Agregácia v SQL

Ján Mazák

FMFI UK Bratislava

Agregácia

Niekedy nás nezaujímajú jednotlivé záznamy relácie, ale jedna agregátna hodnota.

```
/* počet zamestnancov */

SELECT COUNT(e.empno)
FROM employee e
```

Agregačné funkcie: COUNT, SUM, AVG, MAX, MIN...

Agregácia

Riadky možno rozdeliť do skupín pomocou GROUP BY.
Na výstupe bude pre každú skupinu 1 riadok.

```
/* počet zamestnancov v jednotlivých oddeleniach */

SELECT e.deptno, COUNT(e.empno) AS c
FROM employee e
GROUP BY e.deptno
HAVING COUNT(e.empno) > 1
```

HAVING umožňuje filtrovať skupiny.
(Pomenovanie atribútov vytvorených agregáciou za SELECT sa nedá použiť za HAVING.)

Agregácia

Mimo dotazu môžu ísť len agregované hodnoty a atribúty, na ktorých sa všetky záznamy v skupine zhodujú (toto je z pohľadu databázy zaručené len vtedy, ak sa ten atribút nachádza za GROUP BY).

```
/* počet zamestnancov v jednotlivých oddeleniach */

SELECT d.name, COUNT(e.empno) AS c
FROM employee e
    JOIN department d ON e.deptno = d.deptno
GROUP BY d.deptno, d.name
```

(Pridanie `d.name` nemá vplyv na rozdelenie riadkov do skupín.)

Agregácia

Do skupín možno deliť aj podľa viacerých atribútov súčasne (riadky v skupine sa musia zhodovať na všetkých).

```
/* počet zamestnancov v skupinách podľa platu
v jednotlivých oddeleniach */

SELECT d.name, d.salary, COUNT(e.empno) AS c
FROM employee e
    JOIN department d ON e.deptno = d.deptno
GROUP BY d.deptno, d.name, e.salary
```

Agregácia

Postupujte opatrne pri aplikovaní agregačných funkcií na NULL a na potenciálne prázdnu množinu záznamov. Výsledky neraz nie sú intuitívne, treba podrobne naštudovať dokumentáciu a overiť správanie pre konkrétny DBMS. Napríklad:

- `COUNT(stĺpec)` ignoruje NULL, ale `COUNT(*)` ich zaráta (aspoň v MySQL)
- AVG pre neprázdnu množinu ignoruje NULL a výsledok tak môže byť veľmi nereprezentatívny; pre prázdnu vráti NULL

WITH — Common Table Expressions (CTE)

WITH vytvorí reláciu existujúcu len počas výpočtu dotazu.

```
WITH pijanPocetAlkoholov(pijan, c) AS (
    SELECT pijan, COUNT(DISTINCT alkohol)
    FROM lubi
)
SELECT MAX(ppa.c)
FROM pijanPocetAlkoholov ppa
```

WITH sa často používa pri viackrokovom agregovaní. V jednom dotaze možno za WITH vymenovať aj viacero relácií oddelených čiarkou.

Záznamy, kde sa dosahuje extrém (arg max)

```
WITH pijanPocetAlkoholov(pijan, c) AS (
    SELECT pijan, COUNT(DISTINCT alkohol)
    FROM lubi
)
SELECT ppa.pijan
FROM pijanPocetAlkoholov ppa
WHERE ppa.c = (
    SELECT MAX(ppa2.c)
    FROM pijanPocetAlkoholov ppa2
)
```

Literatúra

- https://www.postgresqltutorial.com/postgresql-aggregate-functions/
- https://www.postgresqltutorial.com/ (Sections 4, 5, 7)
- https://learnsql.com/blog/error-with-group-by/
- https://www.postgresql.org/docs/current/functions-aggregate.html

Úlohy: SQL

Databáza: *lubi*(Pijan, Alkohol), *capuje*(Krcma, Alkohol, Cena), *navstivil*(Id, Pijan, Krcma), *vypil*(Id, Alkohol, Mnozstvo)

- počet čapovaných alkoholov
- priemerná cena piva
- najdrahší čapovaný alkohol (všetky, ak ich je viac)
- pijan, ktorý vypil najmenej druhov alkoholu
- tržby jednotlivých krčiem
- krčma s najväčšou celkovou tržbou
- priem. suma prepitá pri 1 návšteve pre jednotlivé krčmy
- koľko najviac alkoholov, ktoré nik neľúbi, je v jednej krčme v ponuke?