

Jazyk SQL SELECT

Michal Valenta

Katedra softwarového inženýrství FIT
České vysoké učení technické v Praze
©Michal Valenta, 2014

BI-DBS, ZS 2014/15

<https://edux.fit.cvut.cz/courses/BI-DBS/>



Agregační funkce

D9. Kolik je filmů natočených v letech 1938-1940?

```
SELECT COUNT(*) AS pocet_filmu_38_40  
FROM Filmy  
WHERE rok BETWEEN 1938 AND 1940;
```

D10. Kolik různých filmů je rezervovaných?

```
SELECT COUNT (DISTINCT jméno_f)  
FROM Rezervace;
```

D11. Jaká je průměrná cena výpůjčky?

```
SELECT AVG(cena)  
FROM Vypujcky;  
nezahrnuje výpůjčky bez ceny  
(s cenou NULL)
```

```
SELECT AVG(COALESCE (cena,0))  
FROM Vypujcky;  
výpůjčky s cenou NULL se přeloží  
jako 0 a započtou se do výsledku
```

Agregační funkce

Syntaxe:

agregační_funkce ({ALL | DISTINCT} sloupec | výraz)

- COUNT, SUM, MAX, MIN, AVG a mnoho dalších
- Výpočet napříč skupinou zdrojových řádků.
- Co s NULL hodnotami ve sloupci?
- Co s duplicitními hodnotami ve sloupci?
- COUNT(\emptyset) = 0

Výjimka

COUNT(A) ... ignoruje NULL

COUNT(*) ... započte NULL

Agregační funkce

D12. Najdi počet výpůjček s cenou výpůjčky do 899 Kč.

```
SELECT COUNT(*)  
FROM Vypujcky  
WHERE cena < 899.00;
```

D13. Zjisti pro zahraniční zaměstnance celkový objem jejich platů přepočtený na EUR.

```
SELECT SUM (plat)/24.65 AS euro_plat  
FROM Zamestnanci  
WHERE rod_c IS NULL;
```

nebo:

```
SELECT SUM(plat/24.65) AS euro_plat  
FROM Zamestnanci  
WHERE rod_c IS NULL;
```

První varianta je zřejmě efektivnější.

GROUP BY – motivace

D14. Zjisti nejvyšší cenu výpujčky a zjisti, které výpujčky se za tuto cenu uskutečnily.

První nápad:

```
SELECT c_kopie, MAX (cena)  
FROM Výpujčky;
```

ERROR: column "vypujcky.c_kopie" must appear in the GROUP BY clause or be used in an aggregate function

Správné řešení uvedeme dále.

GROUP BY

D15. Najdi pro každý film počet herců, kteří v něm hrají.

```
SELECT jmeno_f, COUNT (rod_c_herce) AS pocet_hercu  
FROM Obsazeni  
GROUP BY jméno_f;
```

ZDROJ:

| JMENO_F | HEREC |
|-----------------|-------------|
| Batalion | Vítová H. |
| ... | ... |
| Kristián | Mandlová A |
| Kristián | Nový O. |
| Lízino štěstí | Sulanová Z |
| Madla zpívá | Sulanová Z. |
| Městečko na ... | Boháč L. |
| Městečko na ... | Marvan J. |
| Městečko na ... | Plachta J. |
| ... | ... |
| Rozina sebranec | Glázrová M. |
| Rozina sebranec | Štěpánek P. |
| ... | ... |

VYSLEDEK:

| JMENO_F | POCET_HERCU |
|-----------------|-------------|
| Batalion | 1 |
| ... | ... |
| Kristián | 2 |
| Lízino štěstí | 1 |
| Madla zpívá | 1 |
| | |
| Městečko na ... | 3 |
| ... | ... |
| Rozina sebranec | 2 |
| ... | ... |

Seskupování řádků

D16. Najdi pro každý film z tabulky OBSAZENi počet herců, kteří v něm hrají. Ve výsledku ponech pouze filmy, kde hrají dva a více herců.

```
SELECT jmeno_f, COUNT (herec) AS pocet_hercu  
FROM Obsazeni  
GROUP BY jméno_f  
HAVING COUNT(herec)>1;
```

Výsledek bývá implicitně seřazen podle seskupovacího sloupce.

SELECT se všemi klauzulemi

D17. najdi pro každý film z roku 1945 počet herců, kteří v něm hrají. Ve výsledku ponech filmy, kde hrají dva herci a více. Seřaď výsledek podle počtu herců.

```
SELECT Filmy.jmeno_f, COUNT (herec) AS pocet_hercu  
FROM Obsazení JOIN Filmy USING (jmeno_f)  
WHERE Filmy.rok = 1945  
GROUP BY Filmy.jméno_f  
HAVING COUNT (herec) >= 2  
ORDER BY pocet_hercu;
```

Pořadí vyhodnocení:

- 1 zdroj – klauzule FROM
- 2 selekce – klauzule WHERE
- 3 seskupení – klauzule GROUP BY
- 4 agregační funkce podle výsledků GROUP BY – klauzule SELECT
- 5 selekce na výsledky agregační funkce – klauzule HAVING
- 6 řazení výsledku – klauzule ORDER BY

Nevztažený poddotaz

D18. Vyber filmy, které mají stejného režiséra, jako má film Švadlenka.

```
SELECT F1.jmeno_f  
FROM Filmy F1  
WHERE F1.reziser = (SELECT reziser  
                    FROM Filmy F2  
                    WHERE F2.jmeno_f='Švadlenka');
```

Co když bude v databázi více filmů jménem Švadlenka?

- ❶ Atribut jméno_f je klíčem, dotaz je tedy v tomto případě bezpečný.
- ❷ Pokud nemáme jistotu unikátní hodnoty, **nelze použít “=”**.
- ❸ “=” očekává jako druhý operand jednu hodnotu, nikoliv množinu!

Nevztažený poddotaz

D19. Zjistí nejvyšší cenu výpujčky a zjistí které výpujčky se za tuto cenu uskutečnily.

```
SELECT *  
FROM Vypujcky  
WHERE cena = (SELECT MAX (cena)  
              FROM vypujcky);
```

Vnořený dotaz zde vrátí právě jednu hodnotu.

Vztažené poddotazy

D20. Vyber kina a jejich adresy, kde mají na programu více než 8 filmů.

```
SELECT K.název_k, K.adresa  
FROM Kina K WHERE (SELECT COUNT (jméno_f)  
FROM Představení P  
WHERE P.název_k=K.název_k)>8;
```

Vztažené poddotazy se odvolávají na nadřazený dotaz. Jejich vyhodnocení je obvykle náročnější (dražší) než u dotazů nevztažených.

Vztažené poddotazy

D21. Vyber jména a adresy kin, která hrají alespoň tolik filmů jako kino Mír

```
SELECT DISTINCT K.nazev_k
FROM Kina K
WHERE K.nazev_k <> 'Mír' AND
      (SELECT COUNT(jméno_f)
       FROM Predstaveni P1
       WHERE P1.nazev_k= K.nazev_k) >=
      (SELECT COUNT(jmeno_f)
       FROM Predstaveni P2
       WHERE P2.nazev_k='Mír');
```

Poddotaz v klauzuli SELECT

D22. Vypiš seznam všech filmů a u každého uveď počet jeho kopií.

```
SELECT jmeno_f, COUNT (c_kopie) as pocet_kopii  
FROM Kopie K  
GROUP BY jmeno_f;
```

V odpovědi chybí filmy bez kopií.

```
SELECT F.*, (SELECT COUNT (c_kopie)  
              FROM Kopie K  
              WHERE K.jmeno_f=F.jmeno_f) as pocet_kopii  
FROM Filmy F;
```

Zde jsou ve výsledku i filmy bez kopií, tedy mající 0 kopií.

Poddotaz v klauzuli FROM

D23. Najdi průmernou cenu z minimálních cen kopií pro každého zákazníka.

```
SELECT AVG(T.minim_c)
FROM (SELECT MIN(cena)
      FROM Vypujcky
      GROUP BY rod_c) AS T(minim_c);
```

nebo:

```
SELECT AVG(T.minim_c)
FROM (SELECT MIN(cena) AS minim_c
      FROM Vypujcky
      GROUP BY rod_c) T;
```

Vnější spojení

D24. (znovu) Vypiš seznam všech filmů a u každého uveď počet jeho kopií, včetně filmů bez kopií.

varianta 1: (předchozí slide):

```
SELECT F.*, (SELECT COUNT (c_kopie)
              FROM Kopie K
              WHERE K.jmeno_f=F.jmeno_f) as pocet_kopii
FROM Filmy F;
```

varianta 2: (pomocí vnějšího spojení):

```
SELECT jmeno_f, COUNT (c_kopie) as pocet_kopii
FROM Kopie K RIGHT OUTER JOIN Filmy USING(jmeno_f)
GROUP BY jmeno_f;
```

Vliv prázdné množiny na agregaci

D25. Najdi vedoucí kin, kteří mají zaregistrované výpůjčky kopií za méně než 2000 korun.

```
SELECT DISTINCT jmeno_v  
FROM Kina K JOIN Zakaznici Z on (K.jmeno_v = z.jmeno)  
WHERE (SELECT SUM (V.cena)  
      FROM Vypujcky V  
      WHERE V.rod_c = Z.rod_c) < 2000;
```

Nezahrnuje vedoucí, kteří si nepůjčili nic! (SUM(\emptyset)=NULL)

... včetně těch, kteří si nic nepůjčili.

```
SELECT DISTINCT jmeno_v  
FROM Kina K JOIN Zakaznici Z on (K.jmeno_v = z.jmeno)  
WHERE COALESCE ((SELECT SUM (V.cena)  
                FROM Vypujcky V  
                WHERE V.rod_c = Z.rod_c),0) < 2000;
```


Hodnotové výrazy – výraz CASE

CASE

```
CASE <přepínač>  
WHEN <hodnota1> THEN <výraz1>  
WHEN <hodnota2> THEN <výraz2>  
...  
ELSE <výraz3>  
END
```

D26. Hraje se někde film Falešná kočička?

```
SELECT 'Film falešná kočička se' ||  
      (CASE COUNT(*))  
      WHEN 0 THEN 'ne'  
      ELSE ''  
      END) || 'hraje.'  
FROM Predstaveni  
WHERE jmeno_f = 'Falešná kočička' ;
```

Hodnotové výrazy – výraz CASE

CASE

```
CASE <přepínač>  
WHEN <hodnota1> THEN <výraz1>  
WHEN <hodnota2> THEN <výraz2>  
... ELSE <výraz3>  
END
```

D27. Doplňte seznam výjpuček o příznak levná/drahá.

```
SELECT v.*, (CASE  
                WHEN cena <10 THEN 'levná'  
                WHEN cena >100 Then 'drahá'  
            END)  
FROM Výpůjčka V;
```

Hodnotové výrazy – COALESCE

Funkce COALESCE (V1,V2,..Vn) je ekvivalentní výrazu:

```
CASE  
WHEN V1 IS NOT NULL THEN V1  
WHEN V2 IS NOT NULL THEN V2  
...  
WHEN Vn IS NOT NULL THEN Vn
```

D28. Někteří zaměstnanci nemají plat. Vypiš seznam a místo NULL zobraz 0.

```
SELECT osobni_c, jmeno,  
       COALESCE(PLAT,0) AS Mesicni_prijem  
FROM Zamestnanci;
```

Predikát LIKE

D29. Najdi platy zaměstnanců, kteří jsou z Kolína.

```
SELECT Z.plat  
FROM Zaměstnanci Z  
WHERE Z.adresa LIKE '%Kol_n%';
```

Zástupné symboly

%

skupina znaků (i prázdná)

—

právě jeden znak

řádkové výrazy

Výrazy

Výraz:

$(R.cena, R.datum) = (S.cena, S.datum)$

lze použít namísto:

$R.cena = S.cena \text{ AND } (R.datum = S.datum)$

Výraz:

$(R.cena, R.datum) > (S.cena, S.datum)$

lze použít namísto:

$R.cena > S.cena \text{ OR } (R.cena = S.cena \text{ AND } R.datum > S.datum)$

predikáty IS NULL

- IS [NOT] NULL
- IS [NOT] TRUE
- IS [NOT] FALSE

D30. Vypiš čísla zakázek od výpůjček, které jsou půjčeny neomezeně (chybí hodnota data vrácení)

```
SELECT c_zak  
FROM Vypujcky  
WHERE datum_v IS NULL;
```

Množinový predikát IN

predikát IN – použití

<výraz>[NOT] IN (<výčet_množiny_hodnot>)
<výraz>[NOT] IN (<poddotaz>)

D31. Najdi filmy, s danými režiséry

```
SELECT jméno_f FROM Filmy  
FROM Filmy  
WHERE Reziser IN ('Menzel', 'Chytilová', 'Kachyňa');
```

D32. Najdi adresy kin, ve kterých dávají film Kolja.

```
SELECT adresa  
FROM Kina  
WHERE nazev_k IN (SELECT nazev_k  
                   FROM Predstaveni  
                   WHERE jmeno_f='Kolja');
```

Množinový predikát IN

D33. Najdi jména zákazníků s rezervací filmu od režiséra Menzela

```
SELECT jmeno
FROM Zákazníci
WHERE rod_c IN (SELECT rod_c
                 FROM Rezervace R
                 WHERE R.jmeno_f IN (SELECT F.jmeno_f
                                     FROM Filmy F
                                     WHERE F.reziser = 'Menzel'));
```

- výraz `IN(∅)` vrací `FALSE`
- výraz `IN(ℵ)` vrací `UNKNOWN`

Poznámka: \aleph reprezentuje n-tici (řádek) tvořenou pouze NULL hodnotami.

Množinové predikáty ANY, ALL, SOME

● > SOME

● < SOME

● <> SOME

● = SOME

● > ALL

● < ALL

● <> ALL

● = ALL

synonyma:

● ANY \equiv SOME

● = SOME \equiv IN

● <> ALL \equiv NOT IN

D34. Najdi zaměstnance, kteří mají plat vyšší než všichni zaměstnanci z Prahy.

```
SELECT osobni_c, jmeno
FROM Zamestnanci
WHERE plat > ALL (SELECT Z.plat
                  FROM Zamestnanci Z
                  WHERE Z.adresa LIKE '%Praha%');
```

nebo:

```
SELECT osobni_c, jmeno
FROM Zamestnanci
WHERE plat > (SELECT max(Z.plat)
              FROM Zamestnanci Z
              WHERE Z.adresa LIKE '%Praha%');
```

Množinový predikát UNIQUE

D35. Vypiš jména a adresy z ákazníků, kteří mají nejvýše jednu výpučku.

```
SELECT Z.jmeno, Z.adresa  
FROM Zakaznici Z  
WHERE UNIQUE (SELECT *  
               FROM Vypujcka V  
               WHERE V.rod_c = Z.rod_c);
```

- výraz `UNIQUE(\emptyset)` vrací `TRUE`
- výraz `UNIQUE(\aleph)` vrací `TRUE`
- výraz `EXISTS(\emptyset)` vrací `FALSE`
- výraz `EXISTS(\aleph)` vrací `FALSE`

Poznámka: \aleph reprezentuje n-tici (řádek) tvořenou pouze NULL hodnotami.

Kvantifikace v SQL

- Existenční kvantifikátor $\exists x.P(x)$
v SQL: **[NOT] EXISTS**
prakticky testuje prázdnot/nepřázdnost v množině výsledků
- Univerzální kvantifikátor $\forall x.P(x)$
není v SQL přímo implementován implementovat pomocí \exists :
 $\forall x.P(x) \equiv \neg \exists x.(\neg P(x))$

Každý film má režiséra

Neexistuje film **bez** režiséra.

nebo:

Neexistuje film, pro který **není pravda**, že má režiséra.

Kvantifikace v SQL

D36. Najdi jména zákazníků, kteří mají rezervovaný nějaký film.
D36'. Najdi jména zákazníků takových,
že pro ně existuje záznam o rezervaci některého filmu.

```
SELECT jmeno  
FROM zakazník Z  
WHERE EXISTS (SELECT 1  
               FROM Rezervace  
               WHERE rod_c=Z.rod_c);
```

Nezáleží na tom, co se vybere v klauzuli **SELECT** vnořeného dotazu.
Vyhodnocuje se prázdnot/neprázdnot množiny definované
vnořeným dotazem.

Kvantifikace v SQL

D37. Najdi kina, která nic nehrají.

D37'. Najdi taková kina, pro něž neexistuje představení.

```
SELECT nazev_k  
FROM Kina K  
WHERE NOT EXISTS (SELECT 'X'  
                   FROM Představení  
                   WHERE K.nazev_k=P.nazev_k);
```

Nezáleží na tom, co se vybere v klauzuli **SELECT** vnořeného dotazu.

Vyhodnocuje se prázdnot/neprázdnot množiny definované vnořeným dotazem.

Kvantifikace v SQL

D38. Najdi kino, které hraje **všechna** představení.

D38'. Najdi takové kino, pro něž **neexistuje** představení, které **není na programu** tohoto kina

```
SELECT nazev_k  
FROM Kina K  
WHERE NOT EXISTS (SELECT 1  
                   FROM Predsaveni P  
                   WHERE K.nazev_k <> P.nazev_k);
```

Použita dvojitá negace ve spojení s existenčním kvantifikátorem pro opis univerzálního kvantifikátoru.

Množinové operace

- UNION
- INTERSECT
- EXCEPT ; v Oracle se používá MINUS
- UNION ALL ; neřeší duplicity, je výrazně rychlejší než UNION, netřídí výsledek

D39. Najdi kina, která nic nehrají.

```
(SELECT nazev_k  
FROM Kina)  
EXCEPT  
(SELECT nazev_k  
FROM Predstaveni);
```

Poznánka: Je nezbytné, aby relace (množiny), které vstupují do množinových operací byly vzájemně kompatibilní.

Tedy relace musí mít shodný počet atributů a odpovídající si atributy musí být stejného typu (nemusí se jmenovat stejně).

Množinové operace

D40. Najdi filmy, které jsou rezervované **nebo** půjčené.

```
(SELECT Jmeno_f FROM Rezervace)
```

UNION

```
(SELECT Jmeno_f FROM Vyjpujcky JOIN Filmy USING (c_kopie));
```

D41. Najdi filmy, které jsou rezervované **a** půjčené.

```
(SELECT Jmeno_f FROM Rezervace)
```

INTERSECT

```
(SELECT Jmeno_f FROM Vyjpujcky JOIN Filmy USING (c_kopie));
```

D42. Najdi filmy, které jsou rezervované **a nejsou** půjčené.

```
(SELECT Jmeno_f FROM Rezervace)
```

EXCEPT

```
(SELECT Jmeno_f FROM Vyjpujcky JOIN Filmy USING (c_kopie));
```

V Důsledku eliminace duplicit bývá výsledek implicitně seříděn vzestupně.

Množinové operace

D43. Vypiš adresy zákazníků a zaměstnanců.

```
(SELECT Jmeno,Adresa From Zakaznici)
```

UNION

```
(SELECT Jmeno,Adresa FROM Zamestnanci);
```

Nesmíme zapomenout na kompatibilitnost množin.

... možno zajistit též pomocí **CORRESPONDING**

```
(SELECT * From Zakaznici)
```

UNION CORRESPONDING

```
(SELECT * FROM Zamestnanci);
```