

LABORATOR 5 SQL

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

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. Afișați numele departamentelor pentru care suma alocată salariilor depășește 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. Dați o metodă de rezolvare a cererii anterioare fără să utilizați clauza WITH. Verificați rezultatul obținut. Comentați.
3. Modificați cererea anterioară astfel încât să obțineți același rezultat ca în cazul punctului 1.
4. Modificați cererile de la punctul 1 astfel încât să obțineți același 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 și 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 și trebuie adaptați 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 folosiți în combinație cu cei single-row; în acest caz subcererea poate întoarce una sau mai multe linii.
- Există și subcereri care întorc mai mult de o coloană. În acest caz, expresia cu care se face comparația trebuie să aibă aceeași formă ca și rezultatul cererii.

Observații:

- Subcererile se specifică între paranteze.
- Este recomandat ca subcererile să apară întotdeauna în dreapta operatorului, pentru lizibilitate. Dar este permisă și specificarea lor în stânga operatorului.
- De obicei, în cazurile în care nu avem sincronizare se execută mai întâi cererea interioară și 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). *Acțiune:* 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).
- Echivalențe 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 asigurăm 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 și atunci poate fi evaluat înaintea acesteia; procesarea este similară cu execuția subcererii și apoi execuția cererii exterioare în care se înlocuiește subcererea cu rezultatul obținut 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 condiția implicată de subcerere;
 - b) *pas 2* - execuția subcererii utilizând valoarea de la pasul 1;
 - c) *pas 3* - utilizarea valorilor obținute la pasul 2 pentru întoarcerea sau nu a liniei rezultat a cererii exterioare;
 - d) *pas 4* - repetarea pașilor 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 angajați care au cel mai mare salariu (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. Obțineți top 5 departamente din punct de vedere al numărului de angajați.
7. Determinați cele mai prost plătite 3 job-uri, din punct de vedere al mediei salariilor.
8. Afișați numele, job-ul și salariul celor mai prost plătiți angajați 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);
```

9. Obțineți pentru fiecare job, numele și salariul angajaților care sunt cel mai bine plătiți pe jobul respectiv. Rezolvați problema cu sincronizare și fără sincronizare.
10. Modificați cererea anterioară astfel încât să afișați pentru fiecare job top 3 angajați din punct de vedere al salariului primit.
11. Obțineți codurile și numele departamentelor în care nu lucrează nimeni. Pentru rezolvare utilizați operatorul NOT IN.
12. Determinați salariații care nu au subordonați.

13. Folosind operatorul *ALL*, afișați angajații care câștigă mai mult decât oricare funcționar (CLERK). Ce rezultat este obținut dacă se înlocuiește *ALL* cu *ANY*?
14. Afișați numele și salariul angajaților al căror salariu este mai mare decât salariile medii din toate departamentele. Rezolvați problema în două variante.
15. Obțineți numele salariaților care cea mai mare vechime în fiecare departament.
16. Rezolvați exercițiul anterior ținând cont de vechimea cumulată în timp (se ține cont și de istoric).
17. Obțineți numele salariaților care lucrează într-un departament în care există cel puțin 2 angajați cu salariul în grila de salarizare 1.

OPERATORI PE MULȚIMI

Operatorii pe mulțimi combină rezultatele obținute din două sau mai multe interogări. Cererile care conțin operatori pe mulțimi se numesc cereri compuse.

Există patru operatori pe mulțimi: UNION, UNION ALL, INTERSECT și 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 funcționeze, este necesar ca toate coloanele din clauza WHERE să se afle și în clauza SELECT.
18. Obțineți numărul total de angajați ai companiei, respectiv numărul celor care au fost angajați 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

```

19. Folosind operatorul UNION afișați codul departamentelor, codul job-urilor și valoarea medie a salariilor, pentru:
- fiecare departament și, în cadrul său pentru fiecare job;
 - fiecare departament (indiferent de job);
 - fiecare job(indiferent de departament);
 - întreg tabelul.

20. 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. Ordonați rezultatul după codul angajatului.

21. Folosind operatorul *INTERSECT*, obțineți angajații care au salariul cel mult 3000 și 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;
```

22. Obțineți codul, job-ul și departamentul angajaților care în trecut au mai lucrat pe același job și în același departament ca în prezent. Utilizați operatorul *INTERSECT*.

23. Modificați cererea anterioară astfel încât să obțineți numele angajaților care îndeplinesc condiția impusă.

24. Afișați codurile departamentelor care nu au angajați, implementând operatorul *MINUS*.

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

25. Obțineți codurile angajaților care nu au avut joburi anterioare:

- a) utilizând operatorul *MINUS*;
- b) utilizând operatorul *NOT IN*.

26. Obțineți codul, job-ul și departamentul angajaților care în trecut au lucrat pe alte joburi sau în alte departamente față de prezent. Utilizați operatorul *MINUS*.

27. a) Determinați codurile locațiilor în care nu există departamente. Utilizați operatorul *MINUS*.

- b) Dați o altă metodă de rezolvare.

- c) Modificați cererile anterioare astfel încât să obțineți orașele în care nu funcționează departamente.