Agregácia v SQL

Ján Mazák

FMFI UK Bratislava

Množinové operácie

- EXCEPT — rozdiel množín
- EXCEPT ALL — rozdiel množín
- UNION — zjednotenie (bez zachovania duplicity riadkov)
- UNION ALL — zjednotenie (so zachovaním duplicity riadkov)
- INTERSECT — prienik
- INTERSECT ALL — prienik

Množinové operácie

```
SELECT nazov
FROM kriky

UNION

SELECT nazov
FROM stromy
```

```
SELECT rodne_cislo
FROM employee

EXCEPT

SELECT rodne_cislo
FROM customer
WHERE sales > 100
```

Zoradenie záznamov

Zoznam zamestnancov usporiadaný od najvyššieho platu po najnižší, pri rovnakom plate abecedne:

```
SELECT name, salary, department
FROM employee
ORDER BY salary DESC, name ASC
```

Vnorené dotazy (subqueries)

Okrem operátora EXISTS sa možno pomocou operátora IN (resp. NOT IN) pýtať na prítomnosť v množine.

```
SELECT name
FROM employee e
WHERE e.deptno IN (SELECT deptno
                   FROM department d
                   WHERE d.location = 'New York')
```

IN sa optimalizuje horšie ako join a antijoin (NOT EXISTS), dá sa však v prípade potreby prepísať pomocou iných operátorov.

Vnorené dotazy (subqueries)

Ak je výsledkom vnoreného dotazu tabuľka 1x1, možno ju použiť ako skalár.

```
SELECT name
FROM employee e
WHERE e.deptno = (SELECT deptno
                  FROM department d
                  WHERE d.location = 'New York')
```

Ak by vo vnorenom dotaze vyšlo viac riadkov (alebo žiaden), dôjde k chybe.
(Takýmto chybám treba čo najviac predchádzať, napr. použitím UNIQUE už pri vytváraní tabuľky.)

VIEW

Používateľom možno uľahčiť prístup k dátam vytvorením náhľadu (VIEW), ktorý je iný ako samotné definície tabuliek.

```
CREATE VIEW employee_details AS (
    SELECT e.empno, e.name, e.deptno,
           d.name, d.location
    FROM employee e, department d
    WHERE e.deptno = d.deptno
)

SELECT * FROM employee_details
```

VIEW je permanentný objekt v db. Možno cezeň aj vkladať dáta, ale nerobte to (veľa komplikácií s NULL).

Agregácia

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

```
/* pocet 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.

```
/* pocet zamestnancov v jednotlivych 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).

```
/* pocet zamestnancov v jednotlivych 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
- https://www.postgresqltutorial.com/postgresql-views/managing-postgresql-views/
- https://drive.google.com/file/d/1HCq2KMZ05UvtXGe1nTqNmkwhc3X-NLI3/view

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?