPORARY TABLE JAJ (
    SELECT p2.mesPrijem FROM Pacienti p2 -- 1
    WHERE p2.mesPrijem <= ALL
        ( SELECT p1.mesPrijem FROM Pacienti p1 -- 8
          WHERE p1.mesPrijem IS NOT NULL
        )
);
# SELECT * from JAJ; # 8500
SET @x := (SELECT * from JAJ);
# SELECT @x; # 8500
SELECT p3.krstne, p3.mesPrijem FROM Pacienti p3
WHERE p3.mesPrijem > @x;
```

krstne	mesPrijem
Adam	10000
Stefan	9500
Klara	9000
Zuzana	35000
Tana	20000
Mato	28000
Zoli	32000

```
SELECT T.krstne, T.mesPrijem FROM -- Klara, 9000
    (SELECT p3.krstne, p3.mesPrijem FROM Pacienti p3
     WHERE p3.mesPrijem > @x
    ) T
WHERE T.mesPrijem <= ALL
    (SELECT p3.mesPrijem FROM Pacienti p3
     WHERE p3.mesPrijem > @x
    );
```

krstne	mesPrijem
Klara	9000

d) Vyskúšajte, či to pôjde, namiesto @x písať (SELECT * from JAJ).