Baze de date-Anul 1
Laborator 5

Subcereri corelate.
Analiza top-n.
Clauza WITH.

I. [Subcereri corelate (sincronizate)]

O subcerere (cerere imbricată) corelată poate avea forma următoare:

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*);

Modul de execuţie este următorul :

- 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;
- paşii precedenţi 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.

Obs:
Probleme ce pot apărea la utilizarea subcererilor:

- Folosirea unui operator single row cu o subcerere ce returnează mai mult de o linie. (ORA01427: single-row subquery returns more than one row). Pentru rezolvarea erorii 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 echivalent cu =ANY ;
- NOT IN echivalent cu !=ALL ;
- > ANY echivalent cu mai mare ca minimul ;
- < ANY echivalent cu mai mic decât maximul ;
- > ALL echivalent cu mai mare decât maximul ;
- < ALL echivalent cu mai mic decât minimul ;
- În cazul în care utilizăm NOT IN trebuie avut grijă ca subcererea să nu returneze valori null. În caz contrar, invariabil, rezultatul cererii va fi 'no rows selected'.

Obs: O subcerere (corelată sau necorelată) poate apărea în clauzele:

- SELECT
- FROM (vezi laboratorul 4)
- WHERE
- HAVING (vezi laboratorul 4)

***Operatorul* EXISTS**

- În instrucţiunile *SELECT* imbricate, este permisă utilizarea oricărui operator logic.
- Pentru a testa dacă valoarea recuperată de **cererea externă (cererea părinte)** există în mulţimea valorilor regăsite de cererea internă corelată, se poate utiliza operatorul ***EXISTS***. Dacă subcererea returnează cel puţin o linie, operatorul returnează valoarea *TRUE*. În caz contrar, va fi returnată valoarea *FALSE*.
- Operatorul *EXISTS* asigură că **nu mai este** continuată căutarea în cererea internă după ce aceasta regăseşte o linie.

Exerciţii:

0. a) Să se afişeze **informaţii** despre angajaţii al căror **salariu depăşeşte valoarea medie** a salariilor colegilor săi de departament.

```
SELECT   last_name, salary, department_id
FROM     employees e
WHERE    salary > (SELECT   AVG(salary)
                   FROM     employees
                   WHERE    department_id = e.department_id);
```

b) Analog cu cererea precedentă, afişându-se şi numele departamentului şi media salariilor acestuia şi numărul de angajaţi.

Soluţia 1 (subcerere necorelată în clauza FROM):

```
SELECT   last_name, salary, e.department_id, department_name, sal_med, nr_sal
FROM     employees e, departments d, (SELECT department_id, AVG(salary) sal_med,
                                             COUNT(*) nr_sal
                                     FROM     employees
                                     GROUP BY department_id) ac
WHERE    e.department_id = d.department_id
AND      d.department_id = ac.department_id
AND      salary > (SELECT   AVG(salary)
                   FROM     employees
                   WHERE    department_id = e.department_id);
```

Soluţia 2 (subcerere corelată în clauza SELECT):

```
SELECT   last_name, salary, e. department_id, department_name,
        (SELECT AVG(salary)
         FROM    employees
         WHERE department_id = e. department_id) “Salariu mediu”,
        (SELECT COUNT(*)
         FROM    employees
         WHERE department_id = e. department_id) “Nr angajati”
FROM     employees e join departments d on (e.department_id = d.department_id)
WHERE  salary > (SELECT   AVG(salary)
                 FROM      employees
                 WHERE    department_id = e.department_id);
```

1. Să se afişeze **numele** şi **salariul angajaţilor** al căror salariu este mai mare decât salariile medii din toate departamentele. Se cer 2 variante de rezolvare: cu **operatorul *ALL*** sau cu **funcţia *MAX***.

2. Sa se afiseze **numele** si **salariul** celor mai **prost platiti** angajati din fiecare departament.

Soluţia 1 (cu sincronizare):

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

Soluţia 2 (fără sincronizare):

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

Soluţia 3: Subcerere în clauza FROM

3. Sa se obtina **numele salariatilor** care lucreaza intr-un departament in care **exista cel putin 1** angajat cu salariul egal cu salariul maxim din departamentul 30.

Obs: Deoarece nu este necesar ca instrucţiunea *SELECT* interioară să returneze o anumită valoare, se poate selecta o constantă ('x', '', 1 etc.). De altfel, din punct de vedere al performanţei, selectarea unei constante asigură mai multă rapiditate decât selectarea unei coloane.

4. Sa se obtina **numele primilor 3 angajati** avand salariul maxim. Rezultatul se va afişa în ordine crescătoare a salariilor.

Solutia 1: subcerere sincronizată
Solutia 2: vezi analiza top-n (mai jos)

5. Să se afişeze **codul**, **numele** şi **prenumele** angajaţilor care au **cel puţin doi** subalterni.

6. Să se determine **locaţiile** în care se află **cel puţin** un departament.

Obs: Ca alternativă a lui ***EXISTS***, poate fi utilizat operatorul ***IN***. Scrieţi şi această variantă de rezolvare.

7. Să se determine **departamentele** în care nu există nici un angajat.

```
SELECT   department_id, department_name
FROM     departments d
WHERE    NOT EXISTS (SELECT   'x'
                     FROM     employees
                     WHERE    department_id = d.department_id);
```

Obs: Acest exemplu poate fi rezolvat şi printr-o subcerere necorelată, utilizând operatorul *NOT IN* (vezi şi laboratorul 3). Atenţie la valorile NULL! Scrieţi şi această variantă de rezolvare.

II. [Clauza WITH]

- Cu ajutorul clauzei ***WITH*** se poate defini un bloc de cerere înainte ca acesta să fie utilizat într-o interogare.
- Clauza permite reutilizarea aceluiaşi bloc de cerere într-o instrucţiune *SELECT* complexă. Acest lucru este util atunci când o cerere face referinţă de mai multe ori la acelaşi bloc de cerere, care conţine operaţii *join* şi funcţii agregat.

Exerciţii:

8. Utilizând **clauza *WITH***, să se scrie o cerere care afişează **numele departamentelor** şi **valoarea totală a salariilor** din cadrul acestora. Se vor considera departamentele a căror valoare totală a salariilor este mai mare decât media valorilor totale ale salariilor tuturor angajatilor.

```
WITH val_dep AS (SELECT department_name, SUM(salary) AS total
        FROM departments d join employees e ON (d.department_id = e.department_id)
        GROUP BY department_name
        ),

val_medie  AS (SELECT SUM(total)/COUNT(*) AS medie
        FROM val_dep)

SELECT *
FROM val_dep
WHERE total > (SELECT medie
        FROM val_medie)
ORDER BY department_name;
```

9. Să se afişeze **codul**, **prenumele**, **numele** şi **data angajării**, pentru angajatii condusi de Steven King care au cea mai mare vechime dintre subordonatii lui Steven King. Rezultatul nu va conţine angajaţii din anul 1970.

III. [Analiza top-n]

Pentru aflarea primelor n rezultate ale unei cereri, este utilă pseudocoloana ***ROWNUM***. Aceasta returnează numărul de ordine al unei linii în rezultat.

Exerciţii:

10. Să se determine **primii 10** cei mai bine plătiţi angajaţi.

IV. [Exerciţii – utilizarea alternativă a funcţiei DECODE sau a structurii CASE; din nou NVL şi NVL2; COALESCE; NULLIF]

Obs:

- *NVL(a, b)* – întoarce a, dacă a este NOT NULL, altfel întoarce b;
- *NVL2(a, b, c)* - întoarce b, dacă a este NOT NULL, altfel întoarce c;
- *COALESCE (expr_1, expr_2, …expr_n)* – întoarce prima expresie NOT NULL din listă;
- *NULLIF(a, b)* – întoarce a, dacă a!=b; altfel întoarce NULL ;
- *DECODE (expresie, val_1, val_2, val_3, val_4, …., val_2n-1, val_2n, default)* – dacă *expresie = val_1*, întoarce *val_2*; dacă *expresie = val_3*, întoarce *val_4*; …; altfel întoarce *default*.
- *DECODE* este echivalent cu *CASE*, a cărui structură este:

```
CASE expresie
        WHEN val_1 THEN val_2
        WHEN val_3 THEN val_4
        …
        ELSE default
END
```

CASE poate avea si forma:

```
CASE
        WHEN expr_logica_1 THEN val_2
        WHEN expr_logica_3 THEN val_4
        …
        ELSE default
END
```

11. Să se afişeze **informaţii despre departamente**, în formatul următor: „Departamentul <*department_name*> este condus de {<*manager_id*> | *nimeni*} şi {are numărul de salariaţi <*n*> | nu are salariati}“.

12. Să se afişeze **numele**, **prenumele angajaţilor** şi **lungimea numelui** pentru înregistrările în care aceasta este diferită de lungimea prenumelui.

13. Să se afişeze **numele**, **data angajării**, **salariul** şi o coloană reprezentând **salariul după ce se aplică o mărire**, astfel: pentru salariaţii angajaţi în 1989 creşterea este de 20%, pentru cei angajaţi în 1990 creşterea este de 15%, iar salariul celor angajaţi în anul 1991 creşte cu 10%. Pentru salariaţii angajaţi în alţi ani valoarea nu se modifică.

```
SELECT   last_name, hire_date, salary,
      CASE TO_CHAR(hire_date, 'yyyy')
         WHEN '1989' THEN salary * 1.20
         WHEN '1990' THEN salary * 1.15
         WHEN '1991' THEN salary * 1.10
         ELSE salary
      END "Salariu marit"
FROM    employees;
```

Instrucţiunea din acest exemplu poate fi rescrisă utilizând funcţia ***DECODE*** în modul următor:

```
SELECT   last_name, hire_date, salary,
       DECODE (TO_CHAR(hire_date, 'yyyy'),
                                '1989', salary * 1.20,
                                '1990', salary * 1.15,
                                '1991', salary * 1.10,
                                 salary) "Salariu marit"
FROM     employees;
```

14. Să se afişeze:
    a) suma salariilor, pentru job-urile care incep cu litera S;
    b) media generala a salariilor, pentru job-ul avand salariul maxim;
    c) salariul minim, pentru fiecare din celelalte job-uri.