Jazyk SQL

Přehled SQL

1) jazyk pro definici dat (DDL),
2) jazyk pro manipulaci dat (DML),
3) jazyk pro definice pohledů
4) jazyk pro definice IO
5) jazyk pro přiřazení přístupových práv (DCL)
6) řízení transakcí (TCL)
7) systémový katalog
8) jazyk modulů

Schéma příkladu

KINA(NÁZEV_K, ADRESA, JMÉNO_V)
FILMY(JMÉNO_F, REŽISÉR, ROK)
ZÁKAZNÍCI(ROD_Č, JMÉNO, ADRESA)
ZAMĚSTNANCI(OSOBNÍ_Č, ADRESA, JMÉNO, PLAT)
HERCI(ROD_Č, JMÉNO, ADRESA, SPECIALIZACE)
KOPIE(Č_KOPIE, JMÉNO_F)
VÝPŮJČKY (Č_KOPIE,OSOBNÍ_Č, ROD_Č
,CENA,DATUM_V)
REZERVACE(JMÉNO_F, ROD_Č)
PŘEDSTAVENÍ(NÁZEV_K, JMÉNO_F, DATUM)
OBSAZENÍ(JMÉNO_F, ROD_Č_HERCE, ROLE)

Dotazy v SQL

Výraz relační algebry R1(φ)[A1,A2,...,Aj] lze zapsat takto:

```
SELECT A1,A2,...,Aj
FROM R1
WHERE φ
```

Výraz (R1 ×, R2 ×... × Rk)(φ)[A1,A2,...,Aj]

```
SELECT A1,A2,...,Aj
FROM R1,R2,...,Rk
WHERE φ
```

```
SELECT A1,A2,...,Aj
FROM R1 CROSS JOIN R2 CROSS JOIN... Rk
WHERE φ
```

Výraz (R1∗R2∗. . .∗Rk)(φ)[A1,A2,...,Aj]:

```
SELECT A1, A2, ..., Aj
FROM R1 NATURAL JOIN R2 NATURAL JOIN ... NATURAL JOIN Rk
WHERE φ
```

Dotazy v SQL

Výraz (R1 [t1Θ t2] R2) (φ) [A1,A2,...,Aj]:

```
SELECT A1,A2,...,Aj
FROM R1 JOIN R2 ON (R1.t1Θ R2.t2 )
WHERE φ
```

R1×R2 (φ and R1.t1Θ R2.t2) [A1,A2,...,Aj]

```
SELECT A1,A2,...,Aj
FROM R1 CROSS JOIN R2
WHERE φ and (R1.t1Θ R2.t2)
```

Jednoduché dotazy v SQL

D1. Vypiš tabulku jmen režisérů a roků, kdy natočili nějaký film.

```
SELECT režisér, rok FROM Filmy;
```

FILMY

JMÉNO_F	REŽISÉR	ROK
Nevinná	Bínovec V.	1939
Kristián	Frič M.	1939
Madla zpívá …	Bínovec V.	1940
Noční motýl	Čáp F.	1941
Ohnivé léto	Čáp F.	1939
Dceruška k …	Bínovec V.	1940
Eva tropí …	Frič M.	1939
Cesta do hlubin ...	Frič M.	1939

zdroj

REŽISÉR	ROK
Bínovec V.	1939
Frič M.	1939
Bínovec V.	1940
Čáp F.	1941
Čáp F.	1939
Bínovec V.	1940
Frič M.	1939
Frič M.	1939

odpověď

Jednoduché dotazy v SQL

D1. Vypiš tabulku jmen režisérů a roků, kdy natočili nějaký film.

```
SELECT režisér, rok
FROM Filmy ORDER BY režisér, rok ;
```

FILMY

JMÉNO_F	REŽISÉR	ROK
Nevinná	Bínovec V.	1939
Kristián	Frič M.	1939
Madla zpívá …	Bínovec V.	1940
Noční motýl	Čáp F.	1941
Ohnivé léto	Čáp F.	1939
Dceruška k …	Bínovec V.	1940
Eva tropí …	Frič M.	1939
Cesta do hlubin ...	Frič M.	1939

zdroj

REŽISÉR	ROK
Bínovec V.	1939
Bínovec V.	1940
Bínovec V.	1940
Čáp F.	1939
Čáp F.	1941
Frič M.	1939
Frič M.	1939
Frič M.	1939

?

Odpověď

Jednoduché dotazy v SQL

D1. Ve kterých rocích jednotliví režiséři natočili nějaký film.

```
SELECT DISTINCT režisér, rok
FROM Filmy;
```

FILMY

JMÉNO_F	REŽISÉR	ROK
Nevinná	Bínovec V.	1939
Kristián	Frič M.	1939
Madla zpívá …	Bínovec V.	1940
Noční motýl	Čáp F.	1941
Ohnivé léto	Čáp F.	1939
Dceruška k …	Bínovec V.	1940
Eva tropí …	Frič M.	1939
Cesta do hlubin ...	Frič M.	1939

zdroj

REŽISÉR	ROK
Bínovec V.	1939
Bínovec V.	1940
Čáp F.	1939
Čáp F.	1941
Frič M.	1939

odpověď

Řazení bývá díky DISTINCT implicitní

Jednoduché dotazy v SQL

D2. Najdi filmy natočené před rokem 1940.

```
SELECT * FROM Filmy
WHERE rok < 1940;
```

FILMY	JMÉNO_F	REŽISÉR	ROK
	Nevinná	Bínovec V.	1939
	Kristián	Frič M.	1939
	Ohnivé léto	Čáp F.	1939
	Eva tropí …	Frič M.	1939
	Cesta do hlubin ...	Frič M.	1939

odpověď

Dotazy v SQL

SELECT *specifikace_sloupců*
FROM *specifikace_zdroje*
[WHERE *podmínka_selekce*]
[ORDER BY *specifikace_řazení*]

specifikace_sloupců :=
[{DISTINCT | ALL}] { * | *jm_sloupce* [, *jm_sloupce*] . . .}

Logické operátory

=	je rovno
<> nebo !=	není rovno
<	je menší
>	je větší
<=	je menší nebo rovno
>=	je větší nebo rovno
Between	
Like	

Dotazy podmínky selekce

D3. Vypiš z tabulky FILMY řádky týkající se filmů natočených v letech 1938-1940.

```
SELECT *
FROM Filmy
WHERE rok>=1938
        AND rok<= 1940;
```

JMÉNO_F	REŽISÉR	ROK
Nevinná	Bínovec V.	1939
Život je krásný	Brom L.	1940
Ohnivé léto	Čáp F.	1939
Cesta do hlubin ...	Frič M.	1939

alternativa:

```
SELECT * FROM Filmy
WHERE rok BETWEEN 1938 AND 1940;
```

doplněk:

```
SELECT *  FROM Filmy
WHERE rok NOT BETWEEN 1938 AND 1940;
```

```
SELECT *  FROM Filmy
WHERE NOT (rok BETWEEN 1938 AND 1940);
```

Pravdivostní tabulka

Vyhodnocení logických podmínek.

A	B	A and B	A or B	not A
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	UNKNOWN	UNKNOWN	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	UNKNOWN	FALSE	UNKNOWN	TRUE
UNKNOWN	TRUE	UNKNOWN	TRUE	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN

sémantika porovnání:

x Θ y = UNKNOWN právě když alespoň jedno z x, y je NULL

Tedy: NULL = NULL se vyhodnocuje jako UNKNOWN

Dotazy nad více tabulkami

FILMY(JMÉNO_F, REŽISÉR, ROK)
REZERVACE(JMÉNO_F, ROD_Č)

D4. Najdi režiséry, jejichž některé filmy jsou rezervovány.

FILMY [jméno_f = jméno_f] REZERVACE [REŽISÉR]

```
SELECT DISTINCT režisér
FROM Filmy JOIN Rezervace ON  (Filmy.jméno_f =Rezervace.jméno_f);
```

*{FILMY * REZERVACE} [REŽISÉR]*

```
SELECT DISTINCT režisér
FROM Filmy NATURAL JOIN Rezervace;
```

Použití přirozeného spojení je riskantní

```
SELECT DISTINCT režisér
FROM Filmy JOIN Rezervace
      USING(jméno_f);
```

Dotazy – kvalifikace, alias

D5. Najdi co hrají v kterých kinech. Do výsledku uveďte všechny dostupné atributy.

```
SELECT *
FROM Filmy JOIN  Představení On (Představení.jméno_f=Filmy.jméno_f);
```

Chci-li zadat projekci:

```
SELECT Představení.název_k, Filmy.*,
FROM Filmy JOIN  Představení On (Představení.jméno_f=Filmy.jméno_f)
ORDER BY  Představení.název_k;
```

```
SELECT P.název_k, F.*,
FROM Filmy  F JOIN  Představení  P  On (P.jméno_f=F.jméno_f)
ORDER BY  P.název_k;
```

Dotazy – spojení sama se sebou

D6. Najdi dvojice zákazníků (reprezentovaných rodnými čísly), kteří mají stejnou adresu.

```
SELECT Soused1.rod_č AS první, Soused2.rod_č AS druhý
FROM Zákazníci Soused1 JOIN Zákazníci Soused2 USING (adresa)
WHERE Soused1.rod_č < Soused2.rod_č;
```

Zde jsou použita zástupná jména(alias) za účelem odlišení dvou instancí téže relační tabulky

Dotazy – složitější selekce

D7. Najdi kina, kde hrají filmy natočené v roce 1936 nebo 1939, jejichž režisér není Frič a Bínovec.

Filmy [jméno_f= jméno_f] Představení (rok=1936 or rok=1939 …)

```
SELECT P.název_k, F.jméno_f
FROM Filmy F JOIN  Představení P ON (F.jméno_f = P.jméno_f )
WHERE (F.rok = 1936 OR F.rok = 1939)
          AND NOT(F.režisér = 'Frič' OR F.režisér = 'Bínovec V.');
```

Nebo:

```
SELECT P.název_k, F.jméno_f
FROM Filmy F JOIN  Představení P
ON      (F.jméno_f = P.jméno_f AND (F.rok = 1936 OR F.rok = 1939))
WHERE NOT(F.režisér = 'Frič' OR F.režisér = 'Bínovec V.');
```

Atd. …

Aritmetika

D8. Vypiš pro zahraniční zaměstnance platy přepočtené na EUR. Tito zaměstnanci nemají vyplněno rod_č.

```
SELECT jméno, plat/32.65  AS euro_plat
FROM Zaměstnanci
WHERE rod_č IS NULL; -- cizinci
```

- operátory /, *, - a +
- priorita operátorů
- NULL se propaguje do výsledku, tj. je-li jeden z operandů NULL, je výsledek operace NULL.

Agregační funkce

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

```
SELECT COUNT(*) AS počet_filmů_38_40
FROM Filmy
WHERE rok >= 1938 AND rok <= 1940;
```

D10. Kolik různých filmů je rezervovaných?

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

D13. Jaká je průměrná cena za výpůjčku?

```
SELECT AVG(cena) FROM Výpůjčky;
```

Nejsou zahrnuty výpůjčky bez ceny

```
SELECT AVG(COALESCE(cena,0))
FROM Výpůjčky;
```

Agregační funkce

agregační_funkce([ALL|DISTINCT] *jméno_sloupce*)

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(Ø) = ?

Výjimka:

COUNT(A) X COUNT(*)

Agregační funkce

D11. Najdi počet výpůjček s cenou výpůjčky do 899 Kč.

```
SELECT COUNT(*)
FROM Výpůjčky
WHERE cena ≤ 899.00;
```

D12. Zjisti pro zahraniční zaměstnance celkový objem jejich platů přepočtený na EUR.

```
SELECT SUM(plat)/32.65  AS euro_plat
FROM Zaměstnanci
WHERE rod_č IS NULL;
```

Agregační funkci lze volat ve výrazu

Parametrem agregační funkce může být výraz

```
SELECT SUM(plat/32.65)  AS euro_plat
FROM Zaměstnanci
WHERE rod_č IS NULL;
```

Agregační funkce

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

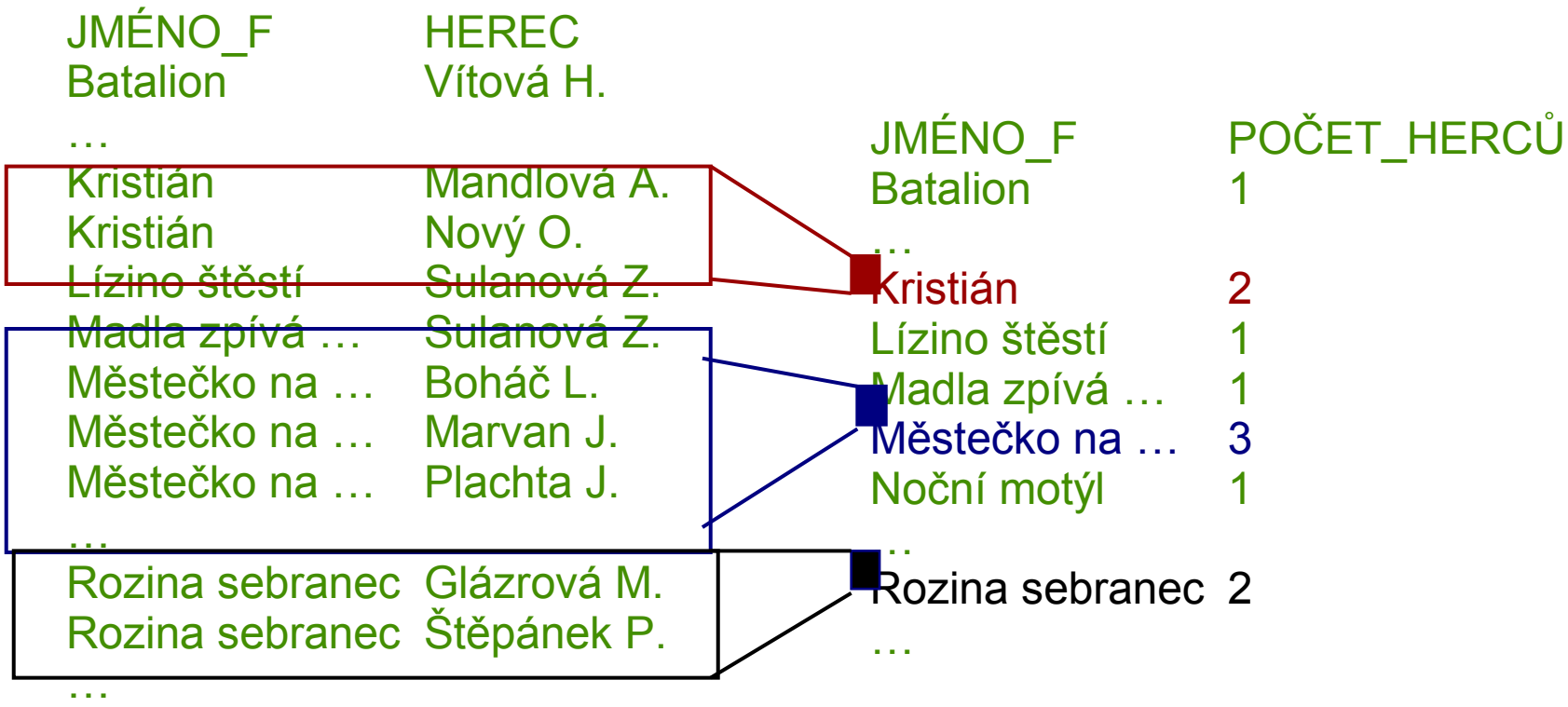
```
SELECT č_kopie, MAX(cena) FROM Výpůjčky;
```

Zatím neumíme

Seskupování řádků

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

```
SELECT jméno_f, COUNT(rod_č_herce) AS počet_herců
                                    FROM Obsazení
                                    GROUP BY jméno_f;
```



Seskupování řádků

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

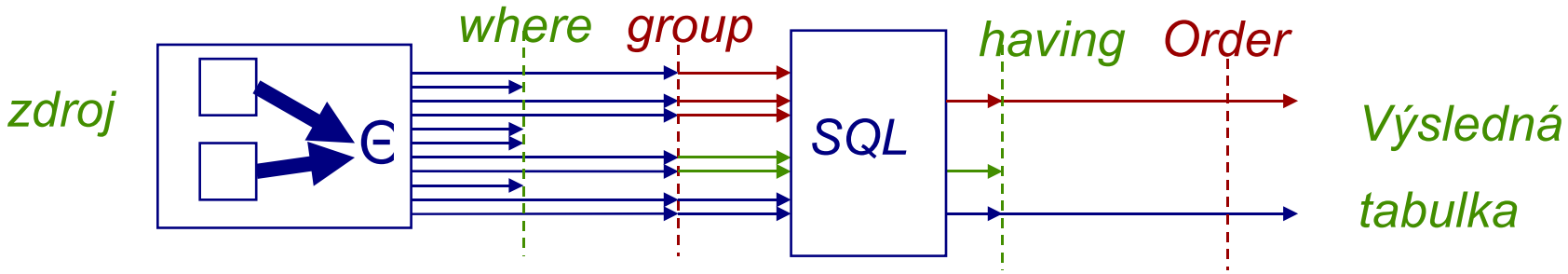
```
SELECT jméno_f, COUNT(herec) AS počet_herců
FROM Obsazení
GROUP BY jméno_f
HAVING COUNT(herec) >1 ;
```

Výsledek bývá implicitně seřazen podle seskupovacího sloupce

Příklad se všemi klauzulemi

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

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



Poddotazy

D21. Vyber filmy, které mají stejného režiséra, jako má film Švadlenka.

```
SELECT F1.jméno_f
FROM Filmy F1
WHERE F1.režisér = ( SELECT   režisér
                     FROM Filmy F2
                     WHERE F2.jméno_f = 'Švadlenka' )
```

skalární operátor

skalární poddotaz

Poddotazy

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

```
SELECT  *
FROM Výpůjčky
WHERE cena = (SELECT MAX(cena)
                FROM Výpůjčky)
```

Vztažený poddotaz

D22. 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 ;
```

Vztažený poddotaz

D23. Vyber jména a adresy kin, kde mají na programu alespoň tolik filmů, jako má kino Mír.

```
SELECT  DISTINCT K.název_k  FROM Kina K
WHERE K.název_k <> 'Mír' AND
(SELECT COUNT(jméno_f)
FROM Představení P1
WHERE P1.název_k=K.název_k)>=(SELECT COUNT(jméno_f)
 FROM Představení P2
 WHERE P2.název_k='Mír'  )
```

Poddotaz v klauzuli Select

D26. Vypiš seznam všech filmů a u každého uveď počet jeho kopií.

```
SELECT jméno_f, COUNT(č_kopie) Počet_kopií
FROM  Kopie K
GROUP BY jméno_f
```

V odpovědi chybí filmy bez kopie

```
SELECT  F.*, (SELECT COUNT(č_kopie)
               FROM  Kopie K
               WHERE K. jméno_f = F. jméno_f) Počet_kopií
FROM Filmy F;
```

Tady nechybí

Vnější spojení

D26 (znovu) Vypiš seznam všech filmů a u každého uveď počet jeho kopií.

```
SELECT  F.*, (SELECT COUNT(č_kopie)
                    FROM  Kopie K
                    WHERE K. jméno_f = F. jméno_f) Počet_kopií
FROM Filmy F;
```

Alternativa:

```
SELECT jméno_f, COUNT(č_kopie) Počet_kopií
FROM  Kopie K RIGHT OUTER JOIN Filmy
          USING(jmeno_f)
GROUP BY jméno_f
```

Vliv prázdné množiny na agregaci

D27. Najdi vedoucí kin, kteří mají zaregistrované výpůjčky kopií za méně než 2000 korun.

```
SELECT  DISTINCT jméno_v
 FROM Kina K JOIN Zákazníci Z ON (K.jméno_v = Z.jméno)
 WHERE (SELECT SUM (V.cena)
              FROM Výpůjčky  V
              WHERE V.rod_č = Z.rod_č) < 2000;
```

?

Zákazníci bez výpůjček z odpovědi vypadnou. *SUM() -> NULL*

```
SELECT  DISTINCT jméno_v
 FROM Kina K JOIN Zákazníci Z ON (K.jméno_v = Z.jméno)
 WHERE COALESCE ((SELECT SUM (V.cena)
                        FROM Výpůjčky  V
                        WHERE V.rod_č = Z.rod_č),0) < 2000;
```

Poddotaz v klauzuli FROM

D28. Najdi průměrnou cenu z minimálních cen kopií pro každého zákazníka.

```
SELECT AVG(T.minim_c)
FROM (SELECT  MIN(cena)
      FROM Výpůjčky
      GROUP BY rod_č)    AS  T(minim_c);
```

```
SELECT AVG(T.minim_c)
FROM (SELECT  MIN(cena) AS minim_c
      FROM Výpůjčky
      GROUP BY rod_č) T;
```

Poddotaz v klauzuli FROM

D30. Vytvoř seznam kopií, do kterého bude přidána odvozená hodnota „počet sesterských kopií“.

```
SELECT K.*, SOUHRN.počet_kopií -1 AS počet_sester
FROM KOPIE K NATURAL JOIN
     (SELECT jméno_f, COUNT(*)  počet_kopií
      FROM   KOPIE
      GROUP BY jméno_f)  SOUHRN
```

Hodnotové výrazy *výrazy CASE*

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

D31. Hraje se někde film Falešná kočička?

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

Hodnotové výrazy

výrazy

CASE

```
CASE
  WHEN <podmínka1>THEN <výraz1>
  WHEN <podmínka2> THEN <výraz2>
  ELSE <výraz3>
END
```

D31. Doplňte seznam výpůjček o příznak levná/drahá?

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

Hodnotové výrazy

- funkce COALESCE

  příklad: COALESCE(Výpůjčky.cena, 0)

Obecněji: COALESCE(V_1,V_2,...,V_n)

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

Hodnotové výrazy

D32. U některých zaměstnanců není vyplněna hodnota ve sloupci PLAT. Vypiš seznam zaměstnanců a v seznamu dodej NULL hodnotě ve sloupci PLAT interpretaci „pracuje zadarmo, tedy s nulovým platem“.

```
SELECT osobní_c
      ,jméno
      ,COALESCE(PLAT,0) Mesicni_prijem
FROM Zaměstnanci;
```

Hodnotové výrazy

- funkce NULLIF

  NULLIF(V1, V2)

  význam:

```
CASE WHEN V1 = V2 THEN NULL
          ELSE V1
END
```

Predikát LIKE

D16. Najdi platy zaměstnanců, kteří jsou z Kolína. Problém je, že nevíme, zda je v datech ‘Kolin’, nebo ‘Kolín‘.

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

Zástupné symboly: % skupina znaků (i prázdná)

_ právě jeden znak

Další predikáty SQL

- řádkové výrazy

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

nahrazuje Boolský výraz

R.cena = S.cena AND (R.datum=S.datum)

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

nahrazuje Boolský výraz

R.cena > S.cena OR (R.cena = S.cena AND R.datum > S.datum)

Další predikáty *IS NULL*

D18. Vypiš čísla zakázek od výpůjček, které jsou půjčeny neomezeně (chybí hodnota data vrácení).

```
SELECT č_zak
FROM Výpůjčky
WHERE datum_v IS NULL;
```

Další predikáty

možnosti: IS [NOT] NULL
IS [NOT] TRUE
IS [NOT] FALSE
IS [NOT] UNKNOWN

podmínka IS	**TRUE**	**FALSE**	**UNKNOWN**
TRUE	**TRUE**	**FALSE**	**FALSE**
FALSE	**FALSE**	**TRUE**	**FALSE**
UNKNOWN	**FALSE**	**FALSE**	**TRUE**

Množinové predikáty *IN*

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

D20. Najdi filmy, s danými režiséry

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

Množinové predikáty *IN*

D19. Najděte adresy kin, ve kterých dávají film Kolja.

```
SELECT adresa FROM Kina
WHERE název_k  IN (SELECT  název_k
                   FROM Představení
                   WHERE jméno_f = 'Kolja');
```

– *výraz* **IN** (∅) vrací FALSE

– *výraz* **IN** (א) vrací UNKNOWN

Množinové predikáty

D21. Najdi jména zákazníků s rezervací filmu od režiséra Menzela.

```
SELECT jméno
FROM Zákazníci
WHERE rod_č IN
(SELECT rod_č FROM Rezervace R
 WHERE R. jméno_f ? (SELECT F.jméno_f
                     FROM Filmy F
                     WHERE F.režisér = 'Menzel'));
```

IN

!!! Pozor na správné použití operátoru = a **IN !!!**

Predikáty ANY, ALL, SOME

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

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

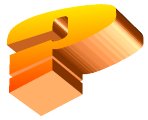
ANY je synonymum pro SOME

Predikáty ANY, ALL, SOME

D22. Najdi zaměstnance, kteří mají plat vyšší než všichni zaměstnanci z Prahy.

```
SELECT osobní_č, jméno
FROM Zaměstnanci
WHERE plat >  ALL (SELECT Z.plat
                   FROM Zaměstnanci Z
                   WHERE Z.adresa LIKE '%Praha%');
```

```
SELECT osobní_č, jméno
FROM Zaměstnanci
WHERE plat > (SELECT max(Z.plat)
              FROM Zaměstnanci Z
              WHERE Z.adresa LIKE '%Praha%');
```



Kvantifikace v SQL

- Existenční kvantifikátor ∃ **x (p(x))**
  SQL: [NOT] EXISTS (poddotaz)
- Univerzální kvantifikátor ∀ **x (p(x))**
  není v SQL implementovaný

Lze použít: ∀ **x (p(x)) = ¬ ∃ x (¬ p(x))**

Př. "Pro všechny filmy platí, že mají režiséra".

Ekvivalentní vyjádření: "**Ne**existuje film, pro který **není** pravdou, že má režiséra".

Jednodušeji:"Každý film má režiséra " je ekvivalentní tvrzení "Neexistuje film bez režiséra".

Kvantifikace v SQL

D23. Najdi jména zákazníků, kteří mají rezervovaný nějaký film

D23'. Najdi jména zákazníků takových, že pro ně existuje záznam o rezervaci některého filmu

```
SELECT Jméno
FROM zákazník Z
WHERE EXISTS (SELECT *  FROM Rezervace
              WHERE rod_č = Z. rod_č);
```

Kvantifikace v SQL

D23b. Najdi kina, která nic nehrají

D23b'. Najdi taková kina, pro něž neexistuje představení

```
SELECT název_k
FROM Kina K
WHERE NOT EXISTS (SELECT *  FROM Představení P
                  WHERE K.název_k = P.název_k);
```

Kvantifikace v SQL *příklad*

Dvojitá negace ve spojení s existenčním kvantifikátorem pro opis univerzálního kvantifikátoru

D24. Najdi kino, které hraje všechna představení

D24'. Najdi takové kino, pro něž **ne**existuje představení, které **ne**ní dáváno tímto kinem

```
SELECT název_k
FROM Kina K
WHERE NOT EXISTS (SELECT *  FROM Představení P
                  WHERE K.název_k <> P.název_k);
```

kardinalita odpovědi?

Další predikáty

UNIQUE poddotaz

D17. Vypiš jména a adresy zákazníků, kteří mají nejvýše jednu výpůjčku

```
SELECT Z.jméno, Z.adresa FROM Zákazníci Z
WHERE UNIQUE( SELECT *
                FROM VYPUJCKA V
               WHERE V.ROD_C = Z.ROD_C);
```

UNIQUE(∅) = TRUE EXISTS(∅) = FALSE

UNIQUE(ℵ) = TRUE EXISTS(ℵ) = FALSE

ℵ tabulka s prázdnými řádky všechny hodnoty jsou NULL

Množinové operace

- UNION
- INTERSECT
- EXCEPT

Množinové operátory eliminují duplikáty

- UNION ALL

Znovu D23b. Najdi kina, která nic nehrají

```
D23b. (SELECT název_k FROM Kina)
      EXCEPT
      (SELECT název_k FROM Představení);
```

Množinové operace

D25. Najdi filmy, které jsou rezervované nebo půjčené

```
(SELECT Jmeno_f FROM Rezervace )
UNION
(SELECT Jmeno_f
 FROM Výpůjčky Join Filmy Using (Č_KOPIE));
```

V důsledku eliminace duplicit bývá výsledek implicitně setříděn vzestupně

Množinové operace

D26. Najdi filmy, které jsou rezervované a nejsou půjčené

```
(SELECT Jmeno_f FROM Rezervace )
EXCEPT
(SELECT Jmeno_f
 FROM Výpůjčky Join Filmy Using (Č_KOPIE));
```

Množinové operace

D27. Najdi filmy, které jsou rezervované a půjčené

```
(SELECT Jmeno_f FROM Rezervace )
INTERSECT
(SELECT Jmeno_f
 FROM Výpůjčky Join Filmy Using (Č_KOPIE));
```

Množinové operace

```
Select Jméno, Adresa From Zákazníci
UNION
Select Jméno, Adresa From Zaměstnanci
```

- Dovětek CORRESPONDING

```
(Select * From Zákazníci)
UNION CORRESPONDING
(Select * From Zaměstnanci)
```

Prázdné hodnoty

Znovu D22: Najdi zaměstnance, kteří mají plat vyšší než všichni zaměstnanci z Prahy.

```
SELECT osobní_č, jméno
FROM Zaměstnanci
WHERE plat >  ALL (SELECT Z.plat
                   FROM Zaměstnanci Z
                   WHERE Z.adresa LIKE '%Praha%');
```

```
SELECT osobní_č, jméno
FROM Zaměstnanci
WHERE plat > (SELECT max(Z.plat)
              FROM Zaměstnanci Z
              WHERE Z.adresa LIKE '%Praha%');
```

Jaká bude odpověď, jestliže neexistují Pražáci?

Aktualizace v SQL

```
DELETE FROM Filmy
WHERE jméno_f = ‘Puška’;
```

Co se bude dít, má-li film kopie, nebo je rezervován?

```
UPDATE Zákazníci  SET jméno = ‘Götzová’
                  WHERE rod_č = '4655292130';
```

```
UPDATE Zákazníci SET jméno = ‘Müller’
                 WHERE jméno = ‘Muller’;
```

Aktualizace v SQL

```
ALTER TABLE Zákazníci
Add Počet_půjček  Number;

UPDATE Zákazníci Z
SET Počet_půjček = (SELECT count(*) from Výpůjčky V
                    WHERE V.rod_č = Z. rod_č);
```

Aktualizace v SQL

```
INSERT INTO Zákazníci (rod_č, jméno)
VALUES ('4804230160',Novák');
```

```
CREATE TABLE Kolik_kopii (rod_c CHAR(10),
                          pocet SMALLINT);
INSERT INTO  Kolik_kopii
        SELECT rod_c, COUNT(c_kopie) FROM Vypujcky
        GROUP BY rod_c;
```

```
CREATE TABLE Kolik_kopii (rod_c CHAR(10),
                          pocet SMALLINT)
AS      SELECT rod_c, COUNT(c_kopie) FROM Vypujcky
        GROUP BY rod_c;
```

Definice dat v SQL

- CREATE TABLE

```
CREATE TABLE VYPUJCKY
(c_kopie    CHAR(3) NOT NULL,
c_zak       CHARACTER(6) NOT NULL,
cena        DECIMAL(5,2),
rod_c       CHARACTER(10) NOT NULL,
datum_v     DATE);
```

Definice dat v SQL

- ALTER TABLE

  ADD *sloupec*, DROP *sloupec*, ALTER *sloupec*,
  ADD CONSTRAINT *io*, DROP CONSTRAINT *io*

Př.: ALTER TABLE KINA ADD pocet_mist INTEGER

- DROP TABLE *tabulka* [CASCADE CONSTRAINTS]

Definice dat v SQL *IO*

- IO sloupce:
  - NOT NULL
  - DEFAULT
  - UNIQUE
  - PRIMARY KEY
  - REFERENCES
  - CHECK
- IO tabulky:

např. složený primární/unikátní/cizí klíč,

- pojmenování IO

Definice dat v SQL

CREATE TABLE *jméno_tabulky*
(*prvek_tabulky*[,*prvek_tabulky*]...)

prvek_tabulky ::={*definice_sloupce* | *IO_tabulky*}

definice_sloupce:= jméno typ [DEFAULT *výraz*]
[*IO_sloupce*] …

IO_sloupce := { [NOT] NULL | UNIQUE | PRIMARY KEY|
REFERENCES *tabulka*[(*klíč*)] |
CHECK(*výraz*)

IO_tabulky :=
UNIQUE(*sloupec*[,*sloupec*]…) |
PRIMARY KEY(*sloupec*[,*sloupec*]…)|
FOREIGN KEY (*sloupec*[,*sloupec*]…) REFERENCES *tabulka*[(*klíč*)] |
CHECK(*výraz*)

Typy dat v SQL

- numerické
- textové
- rozsáhlém znakové řetězce (CLOB)
- rozsáhlém bitové řetězce (BLOB)
- datum a čas
- interval

NULL (je prvkem každého datového typu)

TRUE, FALSE, UNKNOWN

Konverze: implicitní, explicitní (pomocí funkce CAST)

Typy dat v SQL

přesné numerické typy

INTEGER, SMALLINT, NUMERIC, DECIMAL, NUMBER

– DECIMAL(p,q), *p* … přesnost
q … měřítko

aproximativní numerické typy

FLOAT
REAL
DOUBLE PRECISION

Typy dat v SQL

- znakové řetězce

  CHARACTER(n) (délka *n*, zprava mezery)

  CHARACTER VARYING(n) (max.délka *n*)

- datum a čas

  DATE

  TIMESTAMP

  INTERVAL

Příklad DDL

```
. . .
DROP TABLE KINA CASCADE CONSTRAINTS;
CREATE TABLE KINA . . .
. . .
CREATE TABLE PŘEDSTAVENÍ
 (NAZEV_K   Char_Varying(20)          NOT NULL,
  JMENO_F   Char_Varying(20)          NOT NULL,
  DATUM     Date                      NOT NULL,
  PRIMARY KEY (NAZEV_K, JMENO_F),
  FOREIGN KEY (NAZEV_K) REFERENCES KINA,
  FOREIGN KEY (JMENO_F) REFERENCES FILMY);
```

Pohledy

CREATE VIEW
jméno-pohledu [(*v-jméno-atr*[,*v-jméno-atr*]...)]
AS ***dotaz***
[WITH CHECK OPTION]

```
CREATE VIEW Pražáci AS
  SELECT č_čt, jméno, adresa
  FROM Zákazníci WHERE  adresa LIKE '%PRAHA%';
```

```
DROP VIEW Pražáci;
```

```
CREATE VIEW Dlužníci (rod_č, počet_výpůjček) AS
        SELECT rod_č, COUNT(č_kopie) FROM Výpůčky
        GROUP BY rod_č;
```

Pohledy *Aktualizovatelnost pohledu*

pohled je neaktualizovatelný:

- obsahuje-li spojení více tabulek
- obsahuje-li sloupec s odvozenou hodnotu,
- odstiňuje-li projekcí sloupec, na který je uvaleno NOT NULL omezení
- obsahuje-li distinct,
- obsahuje-li agregace a seskupování

Pohledy

```
INSERT INTO Pražáci
VALUES(1234, ’Novák Jiří’,’Pražská 3, Kolín 5’)
```

Dovětek WITH CHECK OPTION

```
CREATE VIEW Pražáci AS
SELECT č_čt, jméno, adresa
FROM Zákazníci WHERE  adresa LIKE '%PRAHA%,
WITH CHECK OPTION;
```

Pohledy - použití

- vytvoření různých pohledů na stejná data
- škálování přístupových práv
- ukrytí složitosti odvození (složitý dotaz skrytý v definici pohledu je navržen pouze jednou)

Příkaz WITH - příklad

Dotaz: Najdi seznam kin, ve kterých hrají všechny filmy s M. Brando.

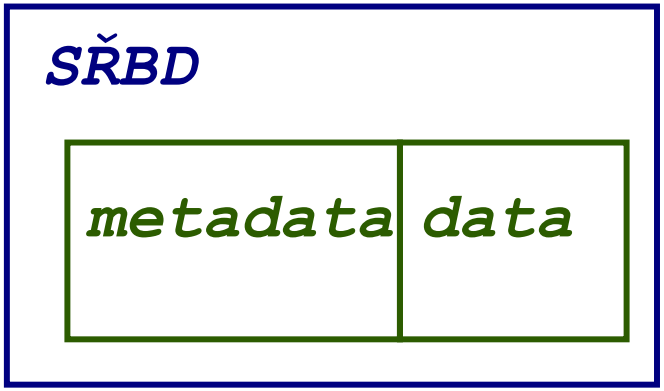
R1:= PROGRAM[NÁZEV_K]
R2:= FILM (HEREC='Brando')[JMÉNO_F]
R:=R1×R2
S:=R \ PROGRAM[NÁZEV_K, JMÉNO_F]
T := S[NÁZEV_K]
U:=PROGRAM[NÁZEV_K] - T

Příkaz WITH - příklad

R1:= PROGRAM[NÁZEV_K]
R2:= FILM (HEREC='Brando')[JMÉNO_F]
R:=R1×R2
S:=R \ PROGRAM[NÁZEV_K, JMÉNO_F]
T := S[NÁZEV_K]
U:=PROGRAM[NÁZEV_K] - T

```
With
    R1 As Select Nazev_k From PROGRAM,
    R2 As Select Jmeno_F From Film Where Herec='Brando',
    R As Select * From R1 Cross Join R2,
    S As (Select * From R) Except (Select Název_k, Jméno_f
                                   From Program),
   T As Select Distinct Nazev_k From S
(Select Distinct Nazev_k From Program)
Except (Select * From T);
```

Systémový katalog



Příklad v databázi ORACLE

Prefix v názvu	rozsah metainformací
USER_	informace pouze o objektech v uživatelově schématu (co uživatel vlastní), např. USER_TABLES
ALL_	jako předešlé, navíc informace o objektech uživateli přístupných, např. ALL_VIEWS
DBA_	informace o všech objektech v databázi, např. DBA_INDEXES

Systémový katalog *příklad*

```
SELECT TABLE_NAME
FROM USER_TABLES
```

```
TABLE_NAME
------------------------------
COUNTRIES
CTENAR
CUSTOMER
DEPT
EMP
...
EXEMPLAR
60 rows selected
```

Systémový katalog

příklad

```
SELECT
 COLUMN_NAME
,CASE
  WHEN DATA_TYP
       DATA_TYP
  WHEN DATA_TYP
       DATA_TYP
  ELSE DATA_TYP
 END AS TYPE
,NULLABLE
FROM USER_TAB_COLUMNS WHERE TABLE_NAME='EMP';
```

COLUMN_NAME	TYPE	NULLABLE
EMPNO	NUMBER(4,0)	N
ENAME	VARCHAR2(10)	Y
JOB	VARCHAR2(9)	Y
MGR	NUMBER(4,0)	Y
HIREDATE	DATE	Y
SAL	NUMBER(7,2)	Y
COMM	NUMBER(7,2)	Y
DEPTNO	NUMBER(2,0)	N

Ochrana dat *uživatelská oprávnění*

- SELECT
- INSERT
- UPDATE
- DELETE
- ALTER
- EXECUTE
- INDEX
- REFERENCE

```
GRANT SELECT ON V_Filmy
TO XNOVAKJ3;

GRANT ALL PRIVILEGES ON
V_filmy TO PUBLIC;

REVOKE INSERT ON Filmy
FROM XNOVAKJ3;
```