

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 puțin o linie rezultat și 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 existența unor linii corespondente în subcerere.

1. Determinați numele și codul angajaților care câștigă mai mult decât angajatul având codul 200.

Varianta 1 - Forma relațională

```
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. Dați o altă metodă de rezolvare pentru problema anterioară, utilizând subcereri și operatorul „>”.
3. Folosind operatorul EXISTS determinați numele departamentelor în care lucrează cel puțin un angajat.

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

4. Dați o altă metodă de rezolvare pentru problema anterioară, utilizând subcereri și operatorul *IN*.
5. Folosind operatorul *EXISTS* determinați codul și numele departamentelor în care nu lucrează nimeni.
6. Afișați codul locației și orașul în care nu funcționează departamente, utilizând:
 - a) *NOT IN*;
 - b) *MINUS*;
 - c) *NOT EXISTS*;
 - d) *Outer Join*.
7. Determinați numele angajaților care au lucrat cel puțin la aceleași proiecte ca și angajatul având codul 202 (au lucrat la toate proiectele la care a lucrat angajatul 202 și eventual la alte proiecte).

Observație: $A \subseteq B \Leftrightarrow A \setminus B = \emptyset$

8. Determinați numele angajaților care au lucrat cel mult la aceleași proiecte ca și angajatul având codul 202.
9. Determinați numele angajaților care au lucrat exact la aceleași proiecte ca și angajatul având codul 202.

Observație: $A = B \Leftrightarrow A \setminus B = \emptyset$ și $B \setminus A = \emptyset$

OPERATORUL DIVISION

10. Afișați codul și numele proiectelor la care au lucrat toți angajații 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. Afișați codul angajaților care au lucrat la toate proiectele care au început în anul 1999.

CERERI IERARHICE

- Clauzele *START WITH* și *CONNECT BY* se utilizează în formularea cererilor ierarhice.
 - *START WITH* specifică o condiție 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 condiție care identifică relația dintre liniile „părinte” și „copil” ale ierarhiei. Condiția trebuie să conțină operatorul *PRIOR* pentru a face referință la linia „părinte”.
 - Interogările ierarhice permit regăsirea datelor pe baza unei relații ierarhice care există între liniile tabelului. Relația „părinte-copil” a unei structuri arborescente permite controlul direcției în care este parcursă ierarhia și 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 referință la linia „părinte”. Plasarea acestui operator determină direcția 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”, și cealaltă expresie pentru fiecare linie a tabelului. Înregistrările pentru care condiția 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 și el un manager etc.
- 12.** Obțineți ierarhia subaltern-șef. Se vor afișa codul și numele angajatului însoțit de codul managerului său. De asemenea, se va afișa și 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. Obțineți ierarhia subaltern-șef, considerând ca rădăcină angajatul având codul 103. Se vor afișa codul și numele angajatului însoțit de codul managerului său. De asemenea, se va afișa și nivelul din ierarhie.

Observație: În relația 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. Obțineți ierarhia șef-subaltern, considerând ca rădăcină angajatul având codul 103. Se vor afișa codul și numele angajatului însoțit de codul managerului său.

Observație: În relația ierarhică părinte este coloana *employee_id*.

15. Obțineți ierarhia subaltern-șef, considerând ca rădăcină angajatul al cărui salariu este minim. Se vor afișa codul și numele angajatului însoțit de codul managerului său.

16. Afișați ierarhia subaltern-șef, considerând ca rădăcină angajatul al cărui cod este 206. Să se afișeze codul, numele și salariul angajatului însoțit de codul managerului său, pentru angajații al căror salariu este mai mare decât 15000.

Verificați și comentați rezultatul obținut în cazul în care:

- a) condiția *salary > 15000* apare în clauza **WHERE**;
- b) condiția *salary > 15000* apare în clauza **CONNECT BY**.

Verificați și comentați rezultatul obținut în cazul în care condiția impusă ar fi fost *salary < 15000*.

17. Afișați codul, numele, data angajării, salariul și managerul pentru:
- a) subalternii direcți ai lui De Haan;
 - b) șeful direct al lui De Haan;
 - c) ierarhia șef-subaltern care începe de la De Haan;
 - d) angajații conduși de subalternii lui De Haan;
 - e) ierarhia subaltern-șef pentru Hunold;
 - f) superiorul șefului direct al lui Hunold.
18. Câți șefi pe linie ierarhică are angajatul 107?
19. Obțineți ierarhia subaltern-șef pentru toți managerii de departamente.
20. Pentru fiecare angajat determinați nivelul său ierarhic în companie.
21. Obțineți nivelul ierarhic în companie al șefilor de departamente.
22. Pentru angajatul având codul 206, afișați o structură arborescentă în care va apărea angajatul, șeful său, superiorul șefului său etc. Coloanele afișate vor fi: codul angajatului, codul managerului, nivelul în ierarhie (LEVEL) și 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. Creați un raport din care să reiasă structura ierarhică șef-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) Modificați cererea anterioară astfel încât din rezultat să fie exclus angajatul De Haan, dar nu și subordonații săi.

Observație: Pentru aceasta se include condiția într-o clauză *WHERE*.

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

Observație: Pentru aceasta se include condiția în clauza *CONNECT BY*.

25. Afișați ierarhia șef-subaltern: codul, prenumele și numele (pe aceeași coloană), codul job-ului și data angajării, pornind de la subordonații direcți ai lui Steven King care au cea mai mare vechime. Rezultatul nu va conține angajații 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. Afișați 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;
```