# Agregace

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/DATAB Databáze

# Klíče

# Příklad

datab_05_agregace_vstup.sql

```
CREATE TABLE studenti (
    id_student  INT,
    jmeno TEXT,
    prijmeni TEXT,
    rocnik INT,
    pohlavi TEXT,
    email TEXT
    );
```

| id_student integer | jmeno text | prijmeni text | rocnik integer | pohlavi text | email text |
|---|---|---|---|---|---|
| 1 | Jan | Novák | 1 | muž | jan.novak@upol.cz |
| 2 | Pavla | Dočkalová | 1 | žena | pavla.dockalova@upol.cz |
| 3 | Karel | Bártek | 2 | muž | karel.bartek@upol.cz |
| 4 | Václav | Snášel | 3 | muž | vaclav.snasel@upol.cz |
| 5 | Iva | Nová | 2 | muž | iva.nova@seznam.cz |
| 5 | Jiří | Malina | 1 | muž | jiri.malina@seznam.cz |

# Příklad

- Vytvoříme klíč, který se skládá ze všech sloupců:

```
ALTER TABLE Studenti
ADD CONSTRAINT vse_key
UNIQUE (id_student,jmeno,prijmeni,rocnik,pohlavi,email);
```

# Superklíč

- Každý řádek z těla relace r je určena hodnotami atributů z K.
- Je klíč vse_key superklíč?

# Kandidátní klíč

- K je **kandidátní klíč** pokud:
  1. K je superklíč.
  2. Žádná vlastní podmnožina K není superklíč.

# Příklad

- Nejdříve odstraníme vse_key:

```
ALTER TABLE Studenti
DROP CONSTRAINT vse_key;
```

- Vytvoříme kandidátní klíč jmeno_key:

```
ALTER TABLE Studenti
ADD CONSTRAINT jmeno_key UNIQUE(jmeno,prijmeni);
```

# Příklad

- Vytvoříme kandidátní klíč id_key:

```
ALTER TABLE Studenti
ADD CONSTRAINT id_key UNIQUE(id_student);
could not create unique index „id_key„
```

- Aktualizujeme řádek:

```
UPDATE studenti
SET id_student=6
WHERE jmeno='Jiří';
```

- Vytvoříme kandidátní klíč id_key.

# Primární klíč

- Jako primární klíč tabulky je vybrán jeden z kandidátních klíčů.
- Ostatní klíče se nazývají alternativní klíče.

- primární klíč:

```
CONSTRAINT c PRIMARY KEY (A1, ..., An)
```

# Příklad

- Odebereme kandidátní klíče id_key:

```
ALTER TABLE studenti
DROP CONSTRAINT id_key;
```

- Vytvoříme primární klíč:

```
ALTER TABLE studenti
ADD CONSTRAINT id_pkey PRIMARY KEY (id_student);
```

- Vytvoříme primární klíč jmeno_key.

```
ALTER TABLE studenti
ADD CONSTRAINT jmeno_pkey PRIMARY KEY (jmeno,prijmeni);
ERROR: multiple primary keys for table „studenti" are not allowed
```

# Příklad

- Při vytvoření tabulky můžeme rovnou přidat do ní primární klíč:

```
CREATE TABLE studenti (
    id_student  INT,
    jmeno TEXT,
    prijmeni TEXT,
    rocnik INT,
    pohlavi TEXT,
    email TEXT,
    CONSTRAINT id_pkey PRIMARY KEY (id_student)
    );
```

# Agregace

# Uspořádání záznamů ve výpisu

- používá se klauzule ORDER BY
- základní syntaxe:

```
SELECT atributy FROM jmeno
[WHERE where_fráze ]
[ORDER BY sloupce[ASC | DESC]]
```

- směr uspořádání
  - ASC – vzestupně (implicitní)
  - DESC – sestupně
- Příklad

```
SELECT * FROM studenti ORDER BY jmeno;
SELECT * FROM studenti ORDER BY prijmeni,jmeno DESC;
```

# Agregační funkce

- slouží k různým matematickým souhrnům
- funkce:
  - COUNT – počet řádků, které v daném atributu nemají NULL
  - MIN – minimální hodnota daného atributu
  - MAX – maximální hodnota daného atributu
  - SUM – součet hodnot atributu přes všechny záznamy
  - AVG – průměr hodnot atributu přes všechny záznamy

# Příklad

- Chceme zjistit počet studentů:

```
SELECT COUNT(id_student)
FROM studenti;
```

- Chceme zjistit počet ženy mezi studenty:

```
SELECT COUNT(id_student) AS pocet_zen
FROM studenti WHERE
pohlavi='zena‘;
```

- Chceme zjistit, kolik celkem získali studenti kreditů:

```
SELECT SUM(kredity) AS kredity_celkem
FROM predmety,zapsani
WHERE id_predmet=predmet AND uspel=TRUE;
```

# Shlukování dat

- používá se klauzule GROUP BY
- kombinuje agregaci s výběrem
- základní syntaxe:

```
SELECT atributy FROM jmeno
[WHERE where_fráze ]
[GROUP BY sloupce]
[HAVING podmínky]
[ORDER BY sloupce[ASC | DESC]]
```

# Příklad

- Chceme zjistit počet studentů v jednotlivých ročnících:

```
SELECT rocnik, COUNT(id_student) AS pocet_studentu
FROM studenti
GROUP BY rocnik;
```

- Chceme jen studenty muže:

```
SELECT rocnik, COUNT(id_student) AS pocet_studentu
FROM studenti
WHERE pohlavi='muž'
GROUP BY rocnik;
```

- Chceme jen ročníky, na kterých studuje alespoň 2 studenti:

```
SELECT rocnik, COUNT(id_student) AS pocet_studentu
FROM studenti
GROUP BY rocnik
HAVING COUNT(id_student)>1;
```

# Bodovaný úkol

- Otevřete soubor datab_05_agregace_ukol.sql.
- Doplňte tyto dotazy:
  - Dotaz, který vypíše knihy a seřadí je podle počtu stran vzestupně.
  - Dotaz, který zobrazí počet knih v databázi, které vyšly před rokem 1989.
  - Dotaz, který vypíše všechny čtenáře, kteří si půjčili více jak jednu knihu.