

SQL - SELECT

Informační a znalostní systémy

1 SQL

SQL - Structured Query Language

- norma pro dotazování nad relačními databáze
- díky přenositelnosti - rozmach relačních databází
- zahrnuje jak dotazování na data, tak změny v databázi

SQL - Structured Query Language

- norma pro dotazování nad relačními databáze
- díky přenositelnosti - rozmach relačních databází
- zahrnuje jak dotazování na data, tak změny v databázi
 - 1 dotazování (čtení) - příkaz SELECT
 - 2 vkládání, aktualizace, mazání - tzv CRUD - příště

Vyhodnocení výrazu

- Jednoduché dotazování - vyhodnocení výrazů

- Syntax:

```
SELECT výraz [, výraz ]*;
```

- Příklad: vyhodnoťte výraz $1 + 1$

```
SELECT 1+1;
```

Vyhodnocení výrazu

- Jednoduché dotazování - vyhodnocení výrazů

- Syntax:

```
SELECT výraz [, výraz ]*;
```

- Příklad: vyhodnoťte výraz $1 + 1$

```
SELECT 1+1;
```

- Příklad: vyhodnoťte výraz, konstantu a funkci

```
SELECT 1+1, 'textová hodnota', now();
```

Získání záznamů z tabulky

- Data jsou typicky uložena v formě záznamů v tabulce

- Syntax:

```
SELECT výčet výrazů a atributů FROM tabulka;
```

- Pro všechny atributy - použít výrazu *

Získání záznamů z tabulky

- Data jsou typicky uložena v formě záznamů v tabulce
 - Syntax:

```
SELECT výčet výrazů a atributů FROM tabulka;
```

- Pro všechny atributy - použít výrazu *
- Příklad: vypíše všechny záznamy z tabulky budova

```
SELECT * FROM budova;
```


Získání záznamů z tabulky

- Data jsou typicky uložena v formě záznamů v tabulce
 - Syntax:

```
SELECT výčet výrazů a atributů FROM tabulka;
```

- Pro všechny atributy - použít výrazu *
- Příklad: vypíše všechny záznamy z tabulky budova

```
SELECT * FROM budova;
```

- Vždy vypisujeme pouze potřebné atributy

z důvodu režie přenosu, ale i alokace paměti

```
SELECT budova, adresa FROM budova;
```

WHERE - podmínka na zobrazený záznam

- Mnohdy vyžadujeme vypsát pouze záznamy splňující nějakou logickou podmínku
 - Syntax:

```
SELECT ... FROM tabulka WHERE podmínka;
```

WHERE - podmínka na zobrazený záznam

- Mnohdy vyžadujeme vypsát pouze záznamy splňující nějakou logickou podmínku

- Syntax:

```
SELECT ... FROM tabulka WHERE podmínka;
```

- Příklad: vypíšte záznamy z tabulky budova, jejichž id je menší než 3

```
SELECT
    budova, adresa
FROM budova
WHERE id_budova<3;
```

WHERE - podmínka na zobrazený záznam

- Mnohdy vyžadujeme vypsát pouze záznamy splňující nějakou logickou podmínku

- Syntax:

```
SELECT ... FROM tabulka WHERE podmínka;
```

- Příklad: vypište záznamy z tabulky budova, jejichž id je menší než 3

```
SELECT
    budova, adresa
FROM budova
WHERE id_budova<3;
```

- Podmínku lze komponovat z více výrazů pomocí logických spojek AND, OR a NOT

ORDER BY - řazení záznamů

- Pořadí záznamů v databázi je "náhodné", proto je řadíme
 - Syntax:

```
SELECT ... FROM tabulka
      ORDER BY výraz {DESC|ASC} [, výraz {DESC,ASC}]*;
```

ORDER BY - řazení záznamů

- Pořadí záznamů v databázi je "náhodné", proto je řadíme

- Syntax:

```
SELECT ... FROM tabulka
      ORDER BY výraz {DESC|ASC} [, výraz {DESC,ASC}]*;
```

- Příklad: seřaďte budovy podle označení

```
SELECT
      budova, adresa
FROM budova
ORDER BY budova;
```

ORDER BY - řazení záznamů

- Pořadí záznamů v databázi je "náhodné", proto je řadíme

- Syntax:

```
SELECT ... FROM tabulka
      ORDER BY výraz {DESC|ASC} [, výraz {DESC,ASC}]*;
```

- Příklad: seřaďte budovy podle označení

```
SELECT
      budova, adresa
FROM budova
ORDER BY budova;
```

- řazení

- 1 vzestupně - nic nebo ASC
- 2 sestupně - DESC

LIMIT OFFSET - stránkování záznamů

- Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

- Syntax:

```
SELECT ... FROM tabulka LIMIT n;
```


LIMIT OFFSET - stránkování záznamů

- Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

- Syntax:

```
SELECT ... FROM tabulka LIMIT n;
```

- Příklad: zobrazte první 3 záznamy z tabulky budova

```
SELECT  
    *  
FROM budova  
LIMIT 3;
```

LIMIT OFFSET - stránkování záznamů

- Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

- Syntax:

```
SELECT ... FROM tabulka LIMIT n;
```

- Příklad: zobrazte první 3 záznamy z tabulky budova

```
SELECT
    *
FROM budova
LIMIT 3;
```

- ke stránkování slouží klíčové slovo OFFSET

od kterého záznamu se bude zobrazovaných n záznamů počítat

- Příklad: zobrazte 2 až (2+3) záznam

```
SELECT
    *
FROM budova
LIMIT 3 OFFSET 2;
```

LIMIT OFFSET - stránkování záznamů

- Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů
 - Syntax:

```
SELECT ... FROM tabulka LIMIT n;
```

LIMIT OFFSET - stránkování záznamů

- Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

- Syntax:

```
SELECT ... FROM tabulka LIMIT n;
```

- Příklad: zobrazte první 3 záznamy z tabulky budova

```
SELECT
    *
FROM budova
LIMIT 3;
```

LIMIT OFFSET - stránkování záznamů

- Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

- Syntax:

```
SELECT ... FROM tabulka LIMIT n;
```

- Příklad: zobrazte první 3 záznamy z tabulky budova

```
SELECT
    *
FROM budova
LIMIT 3;
```

- ke stránkování slouží klíčové slovo OFFSET

od kterého záznamu se bude zobrazovaných n záznamů počítat

- Příklad: zobrazte 2 až (2+3) záznam

```
SELECT
    *
FROM budova
LIMIT 3 OFFSET 2;
```

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```
- 2 Seřadte je podle času (sestupně), identifikátoru senzoru a veličiny:

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```

- 2 Seřadte je podle času (sestupně), identifikátoru senzoru a veličiny:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina;
```

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```

- 2 Seřadte je podle času (sestupně), identifikátoru senzoru a veličiny:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina;
```

- 3 Vypište posledních 5 měření:

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```

- 2 Seřadte je podle času (sestupně), identifikátoru senzoru a veličiny:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina;
```

- 3 Vypište posledních 5 měření:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina LIMIT 5;
```

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```

- 2 Seřadte je podle času (sestupně), identifikátoru senzoru a veličiny:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina;
```

- 3 Vypište posledních 5 měření:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina LIMIT 5;
```

- 4 Vypište časy a místnosti, kde teplota přesáhla 25 C:

Samostatná práce

- 1 Zobrazte všechny záznamy z tabulky mereni:

```
SELECT * FROM mereni;
```

- 2 Seřadte je podle času (sestupně), identifikátoru senzoru a veličiny:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina;
```

- 3 Vypište posledních 5 měření:

```
SELECT * FROM mereni  
ORDER BY cas DESC, id_senzor, id_velicina LIMIT 5;
```

- 4 Vypište časy a místnosti, kde teplota přesáhla 25 C:

```
SELECT cas, id_mistnost FROM mereni  
WHERE id_velicina = 5 AND hodnota > 25;
```

Spojování tabulek

- Díky dekompozici data rozprostřena mezi různé tabulky

```
SELECT * FROM mereni;
```

Spojování tabulek

- Díky dekompozici data rozprostřena mezi různé tabulky
`SELECT * FROM mereni;`
- V tabulce mereni je cizí klíč reprezentovaný atributem `id_velicina` vedoucí k primárnímu klíči `id_velicina` v tabulce `velicina`

```
SELECT * FROM mereni,velicina
WHERE mereni.id_velicina = velicina.id_velicina
```

Spojování tabulek

- Díky dekompozici data rozprostřena mezi různé tabulky

```
SELECT * FROM mereni;
```

- V tabulce mereni je cizí klíč reprezentovaný atributem id_velicina vedoucí k primárnímu klíči id_velicina v tabulce velicina

```
SELECT * FROM mereni,velicina
      WHERE mereni.id_velicina = velicina.id_velicina
```

- Nevýhodou tohoto zápisu - míchání vazebních podmínek a podmínek na záznam. Např

```
SELECT * FROM mereni,velicina
      WHERE mereni.id_velicina = velicina.id_velicina
      AND hodnota > 25
```


Spojování tabulek

- Proto SQL nabízí možnost spojovat tabulky přímo v rámci klíčového slova FROM

```
SELECT *  
    FROM mereni  
        JOIN velicina  
            ON (mereni.id_velicina = velicina.id_velicina)  
WHERE hodnota > 25 ;
```

Spojování tabulek

- Proto SQL nabízí možnost spojovat tabulky přímo v rámci klíčového slova FROM

```
SELECT *  
  FROM mereni  
       JOIN velicina  
          ON (mereni.id_velicina = velicina.id_velicina)  
 WHERE hodnota > 25 ;
```

- Pokud atributy cizího a primárního klíče mají stejné jméno

```
SELECT *  
  FROM mereni  
       JOIN velicina USING (id_velicina)
```

Spojování tabulek

- Proto SQL nabízí možnost spojovat tabulky přímo v rámci klíčového slova FROM

```
SELECT *  
    FROM mereni  
        JOIN velicina  
            ON (mereni.id_velicina = velicina.id_velicina)  
    WHERE hodnota > 25 ;
```

- Pokud atributy cizího a primárního klíče mají stejné jméno

```
SELECT *  
    FROM mereni  
        JOIN velicina USING (id_velicina)
```

- SQL umožňuje spojování tabulek řetěžit

```
SELECT *  
    FROM mereni  
        JOIN velicina USING (id_velicina)  
        JOIN senzor USING (id_senzor)
```

INNER JOIN - Vnitřní spojování tabulek

- V případě nulové minimální kardinality vztahu potřeba definovat způsob spojení
 - 1 INNER (vnitřní) spojení spojuje páry pouze s vyplněnými hodnotami

```
SELECT *  
  FROM mistnost  
       JOIN budova USING (id_budova)
```

INNER JOIN - Vnitřní spojování tabulek

- V případě nulové minimální kardinality vztahu potřeba definovat způsob spojení

- 1 INNER (vnitřní) spojení spojuje páry pouze s vyplněnými hodnotami

```
SELECT *  
FROM mistnost  
JOIN budova USING (id_budova)
```

- 2 OUTER (vnější) spojení doplňuje atributy, je-li to možné

```
SELECT *  
FROM mistnost  
LEFT JOIN budova USING (id_budova)
```

- 1 LEFT - k pravé části se připojuje levá
- 2 RIGHT - k levé části se připojuje pravá
- 3 FULL - kombinace LEFT a RIGHT

Agregace

- SQL umožňuje vyhodnotit i agregační funkce
 - základní agregační funkce: MAX,MIN,SUM,AVG,COUNT
- Příklad: Stanovte průměrnou teplotu na jednotlivých senzorech

Agregace

- SQL umožňuje vyhodnotit i agregační funkce
 - základní agregační funkce: MAX,MIN,SUM,AVG,COUNT
- Příklad: Stanovte průměrnou teplotu na jednotlivých senzorech

1 nejprve si připravíme vstupní data

```
SELECT id_mistnost,hodnota *  
FROM mereni JOIN senzor USING (id_senzor)  
WHERE id_velicina = 5;
```

Agregace

- SQL umožňuje vyhodnotit i agregační funkce
 - základní agregační funkce: MAX,MIN,SUM,AVG,COUNT
- Příklad: Stanovte průměrnou teplotu na jednotlivých senzorech

- 1 nejprve si připravíme vstupní data

```
SELECT id_mistnost,hodnota *  
FROM mereni JOIN senzor USING (id_senzor)  
WHERE id_velicina = 5;
```

- 2 agregujeme je přes atribut id_mistnost

```
SELECT id_mistnost,AVG(hodnota)  
FROM mereni JOIN senzor USING (id_senzor)  
WHERE id_velicina = 5  
GROUP BY id_mistnost;
```


Agregace - podmínky

- Mnohdy je potřeba stanovit podmínku na hodnotu agregační funkce
 - Např. chceme průměry teplot, pouze pokud máme více než 10 měření

Agregace - podmínky

- Mnohdy je potřeba stanovit podmínku na hodnotu agregační funkce
 - Např. chceme průměry teplot, pouze pokud máme více než 10 měření
- podmínky na hodnoty agregační funkce - klíčové slovo HAVING

```
SELECT id_mistnost,AVG(hodnota)
FROM mereni JOIN senzor USING (id_senzor)
WHERE id_velicina = 5
GROUP BY id_mistnost
HAVING count(*)>10 ;
```

Závěrečné poznámky

- pořadí klíčových slov nelze měnit
 - 1 výčet atributů
 - 2 FROM tabulky
 - 3 WHERE podmínky na záznamy
 - 4 GROUP BY - stanovení agregační úrovně, evaluace hodnot agregačních funkcí
 - 5 HAVING - podmínky na evaluované agregační funkce
 - 6 ORDER BY - seřazení záznamů
 - 7 LIMIT - počet zobrazených záznamů
 - 8 OFFSET - pořadí prvního zobrazeného záznamu