

# Jazyk SQL – SELECT II.

Michal Valenta

Katedra softwarového inženýrství  
Fakulta informačních technologií  
České vysoké učení technické v Praze  
©Michal Valenta, 2021

BI-DBS, LS 2020/2021

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



# SQL SELECT z minulé přednášky

- Základní dotazy
- NULL hodnota
- Spojení tabulek

# SQL SELECT z minulé přednášky

- Základní dotazy
- NULL hodnota
- Spojení tabulek

# Obsah

- 1 **Agregační funkce**
- 2 Vnořené dotazy
- 3 Vnější spojení
- 4 Další užitečné konstrukce
  - CASE a COALESCE
  - LIKE, IS NULL, množinové predikáty
  - Kvantifikace
- 5 Množinové operace

# Agregace

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

## D9. Kolik je filmů natočených v letech 1938–1940?

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

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?



## 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;
```

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?

## 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;
```

# Agregace

## 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).
```

# Agregace

## 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)                                SELECT AVG(COALESCE (cena,0))  
FROM Vypujcky;
```

Nezahrnuje výpůjčky bez ceny  
(s cenou NULL).

## 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

# Agregační funkce

Syntaxe:



# Agregační funkce

Syntaxe:

agregační\_funkce ({ALL | DISTINCT} sloupec | výraz)

# Agregační funkce

Syntaxe:

agregační\_funkce ({ALL | DISTINCT} sloupec | výraz)

- COUNT, SUM, MAX, MIN, AVG a mnoho dalších

# Agregační funkce

Syntaxe:

agregační\_funkce ({ALL | DISTINCT} sloupec | výraz)

- COUNT, SUM, MAX, MIN, AVG a mnoho dalších
- Výpočet napříč skupinou zdrojových řádků

# 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?

# 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?

# 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( $\emptyset$ ) =

# 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

# 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



# 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

# 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

# Agregační funkce

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

# Agregační funkce

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

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

# 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.

# 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;
```

# 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:
```



# 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;
```

# 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.

# Seskupování řádků

# Seskupování řádků

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

# Seskupování řádků

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

První nápad:

# Seskupování řádků

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

První nápad:

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

# Seskupování řádků

D14. Zjistí nejvyšší cenu výpůjčky a zjistí, které výpůjčky se za tuto cenu uskutečnily.

První nápad:

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

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

# Seskupování řádků

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

První nápad:

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

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

Správné řešení uvedeme dále.



# Seskupování řádků (GROUP BY)

# Seskupování řádků (GROUP BY)

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

# Seskupování řádků (GROUP BY)

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

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

# Seskupování řádků (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 (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 (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 (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 (HAVING)



# Seskupování řádků (HAVING)

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

# Seskupování řádků (HAVING)

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

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

# Seskupování řádků (HAVING)

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

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

Výsledek bývá implicitně seřazen podle seskupovacího sloupce.

# Pořadí vyhodnocení

# Pořadí vyhodnocení

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.

# Pořadí vyhodnocení

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

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

# Pořadí vyhodnocení

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

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

Pořadí vyhodnocení:

# Pořadí vyhodnocení

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

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

Pořadí vyhodnocení:

- 1 zdroj – klauzule FROM



# Pořadí vyhodnocení

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

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

Pořadí vyhodnocení:

- 1 zdroj – klauzule FROM
- 2 selekce – klauzule WHERE

# Pořadí vyhodnocení

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

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

Pořadí vyhodnocení:

- 1 zdroj – klauzule FROM
- 2 selekce – klauzule WHERE
- 3 seskupení – klauzule GROUP BY

# Pořadí vyhodnocení

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

```
SELECT Filmy.jmeno_f, COUNT (herec) AS pocet_hercu
FROM Obsazeni JOIN Filmy USING (jmeno_f)
WHERE Filmy.rok = 1945
GROUP BY Filmy.jmeno_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

# Pořadí vyhodnocení

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

```
SELECT Filmy.jmeno_f, COUNT (herec) AS pocet_hercu  
FROM Obsazeni JOIN Filmy USING (jmeno_f)  
WHERE Filmy.rok = 1945  
GROUP BY Filmy.jmeno_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

# Pořadí vyhodnocení

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

```
SELECT Filmy.jmeno_f, COUNT (herec) AS pocet_hercu
FROM Obsazeni JOIN Filmy USING (jmeno_f)
WHERE Filmy.rok = 1945
GROUP BY Filmy.jmeno_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

# Pořadí vyhodnocení

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

```
SELECT Filmy.jmeno_f, COUNT (herec) AS pocet_hercu  
FROM Obsazeni JOIN Filmy USING (jmeno_f)  
WHERE Filmy.rok = 1945  
GROUP BY Filmy.jmeno_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

# Obsah

1 Agregační funkce

2 Vnořené dotazy

3 Vnější spojení

4 Další užitečné konstrukce

- CASE a COALESCE
- LIKE, IS NULL, množinové predikáty
- Kvantifikace

5 Množinové operace

# Nevztažené poddotazy



# Nevztažené poddotazy

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

# Nevztažené poddotazy

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');
```

# Nevztažené poddotazy

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?

# Nevztažené poddotazy

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?

# Nevztažené poddotazy

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?

# Nevztažené poddotazy

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?

# Nevztažené poddotazy

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?

- 1 Atribut jmeno\_f je klíčem, dotaz je tedy v tomto případě bezpečný.

# Nevztažené poddotazy

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?

- 1 Atribut jmeno\_f je klíčem, dotaz je tedy v tomto případě bezpečný.
- 2 Pokud nemáme jistotu unikátní hodnoty, **nelze použít "="**.



# Nevztažené poddotazy

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?

- 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

# Nevztažené poddotazy

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

# Nevztažené poddotazy

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

```
SELECT *  
FROM Vypujcky  
WHERE cena =
```

# Nevztažené poddotazy

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

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

# Nevztažené poddotazy

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

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

Vnořený dotaz zde vrátí právě jednu hodnotu.

# Vztažené poddotazy

# Vztažené poddotazy

D20. Vyber jména a adresy kin, kde mají na programu více než 8 filmů.



# Vztažené poddotazy

D20. Vyber jména a adresy kin, kde mají na programu více než 8 filmů.

```
SELECT K.nazev_k, K.adresa  
FROM Kina K WHERE
```

# Vztažené poddotazy

D20. Vyber jména a adresy kin, kde mají na programu více než 8 filmů.

```
SELECT K.nazev_k, K.adresa  
FROM Kina K WHERE (SELECT COUNT (jmeno_f)  
                    FROM Predstaveni P  
                    WHERE
```

## Vztažené poddotazy

D20. Vyber jména a adresy kin, kde mají na programu více než 8 filmů.

```
SELECT K.nazev_k, K.adresa  
FROM Kina K WHERE (SELECT COUNT (jmeno_f)  
FROM Predstaveni P  
WHERE P.nazev_k=K.nazev_k)>8;
```

## Vztažené poddotazy

D20. Vyber jména a adresy kin, kde mají na programu více než 8 filmů.

```
SELECT K.nazev_k, K.adresa  
FROM Kina K WHERE (SELECT COUNT (jmeno_f)  
FROM Predstaveni P  
WHERE P.nazev_k=K.nazev_k)>8;
```

Vztažené poddotazy se odvolávají na nadřazený dotaz.

## Vztažené poddotazy

D20. Vyber jména a adresy kin, kde mají na programu více než 8 filmů.

```
SELECT K.nazev_k, K.adresa  
FROM Kina K WHERE (SELECT COUNT (jmeno_f)  
FROM Predstaveni P  
WHERE P.nazev_k=K.nazev_k)>8;
```

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

# Vztažené poddotazy

# Vztažené poddotazy

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

# 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
```



# 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(jmeno_f)  
       FROM Predstaveni P1  
       WHERE P1.nazev_k=
```

## 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(jmeno_f)  
       FROM Predstaveni P1  
       WHERE P1.nazev_k= K.nazev_k) >=
```

## 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(jmeno_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

# Poddotaz v klauzuli SELECT

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

# 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;
```

# 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.
```

# 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.*,
```



# 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
```

# 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;
```

# 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

# Poddotaz v klauzuli FROM

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

# Poddotaz v klauzuli FROM

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

```
SELECT AVG(T.minim_c)  
FROM
```

# Poddotaz v klauzuli FROM

D23. Najdi průměrnou 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);
```

# Poddotaz v klauzuli FROM

D23. Najdi průměrnou 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:



# Poddotaz v klauzuli FROM

D23. Najdi průměrnou 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
```

# Poddotaz v klauzuli FROM

D23. Najdi průměrnou 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;
```

# Obsah

- 1 Agregáčn  funkce
- 2 Vnořen  dotazy
- 3 Vn jš  spojení
- 4 Dal   u itečné konstrukce
  - CASE a COALESCE
  - LIKE, IS NULL, mno inov  predik ty
  - Kvantifikace
- 5 Mno inov  operace

# Vnější spojení

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

# Vnější spojení

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

Varianta 1 (dotaz D22):

# Vnější spojení

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

Varianta 1 (dotaz D22):

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

# Vnější spojení

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

Varianta 1 (dotaz D22):

```
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):



# Vnější spojení

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

**Varianta 1 (dotaz D22):**

```
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;
```

# Vnější spojení

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

**Varianta 1 (dotaz D22):**

```
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;
```

# POZOR, je třeba rozlišovat

- vnější spojení (outer join) – SQL –

# 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.

# 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 –

# 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.

# 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 –

# 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.



# Obsah

- 1 Agregační funkce
- 2 Vnořené dotazy
- 3 Vnější spojení
- 4 Další užitečné konstrukce
  - CASE a COALESCE
  - LIKE, IS NULL, množinové predikáty
  - Kvantifikace
- 5 Množinové operace

# Agregace a prázdné množiny

# Agregace a prázdné množiny

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

# Agregace a prázdné množiny

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

```
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;
```

# Agregace a prázdné množiny

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

```
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;
```

**Poznámka:** Vedoucí kin mezi zákazníky rozlišíme přes stejné jméno (berte to jen jako ilustrativní příklad, v praxi by takový návrh databáze byl špatný).

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

## Agregace a prázdné množiny

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

```
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;
```

**Poznámka:** Vedoucí kin mezi zákazníky rozlišíme přes stejné jméno (berte to jen jako ilustrativní příklad, v praxi by takový návrh databáze byl špatný).

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

... včetně těch, kteří si nic nepůjčili.

# Agregace a prázdné množiny

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

```
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;
```

**Poznámka:** Vedoucí kin mezi zákazníky rozlišíme přes stejné jméno (berte to jen jako ilustrativní příklad, v praxi by takový návrh databáze byl špatný).

**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;
```

# CASE



# CASE

## CASE

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

# CASE

## CASE

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

D26. Hraje se někde film Falešná kočička?

# 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' ;
```

# CASE

# CASE

## CASE

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

# CASE

## CASE

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

D27. Doplňte seznam výpůjček o příznak levná/drahá.

# CASE

## CASE

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

D27. Doplňte seznam výpůjček o příznak levná/drahá.

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

# COALESCE



# COALESCE

Funkce COALESCE ( $V_1, V_2, \dots, V_n$ ) je ekvivalentní výrazu:

CASE

WHEN  $V_1$  IS NOT NULL THEN  $V_1$

WHEN  $V_2$  IS NOT NULL THEN  $V_2$

...

WHEN  $V_n$  IS NOT NULL THEN  $V_n$

# COALESCE

Funkce COALESCE ( $V_1, V_2, \dots, V_n$ ) 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.

# 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;
```

# LIKE

# LIKE

D29. Najdi platy zaměstnanců, kteří jsou z Kolína.

## D29. Najdi platy zaměstnanců, kteří jsou z Kolína.

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

# LIKE

## D29. Najdi platy zaměstnanců, kteří jsou z Kolína.

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

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

## Zástupné symboly

# LIKE

## D29. Najdi platy zaměstnanců, kteří jsou z Kolína.

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

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

## Zástupné symboly

%	skupina znaků (i prázdná)
—	právě jeden znak

## ESCAPE - zrušení významu zástupných symbolů



# LIKE

## D29. Najdi platy zaměstnanců, kteří jsou z Kolína.

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

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

## Zástupné symboly

%	skupina znaků (i prázdná)
—	právě jeden znak

## ESCAPE - zrušení významu zástupných symbolů

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

# Řádkové výrazy

# Řádkové výrazy

## Řádkové výrazy

# Řádkové výrazy

## Řádkové výrazy

Výraz:

# Řádkové výrazy

## Řádkové výrazy

Výraz:

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

# Řádkové výrazy

## Řádkové výrazy

Výraz:

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

lze použít namísto:

# Řádkové výrazy

## Řádkové výrazy

Výraz:

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

Ize použít namísto:

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

Výraz:

# Řádkové výrazy

## Řádkové výrazy

Výraz:

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

Ize použít namísto:

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

Výraz:

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



# Řádkové výrazy

## Řádkové 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:

# Řádkové výrazy

## Řádkové 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:

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

# IS NULL

# IS NULL

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

# 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).

# 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

# Množinový predikát IN

## Predikát IN – použití



# Množinový predikát IN

## Predikát IN – použití

<výraz>[NOT] IN (<výčet\_množiny\_hodnot>)

<výraz>[NOT] IN (<poddotaz>)

D31. Najdi filmy podle seznamu režisérů.

# Množinový predikát IN

## Predikát IN – použití

<výraz>[NOT] IN (<výčet\_množiny\_hodnot>)  
<výraz>[NOT] IN (<poddotaz>)

D31. Najdi filmy podle seznamu režisérů.

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

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

# Množinový predikát IN

## Predikát IN – použití

<výraz>[NOT] IN (<výčet\_množiny\_hodnot>)  
<výraz>[NOT] IN (<poddotaz>)

### D31. Najdi filmy podle seznamu režisérů.

```
SELECT jméno_f  
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

# Množinový predikát IN

D33. Najdi jména zákazníků s rezervací filmu od režiséra Menzela.

# Množinový predikát IN

D33. Najdi jména zákazníků s rezervací filmu od režiséra Menzela.

```
SELECT jmeno  
FROM Zákazníci  
WHERE rod_c IN
```

# 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
```

# 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'));
```



# 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`

# 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.

# ANY, ALL, SOME

# ANY, ALL, SOME

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

# ANY, ALL, SOME

- $> \text{SOME}$
- $< \text{SOME}$
- $<> \text{SOME}$
- $= \text{SOME}$
- $> \text{ALL}$
- $< \text{ALL}$
- $<> \text{ALL}$
- $= \text{ALL}$

# ANY, ALL, SOME

- $> \text{SOME}$

- $< \text{SOME}$

- $<> \text{SOME}$

- $= \text{SOME}$

- $> \text{ALL}$

- $< \text{ALL}$

- $<> \text{ALL}$

- $= \text{ALL}$

synonyma:

# ANY, ALL, SOME

- $> \text{SOME}$

- $< \text{SOME}$

- $<> \text{SOME}$

- $= \text{SOME}$

- $> \text{ALL}$

- $< \text{ALL}$

- $<> \text{ALL}$

- $= \text{ALL}$

synonyma:

- $\text{ANY} \equiv \text{SOME}$

- $= \text{SOME} \equiv \text{IN}$

- $<> \text{ALL} \equiv \text{NOT IN}$

# ANY, ALL, SOME

- $> \text{SOME}$

- $< \text{SOME}$

- $<> \text{SOME}$

- $= \text{SOME}$

- $> \text{ALL}$

- $< \text{ALL}$

- $<> \text{ALL}$

- $= \text{ALL}$

synonyma:

- $\text{ANY} \equiv \text{SOME}$

- $= \text{SOME} \equiv \text{IN}$

- $<> \text{ALL} \equiv \text{NOT IN}$



# ANY, ALL, SOME

- $> \text{SOME}$

- $< \text{SOME}$

- $<> \text{SOME}$

- $= \text{SOME}$

- $> \text{ALL}$

- $< \text{ALL}$

- $<> \text{ALL}$

- $= \text{ALL}$

synonyma:

- $\text{ANY} \equiv \text{SOME}$

- $= \text{SOME} \equiv \text{IN}$

- $<> \text{ALL} \equiv \text{NOT IN}$

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

# ANY, ALL, SOME

- $> \text{SOME}$

- $< \text{SOME}$

- $<> \text{SOME}$

- $= \text{SOME}$

- $> \text{ALL}$

- $< \text{ALL}$

- $<> \text{ALL}$

- $= \text{ALL}$

synonyma:

- $\text{ANY} \equiv \text{SOME}$

- $= \text{SOME} \equiv \text{IN}$

- $<> \text{ALL} \equiv \text{NOT IN}$

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

```
SELECT osobni_c, jmeno  
FROM Zamestnanci  
WHERE plat > ALL
```

# 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:

# 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 >
```

# 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%');
```

# UNIQUE

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

```
SELECT Z.jmeno, Z.adresa  
FROM Zakaznici Z  
WHERE UNIQUE
```



# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

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

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

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

- výraz `UNIQUE(∅)` vrátí `TRUE`

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

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

- výraz `UNIQUE(∅)` vrací `TRUE`
- výraz `UNIQUE(N)` vrací `TRUE`

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

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

- výraz `UNIQUE(∅)` vrací `TRUE`
- výraz `UNIQUE(N)` vrací `TRUE`
- výraz `EXISTS(∅)` vrací `FALSE`

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

```
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( $\mathbb{N}$ )** vrací **TRUE**
- výraz **EXISTS( $\emptyset$ )** vrací **FALSE**
- výraz **EXISTS( $\mathbb{N}$ )** vrací **FALSE**

# UNIQUE

D35. Vypiš jména a adresy zákazníků, kteří mají **nejvýše** jednu výpůjčku.

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

- výraz `UNIQUE(∅)` vrací `TRUE`
- výraz `UNIQUE(ℵ)` vrací `TRUE`
- výraz `EXISTS(∅)` vrací `FALSE`
- výraz `EXISTS(ℵ)` vrací `FALSE`

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

- Existenční kvantifikátor  $(\exists x)(P(x))$

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: [NOT] EXISTS



# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnotu/neprázdnotu v množině výsledků

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnot/neprázdnot v množině výsledků
- Všeobecný kvantifikátor  $(\forall x)(P(x))$

# Kvantifikace

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

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnot/neprázdnot v množině výsledků
- Všeobecný kvantifikátor  $(\forall x)(P(x))$   
není v SQL implementován přímo, ale pomocí  $\exists$ :  
 $(\forall x)(P(x)) \equiv \neg(\exists x)(\neg P(x))$

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnotu/neprázdnotu 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

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnotu/neprázdnotu 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.

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnot/neprázdnot 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:

# Kvantifikace

- Existenční kvantifikátor  $(\exists x)(P(x))$   
v SQL: **[NOT] EXISTS**  
prakticky testuje prázdnot/neprázdnot 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

D36. Najdi jména zákazníků, kteří mají rezervovaný nějaký film.

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

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
```

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);
```

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

D37. Najdi kina, která nic nehrají.



D37. Najdi kina, která nic nehrají.

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

D37. Najdi kina, která nic nehrají.

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

```
SELECT nazev_k  
FROM Kina K  
WHERE NOT EXISTS
```

# Kvantifikace

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);
```

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

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

# Kvantifikace

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

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

# Kvantifikace

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
```



# Kvantifikace

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 Predstaveni P  
                   WHERE
```

# Kvantifikace

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 Predstaveni P  
                   WHERE K.nazev_k <> P.nazev_k);
```

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 Predstaveni P  
                   WHERE K.nazev_k <> P.nazev_k);
```

**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.

# Obsah

- 1 Agregační funkce
- 2 Vnořené dotazy
- 3 Vnější spojení
- 4 Další užitečné konstrukce
  - CASE a COALESCE
  - LIKE, IS NULL, množinové predikáty
  - Kvantifikace
- 5 Množinové operace

# Množinové operace

# Množinové operace

- UNION

# Množinové operace

- UNION
- INTERSECT

# Množinové operace

- UNION
- INTERSECT
- EXCEPT



# Množinové operace

- UNION
- INTERSECT
- EXCEPT; v Oracle se používá MINUS

# Množinové operace

- UNION
- INTERSECT
- EXCEPT; v Oracle se používá MINUS
- UNION ALL

# 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.

# 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
```

# 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);
```

# 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);
```

**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

# Množinové operace

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



# Množinové operace

D40. Najdi filmy, které jsou rezervované nebo půjčené.

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

UNION

# Množinové operace

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

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

**UNION**

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

# Množinové operace

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

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

**UNION**

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

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

# Množinové operace

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

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

**UNION**

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

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

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

**INTERSECT**

# Množinové operace

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

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

**UNION**

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

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

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

**INTERSECT**

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

# Množinové operace

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

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

**UNION**

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

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

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

**INTERSECT**

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

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

# Množinové operace

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

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

**UNION**

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

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

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

**INTERSECT**

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

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

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

**EXCEPT**

# Množinové operace

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

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

**UNION**

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

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

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

**INTERSECT**

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

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

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

**EXCEPT**

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



# Množinové operace

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

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

**UNION**

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

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

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

**INTERSECT**

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

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

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

**EXCEPT**

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

V důsledku eliminace duplicit bývá výsledek implicitně seříděn vzestupně.

# Množinové operace

D43. Vypiš adresy zákazníků a zaměstnanců.

D43. Vypiš adresy zákazníků a zaměstnanců.

(SELECT Jmeno, Adresa FROM Zakaznici)

UNION

# Množinové operace

D43. Vypiš adresy zákazníků a zaměstnanců.

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

**UNION**

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

# Množinové operace

D43. Vypiš adresy zákazníků a zaměstnanců.

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

**UNION**

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

Nesmíme zapomenout na kompatibilitnost množin.

# 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**

# 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);
```



# K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)

# K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)
- Vnořené dotazy: vztažené a nevztažené, v klauzích SELECT, FROM, WHERE, s operátory =, IN, EXISTS (a negacemi)

# 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

# K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)
- Vnořené dotazy: vztažené a nevztažené, v klauzích SELECT, FROM, WHERE, s operátory =, IN, EXISTS (a negacemi)
- Vnější spojení: OUTER JOIN
- Kvantifikace: existenční (EXISTS), všeobecná (pomocí existenční)

# K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)
- Vnořené dotazy: vztažené a nevztažené, v klauzích 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)

# K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)
- Vnořené dotazy: vztažené a nevztažené, v klauzích 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

# K zapamatování

- Agregace v SQL (včetně GROUP BY a HAVING)
- Vnořené dotazy: vztažené a nevztažené, v klauzích 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