Interogări multi-relaţie. Operaţia de join. Operatori pe mulţimi. Subcereri nesincronizate (necorelate).

I. [Obiective]

În acest laborator vom continua lucrul cu interogări ***multi-relaţie*** (acestea sunt cele care regăsesc date din mai multe tabele). Am introdus deja diferite tipuri de ***join***. Vom relua această operaţie, vom analiza şi o altă metodă de implementare a ei şi, de asemenea, vom utiliza ***operatori pe mulţimi*** şi ***subcereri necorelate*** (fără sincronizare).

Foarte utile în rezolvarea exerciţiilor propuse vor fi ***funcţiile SQL***, prezentate în laboratorul 2.

II. [Join]

Am implementat deja operaţia de ***join*** (compunere a tabelelor) în cadrul unor exemple relative la modelul utilizat în exemple și exerciții (HR).

Join-ul este operaţia de regăsire a datelor din două sau mai multe tabele, pe baza valorilor comune ale unor coloane. De obicei, aceste coloane reprezintă cheia primară, respectiv cheia externă a tabelelor. Reamintim că pentru a realiza un *join* între ***n* tabele**, va fi nevoie de cel puţin ***n* – 1 condiţii de *join***.

Tipuri de join :

- ***Inner join (equijoin, join simplu)*** – corespunde situaţiei în care valorile de pe coloanele ce apar în condiţia de *join* trebuie să fie egale.
- ***Nonequijoin* –** condiţia de *join* conţine alţi operatori decât operatorul de egalitate.

Exemplu Nonequijoin:

```
SELECT last_name, salary, grade_level, lowest_sal, highest_sal
FROM employees, job_grades
WHERE salary BETWEEN lowest_sal AND highest_sal;
```

- ***Left | Right Outer join*** – un *outer join* este utilizat pentru a obţine în rezultat şi înregistrările care nu satisfac condiţia de *join*. Operatorul pentru ***outer join*** este semnul plus inclus între paranteze **(+)**, care se plasează în acea parte a condiţiei de *join* care este **deficitară în informaţie**. Efectul acestui operator este de a uni liniile tabelului care nu este deficitar în informaţie, cărora nu le corespunde nici o linie în celălalt tabel, cu o linie cu valori *null*. Operatorul (+) poate fi plasat în orice parte a condiţiei de *join*, dar **nu în ambele părţi**.

Obs: O condiţie care presupune un *outer join* nu poate utiliza operatorul *IN* şi nu poate fi legată de altă condiţie prin operatorul *OR*.

- ***Full outer join*** – left outer join + right outer join
- ***Self join*** – join-ul unui tabel cu el însuşi. În ce situaţie concretă (relativ la modelul nostru) apărea această operaţie?

Join introdus în standardul SQL3 (SQL:1999)*:*

Pentru *join*, sistemul *Oracle* oferă şi o sintaxă specifică, în conformitate cu standardul *SQL3* (*SQL*:*1999*). Această sintaxă nu aduce beneficii, în privinţa performanţei, faţă de *join*-urile care folosesc sintaxa utilizată anterior. Tipurile de *join* conforme cu *SQL3* sunt definite prin cuvintele cheie *CROSS JOIN* (pentru produs cartezian), *NATURAL JOIN*, *FULL OUTER JOIN*, clauzele *USING* şi *ON*.

Sintaxa corespunzătoare standardului *SQL3* este următoarea:

SELECT *tabel_1.nume_coloană*, *tabel_2.nume_coloană*
FROM *tabel_1*
[***CROSS JOIN*** *tabel_2*]
| [***NATURAL JOIN*** *tabel_2*]
| [***JOIN*** *tabel_2* ***USING*** (*nume_coloană*)]
| [***JOIN*** *tabel_2* ***ON*** (*conditie*)]
| [***LEFT*** | ***RIGHT*** | ***FULL OUTER JOIN*** *tabel_2*
ON (*tabel_1.nume_coloană = tabel_2.nume_coloană*)];

- ***NATURAL JOIN*** presupune existenţa unor coloane având acelaşi nume în ambele tabele. Clauza determină selectarea liniilor din cele două tabele, care au valori egale în aceste coloane. Dacă tipurile de date ale coloanelor cu nume identice sunt diferite, va fi returnată o eroare.

  Coloanele având acelaşi nume în cele două tabele trebuie să nu fie precedate de numele sau *alias*-ul tabelului corespunzător.

Exemplu:

```
SELECT last_name, job_id, job_title
FROM employees
NATURAL JOIN jobs;
```

```
SELECT last_name, e.job_id, job_title
FROM employees e, jobs j
WHERE e.job_id = j.job_id;
```

- ***JOIN tabel_2 USING nume_coloană*** efectuează un ***equijoin*** pe baza coloanei cu numele specificat în sintaxă. Această clauză este utilă dacă există coloane având acelaşi nume, dar tipuri de date diferite. Coloanele referite în clauza *USING* trebuie să nu conţină calificatori (să nu fie precedate de nume de tabele sau *alias*-uri) în nici o apariţie a lor în instrucţiunea *SQL*. Clauzele *NATURAL JOIN* şi *USING* nu pot coexista în aceeaşi instrucţiune *SQL*.

Exemplu:

```
SELECT last_name, department_name, location_id
FROM employees JOIN departments USING (department_id);
```

- ***JOIN tabel_2 ON conditie*** efectuează un *join* pe baza condiţiei exprimate în clauza *ON*. Această clauză permite specificarea separată a condiţiilor de *join*, respectiv a celor de căutare sau filtrare (din clauza *WHERE*).
  În cazul operației *equijoin*, *conditie* are forma următoare :

  tabel_1.nume_coloană = tabel_2.nume_coloană

Exemplu:

```
SELECT last_name, department_name, location_id

FROM employees e JOIN departments d ON (e.department_id = d.department_id);
```

- ***LEFT*, *RIGHT* şi *FULL OUTER JOIN tabel_2 ON*** (*tabel_1.nume_coloană = tabel_2.nume_coloană*) efectuează *outer join* la stânga, dreapta, respectiv în ambele părţi pe baza condiţiei exprimate în clauza *ON*.
  Un *join* care returnează rezultatele unui *inner join*, dar şi cele ale *outer join*-urilor la stânga şi la dreapta se numeşte *full outer join*.

Exemplu OUTER JOIN:

```
SELECT last_name, department_name, location_id

FROM employees e LEFT OUTER JOIN departments d ON (e.department_id = d.department_id);
```

Acestă cerere sql este echivalentă cu:

```
SELECT last_name, department_name, location_id

FROM employees e, departments d WHERE e.department_id = d.department_id(+);
```

Returnează și angajații care nu au departamente. **LEFT OUTER JOIN** returnează toate rândurile (înregistrările) din tabelul din stânga (left), specificat în condiția **ON** și numai acele rândurie din tabelul din dreapta care îndeplinesc condiția din **JOIN**. Astfel, în exemplul de mai sus va returna toate înregistrările din tabelul Employees și numai acele înregistrări din Departments care îndeplinesc condiția.

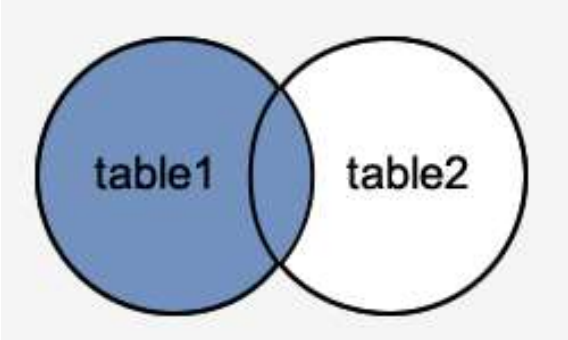

Fig1. (LEFT OUTER JOIN)

LEFT OUTER JOIN (Fig1) va returna toate înregistrările din tabelul 1 și numai acele înregistrări din tabelul 2 care se intersectează cu tabelul 1.

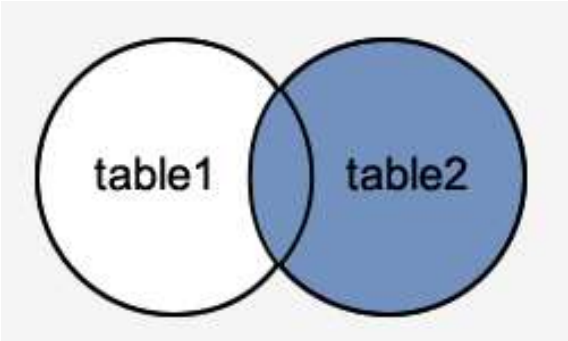

Fig2. (RIGHT OUTER JOIN)

III. [Operatori pe mulţimi]

Operatorii pe mulţimi combină rezultatele obţinute din două sau mai multe interogări. Cererile care conţin operatori pe mulţimi se numesc ***cereri compuse***. Există patru operatori pe mulţimi: *UNION*, *UNION ALL*, *INTERSECT* şi *MINUS*.

Toţi operatorii pe mulţimi au aceeaşi precedenţă. Dacă o instrucţiune *SQL* conţine mai mulţi operatori pe mulţimi, *server*-ul *Oracle* evaluează cererea de la stânga la dreapta (sau de sus în jos). Pentru a schimba această ordine de evaluare, se pot utiliza paranteze.

- ***Operatorul UNION*** returnează toate liniile selectate de două cereri, eliminând duplicatele. Acest operator nu ignoră valorile *null* şi are precedenţă mai mică decât operatorul *IN*.
- Operatorul ***UNION ALL*** returnează toate liniile selectate de două cereri, fără a elimina duplicatele. Precizările făcute asupra operatorului *UNION* sunt valabile şi în cazul operatorului *UNION ALL*. În cererile asupra cărora se aplică *UNION ALL* nu poate fi utilizat cuvântul cheie *DISTINCT*.
- Operatorul ***INTERSECT*** returnează toate liniile comune cererilor asupra cărora se aplică. Acest operator nu ignoră valorile *null*.
- Operatorul ***MINUS*** determină liniile returnate de prima cerere care nu apar în rezultatul celei de-a doua cereri. Pentru ca operatorul *MINUS* să funcţioneze, este necesar ca toate coloanele din clauza *WHERE* să se afle şi în clauza *SELECT*.

Observaţii:

- În mod implicit, pentru toţi operatorii cu excepţia lui *UNION ALL*, rezultatul este ordonat crescător după valorile primei coloane din clauza *SELECT*.
- Pentru o cerere care utilizează operatori pe mulţimi, cu excepţia lui *UNION ALL*, *server*-ul *Oracle* elimină liniile duplicat.
- În instrucţiunile *SELECT* asupra cărora se aplică operatori pe mulţimi, coloanele selectate trebuie să corespundă ca număr şi tip de date. Nu este necesar ca numele coloanelor să fie identice. Numele coloanelor din rezultat sunt determinate de numele care apar în clauza *SELECT* a primei cereri.

IV. [Subcereri]

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

Subcererea returnează o valoare care este utilizată de către instrucţiunea „părinte“. Utilizarea unei subcereri este echivalentă cu efectuarea a două cereri secvenţiale şi 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 şi determină o valoare (sau o mulţime 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;
  - 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**.

Operatorul *NOT* poate fi utilizat în combinaţie cu *IN*, *ANY* şi *ALL*.

V. [Exerciţii - join]

1. Scrieți o cerere pentru a se afisa **numele**, **luna** (în litere) şi **anul angajării** pentru toți salariații din acelasi departament cu Gates, al căror nume conţine litera “a”. Se va exclude Gates.

2. Să se afișeze **codul** şi **numele angajaţilor** care lucrează în același departament cu cel puţin un angajat al cărui nume conţine litera “t”. Se vor afişa, de asemenea, **codul** şi **numele departamentului** respectiv. Rezultatul va fi ordonat alfabetic după nume.

3. Să se afișeze **numele**, **salariul**, **titlul job-ului**, **oraşul** şi **ţara** în care lucrează angajații conduși direct de King.

4. Să se afișeze **codul departamentului**, **numele departamentului**, **numele şi job-ul** tuturor angajaților din departamentele al căror nume conţine şirul ‘ti’. De asemenea, se va lista salariul angajaţilor, în formatul “$99,999.00”. Rezultatul se va ordona alfabetic după numele departamentului, şi în cadrul acestuia, după numele angajaţilor.

SELECT d.department_id, department_name, job_id, last_name, to_char(salary,'$99,999.00')

FROM employees e JOIN departments d ON (e.department_id = d.department_id)

WHERE lower(department_name) like '%ti%'

ORDER BY _____;

5. Cum se poate implementa ***full outer join***?

Obs: Full outer join se poate realiza fie prin reuniunea rezultatelor lui ***right outer join*** şi ***left outer join***, fie utilizând sintaxa specifică standardului SQL3.

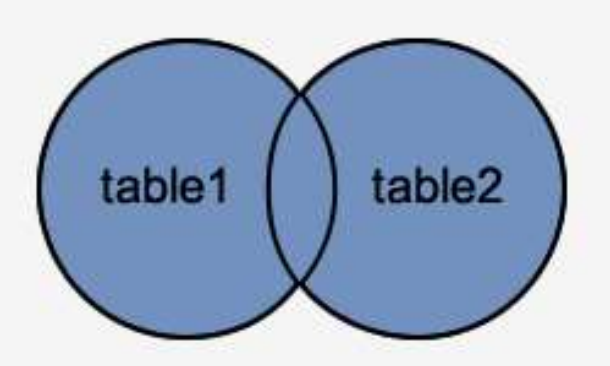

Fig3. (FULL OUTER JOIN)

VI. [Exerciţii - operatori pe mulţimi]

6. Se cer **codurile departamentelor** al căror nume conţine şirul “re” sau în care lucrează angajaţi având codul job-ului “SA_REP”.

Cum este ordonat rezultatul?

7. Ce se întâmplă dacă înlocuim *UNION* cu *UNION ALL* în comanda precedentă?

8. Să se obțină **codurile departamentelor** în care nu lucreaza nimeni (nu este introdus nici un salariat în tabelul *employees*). Se cer două soluţii.

Obs: Operatorii pe mulţimi pot fi utilizaţi în subcereri. Coloanele care apar în clauza *WHERE* a interogării trebuie să corespundă, ca număr şi tip de date, celor din clauza *SELECT* a subcererii.

```
SELECT department_id “Cod departament”
FROM     departments
MINUS
SELECT department_id
FROM     employees;
```

Utilizând subcereri:

```
SELECT department_id
FROM departments
WHERE department_id NOT IN (SELECT DISTINCT NVL(department_id,0)
                            FROM employees);
```

? În a doua variantă, de ce este nevoie de utilizarea funcţiei *NVL*?

9. Se cer codurile departamentelor al căror nume conţine şirul “re” şi în care lucrează angajaţi având codul job-ului “HR_REP”.

```
SELECT department_id “Cod departament”
FROM     employees
WHERE  UPPER(job_id)=’HR_REP’
INTERSECT
SELECT department_id
FROM    departments
WHERE LOWER(department_name) LIKE ‘%re%’;
```

VII. [Exercitii - subcereri necorelate]

10. Folosind subcereri, să se afişeze **numele** şi **data angajării** pentru salariaţii care au fost angajaţi după Gates.

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

11. Folosind subcereri, scrieţi o cerere pentru a afişa **numele** şi **salariul** pentru toţi colegii (din acelaşi departament) lui Gates. Se va exclude Gates.

Se poate înlocui operatorul IN cu = ???

Se va inlocui Gates cu King

12. Folosind subcereri, să se afişeze **numele** şi **salariul** angajaţilor conduşi direct de preşedintele companiei (acesta este considerat angajatul care nu are manager).

13. Scrieți o cerere pentru a afişa **numele**, **codul departamentulu**i 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);
```

14. 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);
```

15. Scrieti o cerere pentru a afișa angajații care câștigă (salariul plus comision) mai mult decât **oricare** funcționar (job-ul conţine şirul “CLERK”). Sortați rezultatele dupa salariu, în ordine descrescătoare.

16. Scrieţi o cerere pentru a afişa **numele angajatilor**, **numele departamentului** şi **salariul angajaţilor** care nu câştigă comision, dar al căror şef direct câştigă comision.

```
SELECT last_name, department_name, salary
FROM employees e JOIN departments d USING (department_id)
WHERE commission_pct is null and
    e.manager_id IN (SELECT employee_id
                     FROM employees
                     WHERE commission_pct is not null);
```

17. Să se afișeze **numele angajatilor**, **departamentul**, **salariul** şi **job-ul** tuturor angajaților al căror salariu și comision **coincid** cu salariul și comisionul unui angajat din Oxford.

```
SELECT last_name, department_id, salary, job_id, employee_id
FROM employees
WHERE (nvl(commission_pct, -1), salary)  IN
       (SELECT nvl(commission_pct, -1), salary
        FROM employees e JOIN departments d ON (e.department_id = d.department_id)
                              JOIN locations l ON (l.location_id = d.location_id)
        WHERE initcap(l.city)='Oxford'
        );
```

18. Să se afişeze **numele angajaţilor**, **codul departamentului** şi **codul job-ului** salariaţilor al căror departament se află în Toronto.