LABORATOR 5 SQL

Clauza WITH. Subcereri. Operatori. Cereri cu sincronizare (corelate)

Clauza WITH

- Este utilizată pentru a simplifica scrierea unor cereri complexe.
- Este utilă atunci când o subcerere este utilizată de mai multe ori deoarece subcererea va fi procesată o singură dată.

1. Afişaţi numele departamentelor pentru care suma alocată salariilor depăşeşte valoare medie alocată pe departamente.

Varianta 1

```
WITH
dept_costuri AS (
        SELECT department_name, SUM(salary) dept_cost
        FROM   employees e, departments d
        WHERE  e.department_id= d.department_id
        GROUP BY department_name)
SELECT *
FROM   dept_costuri
WHERE  dept_cost > (select avg(dept_cost) from dept_costuri)
ORDER BY department_name;
```

Varianta 2

```
WITH
dept_costuri AS (
        SELECT department_name, SUM(salary) dept_cost
        FROM   employees e, departments d
        WHERE  e.department_id= d.department_id
        GROUP BY department_name),
medie_cost AS (
        SELECT AVG(dept_cost) medie
        FROM   dept_costuri)
SELECT *
FROM   dept_costuri, medie_cost
WHERE  dept_cost > medie
ORDER BY department_name;
```

2. Daţi o metodă de rezolvare a cererii anterioare fără să utilizaţi clauza WITH. Verificaţi rezultatul obţinut. Comentaţi.

3. Modificaţi cererea anterioară astfel încât să obţineţi acelaşi rezultat ca în cazul punctului 1.

4. Modificaţi cererile de la punctul 1 astfel încât să obţineţi acelaşi rezultat ca în cazul punctului 2.

SUBCERERI

- O subcerere:
  - este o comandă SELECT inclusă într-o clauză a unei alte comenzi SELECT.
  - poate să apară în:
    - clauza WHERE;
    - clauza FROM (subcererile din clauza FROM se mai numesc şi vizualizări inline);
    - clauza HAVING.
    - lista SELECT.
- De exemplu, forma generală a unei cereri ce utilizează subcereri în clauza WHERE este:

```
SELECT lista_select1
FROM   lista_tabele1
WHERE  expresie1 operator (SELECT expresie2
                           FROM   lista_tabele2);
```

- Operatorii pot fi de două tipuri şi trebuie adaptaţi subcererii:
  - de tip single-row (scalari) (>,<,>=, <=, !=); în acest caz subcererea trebuie să întoarcă o singură linie;
  - de tip multiple-row (IN, ANY, ALL); acești operatori sunt folosiţi în combinaţie cu cei single-row; în acest caz subcererea poate întoarce una sau mai multe linii.
- Există şi subcereri care întorc mai mult de o coloană. În acest caz, expresia cu care se face comparaţia trebuie sa aibă aceeaşi formă ca şi rezultatul cererii.

***Observaţii*:**

- Subcererile se specifică între paranteze.
- Este recomandat ca subcererile să apară întotdeauna în dreapta operatorului, pentru lizibilitate. Dar este permisă şi specificarea lor în stânga operatorului.
- De obicei, în cazurile în care nu avem sincronizare se execută mai întâi cererea interioară şi apoi rezultatul acesteia este înglobat în cererea exterioară pentru rezolvarea ei.
- Posibile probleme ce pot apărea la utilizarea subcererilor:
  - Folosirea unui operator single row cu o subcerere ce întoarce mai mult de o linie. (ORA-01427: single-row subquery returns more than one row). *Acţiune*: Se modifică operatorul într-unul multiple-row.

- Folosirea neadecvată a unei subcereri care poate să nu returneze nici o linie. În acest caz nu apar erori dar rezultatul nu este corect (no rows selected).

- Echivalenţe de operatori:
  - `IN <=> = ANY`
  - `NOT IN <=> != ALL`
  - `> ANY <=> > minim`
  - `< ANY <=> < maxim`
  - `> ALL <=> > maxim`
  - `< ALL <=> < minim`

- În cazul în care utilizăm operatorul NOT IN trebuie să ne asiguram ca subcererea să nu returneze valori null. În caz contrar, invariabil, rezultatul cererii va fi 'no rows selected'.

- Subcererile pot fi:
  - fără sincronizare/necorelate: rezultatul subcererii este independent de cererea exterioară şi atunci poate fi evaluat înaintea acesteia; procesarea este similară cu execuţia subcererii şi apoi execuţia cererii exterioare în care se înlocuieşte subcererea cu rezultatul obţinut la primul pas;
  - cu sincronizare/corelate: rezultatul subcererii depinde de o valoare a fiecărei linii întoarse de cererea exterioară; în acest caz, subcererea trebuie evaluată pentru fiecare linie întoarsă de cererea exterioară. Procesarea presupune :
    a) *pas 1* - încărcarea unei linii rezultat a cererii exterioare ignorând condiţia implicată de subcerere;
    b) *pas 2* - execuţia subcererii utilizând valoarea de la pasul 1;
    c) *pas 3* - utilizarea valorilor obţinute la pasul 2 pentru întoarcerea sau nu a liniei rezultat a cererii exterioare;
    d) *pas 4* - repetarea paşilor 1-3 până la epuizarea liniile returnate de cererea exterioară.

- Forma generală a unei cereri cu sincronizare (corelate) ce utilizează subcereri în clauza WHERE este:

```
SELECT t1.coloana1, t1.coloana2, ...
FROM   tabel1 t1
WHERE  expresie operator (SELECT [t2.]coloana1, [t2.]coloana2
                          FROM   tabel2 [t2]
                          WHERE  [t2.]expresie = t1.expresie)
```

5. Obțineți numele primilor 5 angajaţi care au cel mai mare salariul (top 5 angajați în funcție de salariu). Studiați cele două alternative de rezolvare. Rezultatul obținut va fi mereu același indiferent de varianta aleasă?

Varianta1

```
SELECT last_name, job_id, salary
```

```
FROM  employees e
WHERE 5>=(SELECT COUNT(*)
          FROM   employees
          WHERE  salary > e.salary)
ORDER BY salary DESC;
```

Varianta2

```
SELECT  *
FROM  (SELECT last_name, job_id, salary
       FROM   employees
       ORDER BY salary DESC)
WHERE ROWNUM<=5;
```

6. Afișați numele, job-ul şi salariul celor mai prost plătiţi angajaţi din fiecare departament.

Fără sincronizare

```
SELECT last_name, salary, job_id, department_id
FROM   employees
WHERE (department_id, salary) IN (SELECT department_id, MIN(salary)
                                  FROM   employees
                                  GROUP BY department_id);
```

Cu sincronizare

```
SELECT last_name, salary, job_id, department_id
FROM   employees e
WHERE  salary = (SELECT MIN(salary)
                 FROM   employees
                 WHERE  department_id=e.department_id);
```

7. Obțineți pentru fiecare job, numele şi salariul angajaţilor care sunt cel mai bine plătiţi pe jobul respectiv. Rezolvați problema cu sincronizare şi fără sincronizare.

8. Modificați cererea anterioară astfel încât să afișați pentru fiecare job top 3 angajați din punct de vedere al salariului primit.

9. Obţineți codurile şi numele departamentelor în care nu lucrează nimeni. Pentru rezolvare utilizați operatorul NOT IN.

10. Folosind operatorul *ALL*, afişaţi angajaţii care câştigă mai mult decât oricare funcţionar (CLERK). Ce rezultat este obținut dacă se înlocuieşte *ALL* cu *ANY*?

11. Afișați numele şi salariul angajaţilor al căror salariu este mai mare decât salariile medii din toate departamentele. Rezolvați problema în două variante.

OPERATORI PE MULŢIMI

Operatorii pe mulţimi combină rezultatele obţinute din două sau mai multe interogări. Cererile care conţin operatori pe mulţimi se numesc cereri compuse.

Există patru operatori pe mulţimi: UNION, UNION ALL, INTERSECT şi MINUS.

- Operatorul UNION întoarce toate liniile selectate de două cereri, eliminând duplicatele. Acest operator nu ignoră valorile *null*.
- Operatorul UNION ALL întoarce toate liniile selectate de două cereri, fără a elimina duplicatele. Acest operator nu ignoră valorile *null*. În cererile asupra cărora se aplică UNION ALL nu poate fi utilizat cuvântul cheie DISTINCT.
- Operatorul INTERSECT întoarce toate liniile comune cererilor asupra cărora se aplică. Acest operator nu ignoră valorile *null*.
- Operatorul MINUS determină liniile întoarse de prima cerere care nu apar în rezultatul celei de-a doua cereri. Pentru ca operatorul MINUS să funcţioneze, este necesar ca toate coloanele din clauza WHERE să se afle şi în clauza SELECT.

12. Obțineți numărul total de angajaţi ai companiei, respectiv numărul celor care au fost angajaţi în anul 1997. Afișați informațiile cerute în următoarea formă:

a) pe linii (rezultatul va conține două linii și o coloană);

Indicație: Utilizați operatorul UNION.

```
NUMAR
----------------
Numar total: 107
Numar 1997:  28
```

b) pe coloane (rezultatul va conține două coloane și o linie).

```
Numar total          Numar 1997
-------------------- --------------------
107                  28
```

13. Pentru fiecare angajat obțineți următoarele informații despre job-ul prezent, respectiv joburile sale anterioare: numele job-ului, numele departamentului, respectiv data la care a început să lucreze pe job-ul respectiv. Ordonaţi rezultatul după codul angajatului.

14. Folosind operatorul *INTERSECT*, obțineți angajaţii care au salariul cel mult 3000 şi lucrează în departamentul 50.

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary<3000
INTERSECT
```

```
SELECT employee_id, last_name
FROM   employees
WHERE  department_id = 50;
```

15. Obţineți codul, job-ul şi departamentul angajaţilor care în trecut au mai lucrat pe acelaşi job şi în acelaşi departament ca în prezent. Utilizaţi operatorul *INTERSECT*.

16. Modificaţi cererea anterioară astfel încât să obţineţi numele angajaţilor care îndeplinesc condiţia impusă.

17. Afişați codurile departamentelor care nu au angajaţi, implementând operatorul MINUS.

```
SELECT department_id
FROM   departments
MINUS
SELECT DISTINCT department_id
FROM   employees;
```

OPERATORUL BOOLEAN EXISTS

- Operatorul boolean EXISTS aplicat unei subcereri întoarce valoarea *true* dacă subcererea întoarce cel puţin o linie rezultat şi valoarea *false* în caz contrar.

```
SELECT t1.coloana1, t1.coloana2, ...
FROM   tabel1 t1
WHERE [NOT] EXISTS (SELECT 'X'
                    FROM   tabel2 [t2]
                    WHERE  [t2.]expresie = t1.expresie);
```

- Avantajul utilizării operatorului EXISTS este că odată ce subcererea întoarce o linie rezultat, evaluarea acesteia este oprită. Deci, operatorul EXISTS este utilizat atunci când ne interesează numai existenţa unor linii corespondente în subcerere.

18. Determinaţi numele şi codul angajaţilor care câştigă mai mult decât angajatul având codul 200.

Varianta 1 - Forma relaţională

```
SELECT a.employee_id, a.last_name
FROM   employees a, employees b
WHERE  a.salary > b.salary
AND    b.employee_id = 200;
```

Varianta 2 - Forma procedurală

```
SELECT employee_id, last_name
FROM   employees e
WHERE  EXISTS (SELECT 1
```

```
                FROM   employees
                WHERE  employee_id = 200
                AND    e.salary >salary);
```

19. Daţi o altă metodă de rezolvare pentru problema anterioară, utilizând subcereri şi operatorul „>”.

20. Folosind operatorul EXISTS determinaţi numele departamentelor în care lucrează cel puţin un angajat.

```
SELECT department_id, department_name
FROM   departments d
WHERE  EXISTS (SELECT 'x'
               FROM   employees
               WHERE  department_id = d.department_id);
```

21. Daţi o altă metodă de rezolvare pentru problema anterioară, utilizând subcereri şi operatorul *IN*.

TEMĂ

22. Determinați cele mai prost plătite 3 job-uri, din punct de vedere al mediei salariilor.

23. Obţineți top 5 departamente din punct de vedere al numărului de angajaţi.

24. Determinați salariaţii care nu au subordonaţi.

25. Obţineți numele salariaților care cea mai mare vechime în fiecare departament.

26. Rezolvați exercițiul anterior ținând cont de vechimea cumulată în timp (se ține cont și de istoric).

27. Obţineți numele salariaţilor care lucrează într-un departament în care există cel puţin 2 angajaţi cu salariul în grila de salarizare 1.

28. Obţineți codurile angajaţilor care nu au avut joburi anterioare:
- **a)** utilizând operatorul *MINUS*;
- **b)** utilizând operatorul *NOT IN*.

29. Obţineți codul, job-ul şi departamentul angajaţilor care în trecut au lucrat pe alte joburi sau în alte departamente faţă de prezent. Utilizaţi operatorul *MINUS*.

30. **a)** Determinați codurile locaţiilor în care nu există departamente. Utilizaţi operatorul *MINUS*.
- **b)** Daţi o altă metodă de rezolvare.
- **c)** Modificați cererile anterioare astfel încât să obțineți orașele în care nu funcționează departamente.

31. Folosind operatorul *EXISTS* determinaţi codul şi numele departamentelor în care nu lucrează nimeni.

32. Afişaţi codul locaţiei şi oraşul în care nu funcţionează departamente, utilizând:
1. *NOT IN*;
2. *MINUS*;
3. *NOT EXISTS*;

4. *Outer Join*.

33. Determinaţi numele angajaţilor care au lucrat cel puţin la aceleaşi proiecte ca şi angajatul având codul 202 (au lucrat la toate proiectele la care a lucrat angajatul 202 şi eventual la alte proiecte).
***Observaţie*: $A \subseteq B \Leftrightarrow A \setminus B = \emptyset$**

34. Determinaţi numele angajaţilor care au lucrat cel mult la aceleaşi proiecte ca şi angajatul având codul 202.

35. Determinaţi numele angajaţilor care au lucrat exact la aceleaşi proiecte ca şi angajatul având codul 202.
***Observaţie*: $A = B \Leftrightarrow A \setminus B = \emptyset$ şi $B \setminus A = \emptyset$**