LABORATOR 5 SQL

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

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.

1. 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);
```

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

3. 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);
```

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

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

6. Afişaţi codul locaţiei şi oraşul în care nu funcţionează departamente, utilizând:
a) *NOT IN*;
b) *MINUS*;
c) *NOT EXISTS*;
d) *Outer Join*.

7. 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 ⊆ B ⇔ A \ B = Ø**

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

9. Determinaţi numele angajaţilor care au lucrat exact la aceleaşi proiecte ca şi angajatul având codul 202.
***Observaţie*: A = B ⇔ A \ B = Ø şi B \ A = Ø**

OPERATORUL DIVISION

10. Afişaţi codul şi numele proiectelor la care au lucrat toţi angajaţii din departamentul 20.

Varianta 1

```
SELECT  p.project_id, project_name
FROM    projects p, work w
WHERE   p.project_id=w.project_id
AND     employee_id IN (SELECT employee_id
                        FROM   employees
                        WHERE  department_id =20)
GROUP BY p.project_id, project_name
HAVING COUNT(*)=(SELECT COUNT(*)
                 FROM   employees
                 WHERE  department_id =20);
```

Varianta 2

```
SELECT DISTINCT p.project_id, project_name
FROM   projects p, work w
WHERE  p.project_id=w.project_id
AND NOT EXISTS (SELECT 'X'
                FROM   employees e
```

```
WHERE  department_id=20
AND NOT EXISTS (SELECT 'X'
                FROM   work w1
                WHERE  e.employee_id=w1.employee_id
                AND    w.project_id=w1.project_id));
```

11. Afişaţi codul angajaţilor care au lucrat la toate proiectele care au început în anul 1999.

CERERI IERARHICE

- Clauzele *START WITH* şi *CONNECT BY* se utilizează în formularea cererilor ierarhice.
  - *START WITH* specifică o condiţie care identifică liniile ce urmează să fie considerate ca rădăcini ale cererii ierarhice respective. Dacă se omite această clauză, sistemul *Oracle* utilizează toate liniile din tabel drept linii rădăcină.
  - Clauza *CONNECT BY* specifică o condiţie care identifică relaţia dintre liniile „părinte“ şi „copil“ ale ierarhiei. Condiţia trebuie să conţină operatorul *PRIOR* pentru a face referinţă la linia „părinte“.
- Interogările ierarhice permit regăsirea datelor pe baza unei relaţii ierarhice care există între liniile tabelului. Relaţia „părinte-copil“ a unei structuri arborescente permite controlul direcţiei în care este parcursă ierarhia şi stabilirea rădăcinii ierarhiei.
- Pseudocoloana *LEVEL* poate fi utilă într-o cerere ierarhică. Aceasta determină lungimea drumului de la rădăcină la un nod.
- Operatorul *PRIOR* face referinţă la linia „părinte“. Plasarea acestui operator determină direcţia interogării, dinspre „părinte“ spre „copil“ (*top-down*) sau invers (*bottom-up*)
  - CONNECT BY PRIOR cheie_parinte = cheie_copil (top-down);
  - CONNECT BY cheie_copil = PRIOR cheie_parinte (*bottom-up*);
- Liniile „părinte“ ale interogării sunt identificate prin clauza *START WITH*. Pentru a găsi liniile „copil“, *server*-ul evaluează expresia din dreptul operatorului *PRIOR* pentru linia „părinte“, şi cealaltă expresie pentru fiecare linie a tabelului. Înregistrările pentru care condiţia este adevărată vor fi liniile „copil“. Spre deosebire de *START WITH*, în clauza *CONNECT BY* nu pot fi utilizate subcereri.
- În tabelul *employees*, se poate imagina o structură arborescentă pe baza managerilor (coloana *manager_id*). Un angajat are un manager, la rândul său managerul are şi el un manager etc.

12. Obţineţi ierarhia subaltern-şef. Se vor afişa codul şi numele angajatului însoţit de codul managerului său. De asemenea, se va afişa şi nivelul din ierarhie.

```
SELECT LEVEL, employee_id, last_name, manager_id
FROM   employees
CONNECT BY PRIOR manager_id = employee_id;
```

```
LEVEL    EMPLOYEE_ID     LAST_NAME       MANAGER_ID
-------------------------------------------------
1        100             King
-------------------------------------------------
1        101             Kochhar         100
2        100             King
-------------------------------------------------
1        102             De Haan         100
2        100             King
-------------------------------------------------
1        103             Hunold          102
2        102             De Haan         100
3        100             King
-------------------------------------------------
....
```

13. Obţineţi ierarhia subaltern-şef, considerând ca rădăcină angajatul având codul 103. Se vor afişa codul şi numele angajatului însoţit de codul managerului său. De asemenea, se va afişa şi nivelul din ierarhie.
Observaţie: În relaţia ierarhică părinte este coloana *manager_id*.

```
SELECT LEVEL, employee_id, last_name, manager_id
FROM   employees
START  WITH employee_id =103
CONNECT BY PRIOR manager_id = employee_id;
```

```
LEVEL    EMPLOYEE_ID     LAST_NAME       MANAGER_ID
-------------------------------------------------
1        103             Hunold          102
2        102             De Haan         100
3        100             King
```

14. Obţineţi ierarhia şef-subaltern, considerând ca rădăcină angajatul având codul 103. Se vor afişa codul şi numele angajatului însoţit de codul managerului său.
Observaţie: În relaţia ierarhică părinte este coloana *employee_id*.

15. Obţineţi ierarhia subaltern-şef, considerând ca rădăcină angajatul al cărui salariu este minim. Se vor afişa codul şi numele angajatului însoţit de codul managerului său.

16. Afişaţi ierarhia subaltern-şef, considerând ca rădăcină angajatul al cărui cod este 206. Să se afişeze codul, numele şi salariul angajatului însoţit de codul managerului său, pentru angajaţii al căror salariu este mai mare decât 15000.

Verificaţi şi comentaţi rezultatul obţinut în cazul în care:

a) condiţia *salary* > 15000 apare în clauza WHERE;

b) condiţia *salary* > 15000 apare în clauza CONNECT BY.

Verificaţi şi comentaţi rezultatul obţinut în cazul în care condiţia impusă ar fi fost *salary* < 15000.

17. Afişaţi codul, numele, data angajării, salariul şi managerul pentru:

a) subalternii direcţi ai lui De Haan;

b) şeful direct al lui De Haan;

c) ierarhia şef-subaltern care începe de la De Haan;

d) angajaţii conduşi de subalternii lui De Haan;

e) ierarhia subaltern-şef pentru Hunold;

f) superiorul şefului direct al lui Hunold.

18. Câţi şefi pe linie ierarhică are angajatul 107?

19. Obţineţi ierarhia subaltern-şef pentru toţi managerii de departamente.

20. Pentru fiecare angajat determinaţi nivelul său ierarhic în companie.

21. Obţineţi nivelul ierarhic în companie al şefilor de departamente.

22. Pentru angajatul având codul 206, afişaţi o structură arborescentă în care va apărea angajatul, şeful său, superiorul şefului său etc. Coloanele afişate vor fi: codul angajatului, codul managerului, nivelul în ierarhie (LEVEL) şi numele angajatului.

```
SET LINESIZE 100
COLUMN nume FORMAT a25;

SELECT employee_id, manager_id, LEVEL, last_name,
       LPAD(last_name, length(last_name)+level*2-2, '_') nume
FROM   employees
START WITH employee_id=206
CONNECT BY employee_id=PRIOR manager_id;
```

23. Creaţi un raport din care să reiasă structura ierarhică şef-subordonat.

```
SET LINESIZE 100
SET PAGESIZE 100
COLUMN nume FORMAT a25;

SELECT LPAD(' ', 3*LEVEL-3)||last_name nume, LEVEL,
       employee_id, manager_id, department_id
FROM   employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id= manager_id;
```

24. a) Modificaţi cererea anterioară astfel încât din rezultat să fie exclus angajatul De Haan, dar nu şi subordonaţii săi.

Observaţie: Pentru aceasta se include condiţia într-o clauză *WHERE*.

b) Modificaţi cererea anterioară astfel încât din rezultat să fie exclus angajatul De Haan împreună cu subordonaţii săi.

Observaţie: Pentru aceasta se include condiţia în clauza *CONNECT BY*.

25. Afişaţi ierarhia şef-subaltern: codul, prenumele şi numele (pe aceeaşi coloană), codul job-ului şi data angajării, pornind de la subordonaţii direcţi ai lui Steven King care au cea mai mare vechime. Rezultatul nu va conţine angajaţii în anul 1970.

```
WITH emp_sk AS
 (SELECT   employee_id, hire_date
  FROM     employees
  WHERE    manager_id = (SELECT employee_id
                         FROM   employees
                         WHERE  INITCAP(last_name) = 'King'
                         AND    INITCAP(first_name) = 'Steven'))
   SELECT  employee_id, INITCAP(first_name) ||' '||UPPER(last_name),
           job_id, hire_date,manager_id
   FROM    employees
   WHERE   TO_CHAR(hire_date, 'yyyy') != 1970
   START WITH employee_id  IN
                (SELECT   employee_id
                 FROM     emp_sk
                 WHERE    hire_date = (SELECT   MIN(hire_date)
                                       FROM     emp_sk))
   CONNECT BY PRIOR employee_id = manager_id;
```

26. Afişaţi textul *true* dacă în tabelul *job_grades* minimul marginii superioare a grilelor de salarizare este mai mic decât 10000. Este necesară utilizarea clauzei GROUP BY?

```
SELECT 'true'
FROM    job_grades
HAVING  MIN(highest_sal)<10000;
```