

Baze de date

Laborator 6

Subcereri nesincronizate (necorelate)

I. [Subcereri]

O subcerere este o **comandă SELECT** încapsulată într-o clauză a altei instrucțiuni SQL, numită instrucțiune „părinte”. Utilizând subcereri, se pot construi interogări complexe pe baza unor instrucțiuni simple. Subcererile mai sunt numite instrucțiuni **SELECT** imbricate sau interioare.

Subcererea returnează o valoare care este utilizată de către instrucțiunea „părinte”. Utilizarea unei subcereri este echivalentă cu efectuarea a două cereri secvențiale și utilizarea rezultatului cererii interne ca valoare de căutare în cererea externă (principală).

Subcererile sunt de 2 tipuri:

➤ **Necorelate** (nesincronizate), de forma:

```
SELECT   lista_select
FROM     nume_tabel
WHERE     expresie operator ( SELECT lista_select
                                FROM   nume_tabel);
```

- cererea internă este executată prima și determină o valoare (sau o mulțime de valori);
- cererea externă se execută o singură dată, utilizând valorile returnate de cererea internă.

➤ **Corelate** (sincronizate), de forma:

```
SELECT nume_coloană_1[, nume_coloană_2 ...]
FROM   nume_tabel_1 extern
WHERE   expresie operator
         ( SELECT nume_coloană_1 [, nume_coloană_2 ...]
           FROM   nume_tabel_2
           WHERE   expresie_1 = extern.expresie_2);
```

- cererea externă determină o linie candidat;
- cererea internă este executată utilizând valoarea liniei candidat;
- valorile rezultate din cererea internă sunt utilizate pentru calificarea sau descalificarea liniei candidat;
- pașii precedenți se repetă până când nu mai există linii candidat.

Obs: **operator** poate fi:

- **single-row operator** (>, =, >=, <, <=, <>), care poate fi utilizat dacă subcererea returnează **o singură linie**;
- **multiple-row operator** (IN, ANY, ALL), care poate fi folosit dacă subcererea returnează **mai mult de o linie**.

Operatorul **NOT** poate fi utilizat în combinație cu **IN**, **ANY** și **ALL**.

II. [Exercitii - subcereri necorelate]

1. Folosind subcereri, să se afișeze **numele** și **data angajării** pentru salariații care au fost angajați după Gates.

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date > (SELECT hire_date
                   FROM employees
                   WHERE INITCAP(last_name)='Gates');
```

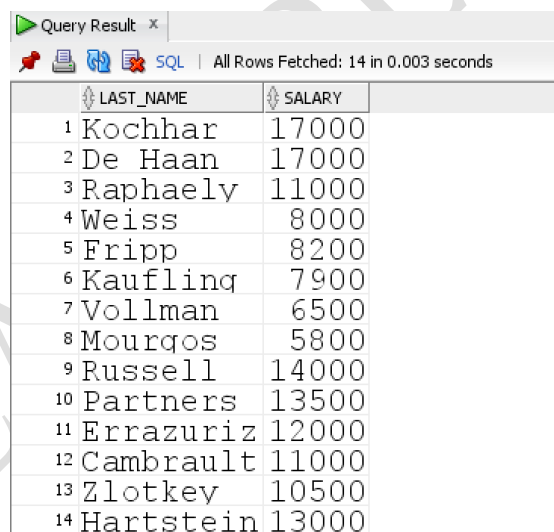
2. Folosind subcereri, scrieți o cerere pentru a afișa **numele** și **salariul** pentru toți colegii (din același departament) lui Gates. Se va exclude Gates.

Se poate înlocui operatorul IN cu = ???

Se va înlocui Gates cu King

3. Folosind subcereri, să se afișeze **numele** și **salariul** angajaților conduși direct de președintele companiei (acesta este considerat angajatul care nu are manager).

Cererea trebuie să returneze 14 angajați, după cum urmează:



	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Raphaely	11000
4	Weiss	8000
5	Fripp	8200
6	Kaufling	7900
7	Vollman	6500
8	Mourgos	5800
9	Russell	14000
10	Partners	13500
11	Errazuriz	12000
12	Cambrault	11000
13	Zlotkey	10500
14	Hartstein	13000

4. Scrieți o cerere pentru a afișa **numele**, **codul departamentului** și **salariul** angajaților al căror cod de departament și salariu coincid cu codul departamentului și salariul unui angajat care câștigă comision.

```
SELECT last_name, department_id, salary
FROM employees
WHERE (department_id, salary) IN (SELECT department_id, salary
                                   FROM employees
                                   WHERE commission_pct is not null);
```

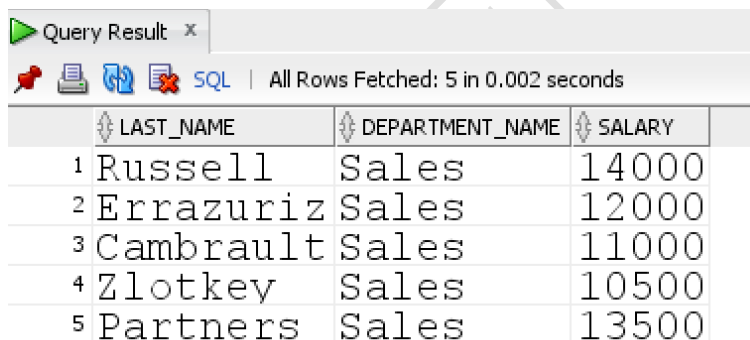
5. Să se afișeze **codul, numele și salariul** tuturor angajaților al căror salariu este mai mare decât salariul mediu.

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary)
                FROM employees);
```

6. Scrieti o cerere pentru a afișa angajații care câștigă (castiga = salariul plus commission din salariu) mai mult decât **oricare** funcționar (job-ul conține șirul "CLERK"). Sortați rezultatele după salariu, în ordine descrescătoare.

7. Scrieți o cerere pentru a afișa **numele angajaților, numele departamentului și salariul angajaților** care câștigă comision, dar al căror șef direct nu câștigă comision.

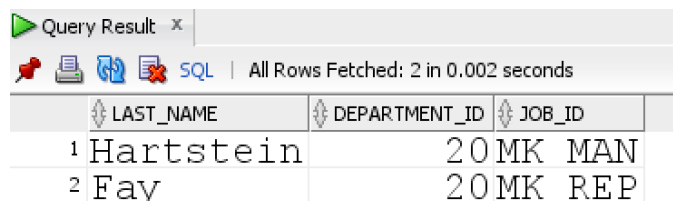
Cererea trebuie sa returneze 5 angajati, dupa cum urmeaza:



	LAST_NAME	DEPARTMENT_NAME	SALARY
1	Russell	Sales	14000
2	Errazuriz	Sales	12000
3	Cambault	Sales	11000
4	Zlotkey	Sales	10500
5	Partners	Sales	13500

8. Să se afișeze **numele angajaților, codul departamentului și codul job-ului** salariaților al căror departament se află în Toronto.

Cererea trebuie sa returneze 2 angajati, dupa cum urmeaza:



	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Hartstein	20	MAN
2	Fay	20	REP

9. Să se obțină **codurile departamentelor** în care nu lucreaza nimeni (nu este introdus niciun salariat în tabelul employees). Sa se utilizeze subcereri. De ce este nevoie de utilizarea funcției NVL?

10. Este posibilă introducerea de înregistrări prin intermediul subcererilor (specificate în locul tabelului). Ce reprezintă, de fapt, aceste subcereri? Să se analizeze următoarele comenzi **INSERT**:

```
INSERT INTO emp_pnu (employee_id, last_name, email, hire_date, job_id, salary,
                    commission_pct)
VALUES (252, 'Nume252', 'nume252 @emp.com', SYSDATE, 'SA_REP', 5000, NULL);
```

```
SELECT employee_id, last_name, email, hire_date, job_id, salary, commission_pct
FROM emp_pnu
WHERE employee_id = 252;
```

ROLLBACK;

```
INSERT INTO
    (SELECT employee_id, last_name, email, hire_date, job_id, salary,
     commission_pct
     FROM emp_pnu)
VALUES (252, 'Nume252', 'nume252 @emp.com', SYSDATE, 'SA_REP', 5000, NULL);
```

```
SELECT employee_id, last_name, email, hire_date, job_id, salary, commission_pct
FROM emp_pnu
WHERE employee_id = 252;
```

ROLLBACK;

11. Să se creeze tabelul **SUBALTERNI_PNU** care să conțină codul, numele și prenumele angajaților care îl au manager pe Steven King, alături de codul și numele lui King. Coloanele se vor numi cod, nume, prenume, cod_manager, nume_manager.