Jazyk SQL – SELECT II.

Michal Valenta

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

BI-DBS, LS 2021/2022

https://courses.fit.cvut.cz/BI-DBS/

SQL SELECT z minulé přednášky

- Základní dotazy
- NULL hodnota
- Spojení tabulek

Agregace

D12. Kolik je filmů natočených v letech 1989-2000

```
SELECT COUNT(*) AS pocet_filmu_89_00
FROM filmy
WHERE rok BETWEEN 1989 AND 2000;
```

D10. Kolik různých filmů se hraje?

```
SELECT COUNT (DISTINCT id_filmu)
FROM predstaveni;
```

D16. Jaký je průměrný plat?

```
SELECT AVG(plat)
FROM zamestnanci;
```

Nezahrnuje zaměstnance bez platu (s platem NULL).

```
SELECT AVG(COALESCE (plat,0))
FROM zamestnanci;
```

Zaměstnanci s platem 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

D14. Najdi počet zaměstnanců s platem menším než 10 000 Kč.

```
SELECT COUNT(*)
FROM zamestnanci
WHERE plat < 10000;
```

D15. Zjisti pro zahraniční zaměstnance celkový objem jejich platů přepočtený na EUR.

```
SELECT SUM (plat)/25.47 AS euro_plat
FROM zaměstnanci
WHERE rodne_cislo IS NULL;
```

nebo:

```
SELECT SUM(plat/25.47) AS euro_plat
FROM zaměstnanci
WHERE rodne_cislo IS NULL;
```

První varianta je zřejmě efektivnější.

Seskupování řádků

D17. Najdi jména zaměstnanců s nejvyšším platem.

První nápad:

```
SELECT jmeno, prijmeni, MAX(plat)
FROM zamestnanci;
```

```
ERROR: column "zamestnanci.jmeno" must appear in the GROUP BY clause or be used in an aggregate function
```

Správné řešení uvedeme dále.

Seskupování řádků (GROUP BY)

D18. Najdi pro každý film počet herců, kteří v něm hrají.

```
SELECT jmeno_f, COUNT (herec) AS pocet_hercu
FROM obsazeni1
GROUP BY jmeno_f;
```

ZDROJ:

JMENO_F	HEREC
...	...
Bůh masakru	Jodie Foster
Bůh masakru	Kate Winslet
Černá labuť	Natalie Portman
...	...
Jeden musí z kola ven ...	Colin Firth
Jeden musí z kola ven ...	Gary Oldman
Kůže, kterou nosím	Antonio Banderas
...	...
Na samotě u lesa	Josef Kemr
Na samotě u lesa	Ladislav Smoljak
Na samotě u lesa	Zdeněk Svěrák
Samotáři	Jiří Macháček
Válka Bohů	Henry Cavill
...	...

VÝSLEDEK:

JMENO_F	HEREC
...	...
Bůh masakru	2
Černá labuť slunce	1
...	...
Jeden musí z kola ven ...	2
Kůže, kterou nosím	1
...	...
Na samotě u lesa	3
Samotáři	1
Válka Bohů	1
...	...

Seskupování řádků (HAVING)

D19. Najdi pro každý film z tabulky obsazeni1 počet herců, kteří v něm hrají, pouze pro případy, je-li jich více než 1.

```
SELECT jmeno_f, COUNT(herec) AS pocet_hercu
FROM obsazeni1
GROUP BY jmeno_f
HAVING COUNT(herec) > 1;
```

Výsledek bývá implicitně seřazen podle seskupovacího sloupce.

Pořadí vyhodnocení

D20. Najdi pro každý film z roku 2011 počet herců, kteří v něm hrají, pouze pro případy, je-li jich více než 1 a seřaď je sestupně dle jména.

```
SELECT obsazeni.jmeno_f, COUNT (herec) AS pocet_hercu
FROM obsazeni1, filmy
WHERE obsazeni1.jmeno_f = filmy.nazev AND rok = 2011
GROUP BY obsazeni1.jmeno_f
HAVING COUNT(*) >= 2
ORDER BY jmeno_f;
```

Pořadí vyhodnocení:

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

Nevztažené poddotazy

DD1. Vyber filmy, které mají stejného režiséra, jako má film Samotáři.

```
SELECT f1.jmeno_f
FROM filmy1 f1
WHERE f1.reziser = (SELECT reziser
                    FROM filmy1 f2
                    WHERE f2.jmeno_f='Samotáři');
```

Co když bude v databázi více filmů jménem Samotáři?

1. Atribut jmeno_f je klíčem, dotaz je tedy v tomto případě bezpečný.
2. Pokud nemáme jistotu unikátní hodnoty, nelze použít "=".
3. "=" očekává jako druhý operand jednu hodnotu, nikoliv množinu!

Nevztažené poddotazy

DD2. Zjisti nejvyšší plat zaměstnance a zjisti jeho jméno a příjmení.

```
SELECT jmeno, prijmeni
FROM zamestnanci
WHERE plat = (SELECT MAX(plat)
              FROM zamestnanci);
```

Vnořený dotaz zde vrátí právě jednu hodnotu.

Vztažené poddotazy

D21. Vyber kina a jejich adresy, kde mají na programu více jak 3 filmy.

```
SELECT distinct nazev, mesto, ulice, cislo_popisne
FROM kina k
WHERE 3 < (SELECT COUNT (id_kina)
           FROM predstaveni p
           WHERE p.id_kina = k.id_kina);
```

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 nazev, mesto, ulice, cislo_popisne
FROM kina k
WHERE (SELECT COUNT (id_kina)
      FROM predstaveni p
      WHERE p.id_kina = k.id_kina) > 3;
```

Poddotaz v klauzuli SELECT

D24. Pro jednotlivá multikina s číslem id_kina 4, 5 a 6 zjisti počet sálů se stejnou kapacitou a také celkovou kapacitu multikina.

```
SELECT id_kina as id_multikina, kapacita as kapacita_salu,
       COUNT(cislo) as pocet_salu_s_touto_kapacitou,
       (SELECT SUM(kapacita) FROM saly s1
       WHERE s1.id_kina=s2.id_kina) as celkova_kapacita_multikina
FROM saly s2
WHERE id_kina between 4 and 6
GROUP BY id_kina, kapacita
ORDER BY id_multikina, kapacita_salu;
```

Poddotaz v klauzuli FROM

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

```
SELECT AVG(T.MIN_plat_kina)
FROM (SELECT MIN(plat)
      FROM zamestnanci
      GROUP BY id_kina) AS T(MIN_plat_kina);
```

nebo:

```
SELECT AVG(T.MIN_plat_kina)
FROM (SELECT MIN(plat) AS MIN_plat_kina
      FROM zamestnanci
      GROUP BY id_kina) T;
```

Vnější spojení

DD3. Vypiš seznam všech filmů a u každého uveď počet herců, zaznamenaných herců.

Varianta 1 (dotaz D22):

```
SELECT F.*,
       (SELECT COUNT(id_herce)
       FROM obsazeni O
       WHERE F.id_filmu = O.id_filmu) AS pocet_hercu
FROM filmy F;
```

Varianta 2 (pomocí vnějšího spojení):

```
SELECT nazev, COUNT(id_herce) AS pocet_hercu
FROM obsazeni O RIGHT OUTER JOIN filmy F USING(id_filmu)
GROUP BY nazev;
```

POZOR, je třeba rozlišovat

- vnější spojení (outer join) – SQL – Normální spojení + levá/pravá/obě relace dodají n-tice, které na druhé straně spojení nemají partnera (chybějící sloupce musí být doplněny pomocí NULL hodnot).
  Prakticky užitečná konstrukce, kterou jsem v RA nezaváděl.
- polo spojení (semi-join) – RA – Redukce n-tic relace na ty, které jsou spojitelné s nějakou n-ticí druhé relace.
  Syntaktická zkratka za spojení a následně projekci na atributy levé nebo pravé relace.
- anti-join – RA – Redukce n-tic relace na ty, které nejsou spojitelné s žádnou n-ticí druhé relace.
  Syntaktická zkratka za množinový rozdíl původní množiny a příslušného polo spojení.

Agregace a prázdné množiny

D25. Vypiš pracovní pozice, pro které je celková suma platu zaměstnanců na této pozici pracujících, menší než 20 000 Kč.

```
SELECT DISTINCT id_pozice, popis_pozice
FROM prac_pozice p
WHERE 20000 > (SELECT SUM (plat)
               FROM zamestnanci z
               WHERE z.id_pozice = p.id_pozice);
```

Nezahrnuje neplacené zaměstnance a pozice, které nemají zaměstnance! (SUM(∅)=NULL)

... včetně těch, kteří nemají plat nebo nejsou evidováni.

```
SELECT DISTINCT id_pozice, popis_pozice
FROM prac_pozice p
WHERE 20000 > COALESCE ((SELECT SUM (plat)
               FROM zamestnanci z
               WHERE z.id_pozice = p.id_pozice),0);
```

CASE

CASE

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

D26. Hraje se někde film Pretty Woman?

```
SELECT CASE COUNT(*)
       WHEN 0 then 'NE'
       ELSE 'ANO'
       END AS 'Hraje se film Pretty Woman?'
FROM predstaveni1
WHERE jmeno_f = 'Pretty Woman';
```

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
```

D25. Vypiš pracovní pozice, pro které je celková suma platu zaměstnanců na této pozici pracujících, menší než 20 000 Kč.

```
SELECT DISTINCT id_pozice, popis_pozice
FROM prac_pozice p
WHERE 20000 > COALESCE ((SELECT SUM (plat)
            FROM zamestnanci z
            WHERE z.id_pozice = p.id_pozice),0);
```

LIKE

D44. Pro kina z Prahy vypiš seznam filmů, které mají na programu. Ve výpisu nechť jsou jen kina, kde hrají 2 a více filmů.

Problém: Město je uvedeno jako součást celé adresy. Navíc nevíme, zda s diakritikou či bez.

```
SELECT k.nazev_k, k.adresa, COUNT(jmeno_f) as pocet_filmu
FROM kina1 k, predstaveni1 p
WHERE k.nazev_k = p.nazev_k and k.adresa LIKE 'Pra_a%'
GROUP BY k.nazev_k, k.adresa
HAVING COUNT(jmeno_f) >= 2;
```

Zástupné symboly

%	skupina znaků (i prázdná)
_	právě jeden znak

ESCAPE - zrušení významu zástupných symbolů

```
LIKE '%AAA\%BBB%' ESCAPE '\'
```

UNIQUE

D39. Vypiš jména a pracovní pozice zaměstnanců, přičemž ať vždy alespoň 3 pracují na stejné pozici.

```
SELECT id_zam, jmeno, id_pozice
FROM zamestnanci z1
WHERE NOT UNIQUE
            SELECT z2.id_pozice
            FROM zamestnanci z2
            WHERE z1.id_pozice=z2.id_pozice
                        AND z1.id_zam != z2.id_zam);
```

- 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.

Řádkové výrazy

Řádkové výrazy

Výraz:
(R.cena, R.datum) = (S.cena, S.datum)
lze použít namísto:
R.cena = S.cena AND (R.datum=S.datum)
Výraz:
(R.cena, R.datum) > (S.cena, S.datum)
lze použít namísto:
R.cena > S.cena OR (R.cena = S.cena AND R.datum > S.datum)

IS NULL

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

D11. Vypiš pro zahraniční zaměstnance jejich platy přepočtené na EUR.

```
SELECT jmeno, prijmeni, plat/24.90 as euro_plat
FROM zamestnanci
WHERE rodne_cislo IS NULL;
```

Množinový predikát IN

Predikát IN – použití

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

D27. Najdi adresy kin, ve kterých dávají film Samotáři.

```
SELECT distinct adresa
FROM kina1
WHERE nazev_k IN
        (SELECT nazev_k FROM predstaveni1
        WHERE jmeno_f = 'Samotáři');
```

D28. Najdi všechny filmy, které dávají v kinech 'Cinema City Zličín' a 'Golden Apple Cinema'.

```
SELECT DISTINCT jmeno_f
FROM predstaveni1
WHERE nazev_k IN ('Cinema City Zličín', 'Golden Apple Cinema');
```

Množinový predikát IN

D30. Najdi adresy kin, ve kterých dávají filmy režiséra Ladislava Smoljaka.

```
SELECT adresa
FROM kina1
WHERE nazev_k IN (SELECT p.nazev_k
                  FROM predstaveni1 p
                  WHERE p.jmeno_f =
                        (SELECT f.jmeno_f
                        FROM filmy1 f
                        WHERE reziser = 'Ladislav Smoljak'));
```

- výraz IN($\emptyset$) vrací FALSE
- výraz IN($\aleph$) vrací UNKNOWN

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

ANY, ALL, SOME

- $>$ SOME
- $<$ SOME
- $<>$ SOME
- $=$ SOME

- $>$ ALL
- $<$ ALL
- $<>$ALL
- $=$ALL

synonyma:

- ANY $\equiv$ SOME
- $=$ SOME $\equiv$ IN
- $<>$ ALL $\equiv$ NOT IN

D32. Najdi zaměstnance, kteří mají plat vyšší než všichni Drábkové.

```
SELECT id_zam, prijmeni
FROM zamestnanci
WHERE plat > ALL  (SELECT z.plat
                   FROM zamestnanci z
                   WHERE z.prijmeni = 'Drábek');
```

nebo:

```
SELECT id_zam, prijmeni
FROM zamestnanci
WHERE plat = (SELECT MAX(z.plat)
                   FROM zamestnanci z
                   WHERE z.prijmeni = 'Drábek');
```

Kvantifikace

- Existenční kvantifikátor $(\exists x)(P(x))$
  v SQL: [NOT] EXISTS
  prakticky testuje prázdnost/neprázdnost v množině výsledků
- Všeobecný kvantifikátor $(\forall x)(P(x))$
  není v SQL implementován přímo, ale 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

D34. Najdi jména filmů, které jsou na programu nějakého kina.
D34'. Najdi jména těch filmů, že existuje představení, ve kterém se hrají.

```
SELECT jmeno_f
FROM filmy1 f
WHERE EXISTS (SELECT *
                    FROM predstaveni1 p
                    WHERE p.jmeno_f =f.jmeno_f);
```

Nezáleží na tom, co se vybere v klauzuli SELECT vnořeného dotazu.
Vyhodnocuje se prázdnost/neprázdnost množiny definované vnořeným dotazem.

Kvantifikace

D36. Najdi kina, která nemají na programu žádný film, ve kterém hraje Jiří Macháček.
D36' Najdi kina, pro která neexistuje představení, ve kterém hraje Jiří Macháček.

```
SELECT *
FROM kina1 k
WHERE NOT EXISTS
        (SELECT *
        FROM predstaveni1 p
        WHERE p.nazev_k = k.nazev_k AND p.jmeno_f IN
                (SELECT jmeno_f
                FROM obsazeni1
                WHERE herec = 'Jiří Macháček'));
```

Nezáleží na tom, co se vybere v klauzuli SELECT vnořeného dotazu.
Vyhodnocuje se prázdnost/neprázdnost množiny definované vnořeným dotazem.

Kvantifikace

DD4. Najdi kino, které hraje všechna představení.
DD4'. Najdi takové kino, pro něž neexistuje představení, které není na programu tohoto kina.

```
SELECT nazev
FROM kina K
WHERE NOT EXISTS (SELECT 1
                  FROM predstaveni P
                  WHERE K.id_kina <> P.id_kina);
```

Poznámka 1: Výsledkem bude buď jedno kino nebo prázdná množina.
Poznámka 2: Použita dvojitá negace ve spojení s existenčním kvantifikátorem pro opis univerzálního kvantifikátoru.
Poznámka 3: Dotaz na všeobecnou kvantifikaci lze v SQL formulovat ještě stejně jak jsme to dělali v relační algebře (přes univerzum vytvořené kartézským součinem), pak s použitím příkazu WITH (viz další přednáška).
Poznámka 4: Lze to i pomocí agregačních funkcí.

Množinové operace

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

D35. Najdi filmy, které se nehrají (nejsou na programu).

```
(SELECT jmeno_f
FROM filmy1)
EXCEPT
(SELECT jmeno_f
FROM obsazeni1);
```

Poznámka: 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

D37. Najdi filmy, které Woody Allen režíroval a nebo v nich hraje.

```
(SELECT jmeno_f FROM filmy1 WHERE reziser='Woody Allen')
UNION
(SELECT jmeno_f FROM obsazeni1 WHERE herec = 'Woody Allen');
```

D38. Najdi filmy, které Woody Allen režíroval a zároveň v nich hraje.

```
(SELECT jmeno_f FROM filmy1 WHERE reziser='Woody Allen')
INTERSECT
(SELECT jmeno_f FROM obsazeni1 WHERE herec = 'Woody Allen');
```

DD5. Najdi filmy, které Woody Allen režíroval a zároveň v nich nehraje.

```
(SELECT jmeno_f FROM filmy1 WHERE reziser='Woody Allen')
EXCEPT
(SELECT jmeno_f FROM obsazeni1 WHERE herec <> 'Woody Allen');
```

V důsledku eliminace duplicit bývá výsledek implicitně setříděn vzestupně.

Množinové operace

DD6. Vypiš adresy zákazníků a zaměstnanců.

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

Nesmíme zapomenout na kompatibilnost množin.

... možno zajistit též pomocí CORRESPONDING

```
(SELECT * FROM Zakaznici)
UNION CORRESPONDING
(SELECT * FROM Zamestnanci);
```

K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)
- Vnořené dotazy: vztažené a nevztažené, v klauzulích SELECT, FROM, WHERE, s operátory =, IN, EXISTS (a negacemi)
- Vnější spojení: OUTER JOIN
- Kvantifikace: existenční (EXISTS), všeobecná (pomocí existenční)
- Množinové operace (pozor na kompatibilitu množin)
- Složitější dotazy lze formulovat různými způsoby
- Pozor na NULL hodnoty