

**Interogări multi-relație. Operația de join. Operatori pe mulțimi.
Subcereri nesincronizate (necorelate).**

I. [Obiective]

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

Foarte utile în rezolvarea exercițiilor propuse vor fi **funcțiile SQL**, prezentate în laboratorul 2.

II. [Join]

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

Join-ul este operația 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 puțin **n – 1 condiții de join**.

Tipuri de join :

- **Inner join (equijoin, join simplu)** – corespunde situației în care valorile de pe coloanele ce apar în condiția de **join** trebuie să fie egale.
- **Nonequijoin** – condiția de **join** conține alți 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 obține în rezultat și înregistrările care nu satisfac condiția de **join**. Operatorul pentru **outer join** este semnul plus inclus între paranteze (**+**), care se plasează în acea parte a condiției de **join** care este **deficitară în informație**. Efectul acestui operator este de a uni liniile tabelului care nu este deficitar în informație, cărora nu le corespunde nici o linie în celălalt tabel, cu o linie cu valori **null**. Operatorul (+) poate fi plasat în orice parte a condiției de **join**, dar **nu în ambele părți**.

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

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

Join introdus în standardul SQL3 (SQL:1999):

Pentru *join*, sistemul *Oracle* oferă și o sintaxă specifică, în conformitate cu standardul SQL3 (SQL:1999). Această sintaxă nu aduce beneficii, în privința performanței, față 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* și *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 existența unor coloane având același 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 același 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	SELECT last_name, e.job_id, job_title
FROM employees	FROM employees e, jobs j
NATURAL JOIN jobs;	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 același nume, dar tipuri de date diferite. Coloanele referite în clauza *USING* trebuie să nu conțină calificatori (să nu fie precedate de nume de tabele sau *alias*-uri) în nici o apariție a lor în instrucțiunea SQL. Clauzele *NATURAL JOIN* și *USING* nu pot coexista în aceeași instrucțiune 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 condiției exprimate în clauza **ON**. Această clauză permite specificarea separată a condițiilor 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 și FULL OUTER JOIN tabel_2 ON** (*tabel_1.nume_coloană = tabel_2.nume_coloană*) efectuează *outer join* la stânga, dreapta, respectiv în ambele părți pe baza condiției exprimate în clauza **ON**.

Un *join* care returnează rezultatele unui *inner join*, dar și cele ale *outer join*-urilor la stânga și la dreapta se numește *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 rânduri 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.

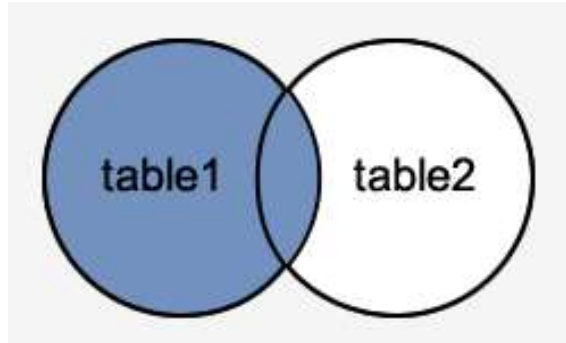


Fig1. (LEFT OUTER JOIN)

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

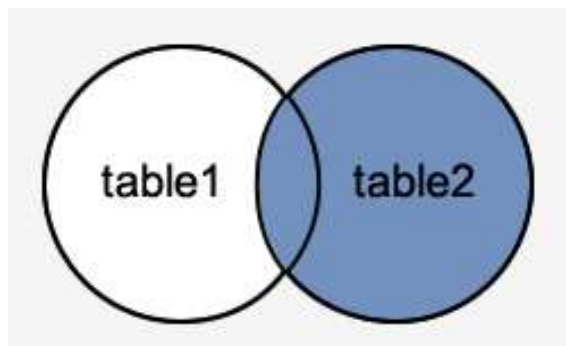


Fig2. (RIGHT OUTER JOIN)

III. [Operatori pe mulțimi]

Operatorii pe mulțimi combină rezultatele obținute din două sau mai multe interogări. Cererile care conțin operatori pe mulțimi se numesc **cereri compuse**. Există patru operatori pe mulțimi: *UNION*, *UNION ALL*, *INTERSECT* și *MINUS*.

Toți operatorii pe mulțimi au aceeași precedență. Dacă o instrucțiune SQL conține mai mulți operatori pe mulțimi, 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* și are precedență 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 și 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 funcționeze, este necesar ca toate coloanele din clauza *WHERE* să se afle și în clauza *SELECT*.

Observații:

- În mod implicit, pentru toți operatorii cu excepția lui *UNION ALL*, rezultatul este ordonat crescător după valorile primei coloane din clauza *SELECT*.
- Pentru o cerere care utilizează operatori pe mulțimi, cu excepția lui *UNION ALL*, server-ul *Oracle* elimină liniile duplicate.
- În instrucțiunile *SELECT* asupra cărora se aplică operatori pe mulțimi, coloanele selectate trebuie să corespundă ca număr și 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 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**.

V. [Exerciții - join]

1. Scrieți o cerere pentru a se afișa **numele, luna** (în litere) și **anul angajării** pentru toți salariații din același departament cu Gates, al cărui nume conține litera "a". Se va exclude Gates.

2. Să se afișeze **codul și numele angajaților** care lucrează în același departament cu cel puțin un angajat al cărui nume conține litera "t". Se vor afișa, de asemenea, **codul și numele departamentului** respectiv. Rezultatul va fi ordonat alfabetic după nume.

3. Să se afișeze **numele, salariul, titlul job-ului, orașul și țara** în care lucrează angajații conduși direct de King.

4. Să se afișeze **codul departamentului, numele departamentului, numele și job-ul** tuturor angajaților din departamentele al căror nume conține șirul 'ti'. De asemenea, se va lista salariul angajaților, în formatul "\$99,999.00". Rezultatul se va ordona alfabetic după numele departamentului, și în cadrul acestuia, după numele angajaților.

```
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** și **left outer join**, fie utilizând sintaxa specifică standardului SQL3.

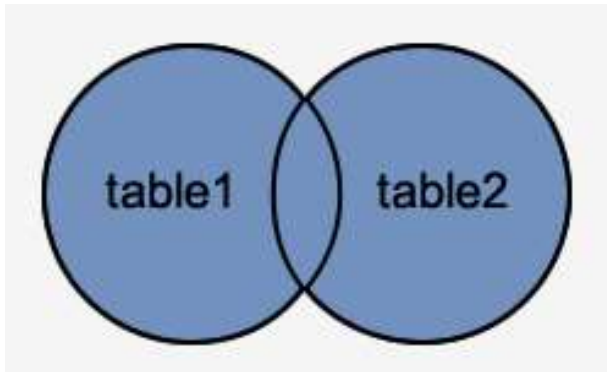


Fig3. (FULL OUTER JOIN)

VI. [Exerciții - operatori pe mulțimi]

6. Se cer **codurile departamentelor** al căror nume conține șirul “re” sau în care lucrează angajați 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 soluții.

Obs: Operatorii pe mulțimi pot fi utilizați în subcereri. Coloanele care apar în clauza *WHERE* a interogării trebuie să corespundă, ca număr și 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 funcției *NVL*?

9. Se cer codurile departamentelor al căror nume conține șirul “re” și în care lucrează angajați 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 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');
```

11. 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

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

13. 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);
```


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 conține șirul "CLERK"). Sortați rezultatele după salariu, în ordine descrescătoare.

16. Scrieți o cerere pentru a afișa **numele angajaților, numele departamentului și salariul angajaților** care nu câștigă comision, dar al căror șef direct câștigă 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 manager_id
                      FROM employees
                      WHERE commission_pct is not null);
```

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