

## **LABORATOR 3 SQL**

### **CERERI MULTITABEL. SUBCERERI. STANDARDUL SQL 1999 (SQL3).**

- Dacă în clauza FROM a unei comenzi SELECT apar mai multe tabele, atunci se realizează **produsul cartezian** al acestora
- Dacă este necesară obținerea de informații corelate din mai multe tabele, atunci se utilizează **condiții de join**.
- **Join-ul** este operația de regăsire a datelor din două sau mai multe tabele, pe baza valorilor comune ale unor coloane. Condițiile de corelare utilizează de obicei coloanele cheie primară și cheie externă.
- Pentru claritatea și eficiența accesului la baza de date se recomandă prefixarea numelor coloanelor cu numele tabelului din care fac parte (*nume\_tabel.nume\_coloana*).
- De asemenea, există posibilitatea de a utiliza aliasuri pentru tabelele din clauza FROM și utilizarea acestora în cadrul comenzii SELECT respective (*alias\_tabel.nume\_coloana*).
- Identificarea prin *nume\_tabel.nume\_coloana* sau *alias\_tabel.nume\_coloana* este obligatorie atunci când se face referință la o coloană ce apare în mai mult de un tabel din clauza FROM.

Tipuri de *join*:

- **equijoin** (se mai numește *inner join* sau *simple join*) – reprezintă compunerea a două tabele diferite după o condiție ce conține operatorul de egalitate;

```
SELECT last_name, department_name, location_id, e.department_id
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;
```

- **nonequijoin** – reprezintă compunerea a două tabele după o condiție ce nu conține operatorul de egalitate;

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

- **outerjoin** – reprezintă compunerea externă a două tabele; tuplurile tabelului în dreptul căruia apare operatorul extern (+) sunt completate cu valori *null* dacă pentru acestea nu există niciun tuplu din celălalt tabel pentru care să fie îndeplinită condiția de corelare; operatorul extern (+) poate fi plasat în orice parte a condiției de *join*, dar nu în ambele părți ale acesteia;

#### **Observație:**

- Join-ul a două tabele ce obține doar liniile ce corespund condiției de join se numește INNER JOIN.

- Join-ul a două tabele ce obține atât liniile ce corespund condiției de join cât și cele care nu îndeplinesc această condiție se numește OUTER JOIN. Liniile dintr-un tabel care nu au corespondent în celălalt tabel sunt completate cu valori *null*.

```
SELECT last_name, department_name, location_id
FROM employees e, departments d
WHERE e.department_id(+) = d.department_id;
```

- **selfjoin** – reprezintă compunerea unui tabel cu el însuși după o condiție dată;

```
SELECT sef.last_name, angajat.last_name
FROM employees sef, employees angajat
WHERE sef.employee_id = angajat.manager_id
ORDER BY sef.last_name;
```

1. Afișați pentru fiecare angajat numele, codul și numele departamentului în care lucrează.

```
SELECT last_name, e.department_id, department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

2. Afișați numele angajatului, numele departamentului pentru toți angajații care câștigă comision.
3. Afișați numele job-urile care există în departamentul 30.
4. Afișați numele, job-ul și numele departamentului pentru toți angajații care lucrează în Seattle.

```
SELECT last_name, job_id, department_name
FROM employees e, departments d, locations s
WHERE e.department_id = d.department_id
AND d.location_id = s.location_id
AND city = 'Seattle';
```

5. Afișați numele, salariul, data angajării și numele departamentului pentru toți programatorii (numele jobului este Programmer) care lucrează în America de Nord sau în America de Sud (numele regiunii este Americas).
6. Afișați numele salariaților și numele departamentelor în care lucrează. Se vor afișa și salariații al căror departament nu este cunoscut (*left outer join*).

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

7. Afișați numele departamentelor și numele salariaților care lucrează în acestea. Se vor afișa și departamentele care nu au salariați (*right outer join*).

8. Afișați numele departamentelor și numele salariaților care lucrează în acestea. Se vor afișa și salariații al căror departament nu este cunoscut, respectiv și departamentele care nu au salariați (*full outer join*).

**Observație:** *full outer join = left outer join UNION right outer join*

9. Afișați numele, job-ul, numele departamentului, salariul și grila de salarizare pentru toți angajații.
10. Afișați codul angajatului și numele acestuia, împreună cu numele și codul șefului său direct. Etichetați coloanele CodAng, NumeAng, CodMgr, NumeMgr.

```
SELECT a.employee_id "CodAng", a.last_name "NumeAng",  
       b.employee_id "CodMgr", b.last_name "NumeMgr"  
FROM   employees a, employees b  
WHERE  a.manager_id = b.employee_id;
```

11. Modificați cererea anterioară astfel încât să afișați toți salariații, inclusiv pe cei care nu au șef.
12. Afișați numele salariatului și data angajării împreună cu numele și data angajării șefului direct pentru salariații care au fost angajați înaintea șefilor lor.
13. Pentru fiecare angajat din departamentele 20 și 30 afișați numele, codul departamentului și toți colegii săi (salariații care lucrează în același departament cu el).
14. Afișați numele și data angajării pentru salariații care au fost angajați după Fay.

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

15. Rezolvați exercițiul anterior utilizând join-uri.

```
SELECT a.last_name, a.hire_date  
FROM   employees a, employees b  
WHERE  UPPER(b.last_name) = 'FAY'  
AND    a.hire_date > b.hire_date;
```

16. Scrieți o cerere pentru a afișa numele și salariul pentru toți colegii (din același departament) lui Fay. Se va exclude Fay.

```
SELECT last_name, salary  
FROM   employees  
WHERE  last_name <> 'Fay'  
AND    department_id = (SELECT department_id  
                       FROM employees  
                       WHERE last_name = 'Fay');
```

**17.** Afișați numele și salariul angajaților conduși direct de Steven King.

```
SELECT last_name, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                     FROM   employees
                     WHERE  UPPER(last_name) = 'KING'
                     AND    UPPER(first_name) = 'STEVEN');
```

**18.** Afișați numele și job-ul tuturor angajaților din departamentul 'Sales'.

```
SELECT last_name, job_id
FROM   employees
WHERE  department_id = (SELECT department_id
                       FROM   departments
                       WHERE  department_name = 'Sales');
```

**19.** Rezolvați exercițiul anterior utilizând *join-uri*.

**20.** Afișați numele angajaților, numărul departamentului și job-ul tuturor salariaților al căror departament este localizat în Seattle.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  department_id IN
      (SELECT department_id
       FROM   departments
       WHERE  location_id = (SELECT location_id
                             FROM   locations
                             WHERE  city = 'Seattle'));
```

**21.** Rezolvați exercițiul anterior utilizând *join-uri*.

**22.** Aflați dacă există angajați care nu lucrează în departamentul Sales, dar au aceleași câștiguri (salariu și comision) ca și un angajat din departamentul Sales.

Observație: Angajatul cu codul 178 îndeplinește condiția (nu lucrează în niciun departament), deci ar trebui afișat.

```
SELECT last_name, salary, commission_pct, department_id
FROM   employees
WHERE  (salary, commission_pct) IN
      (SELECT salary, commission_pct
       FROM   employees e, departments d
       WHERE  e.department_id = d.department_id
       AND    department_name = 'Sales')
AND    department_id <> (SELECT department_id
```

```
FROM departments
WHERE department_name = 'Sales';
```

## TEMĂ

23. Afișați salariații care au același manager ca și angajatul având codul 140.
24. Afișați numele departamentelor care funcționează în America.
25. Afișați numele angajatului, numele șefului său direct, respectiv numele șefului căruia i se subordonează șeful său direct.
26. Afișați pentru fiecare salariat angajat în luna martie numele său, data angajării și numele jobului.
27. Afișați pentru fiecare salariat al cărui câștig total lunar este mai mare decât 12000 numele său, câștigul total lunar și numele departamentului în care lucrează.
28. Afișați pentru fiecare angajat codul său și numele joburilor sale anterioare, precum și intervalul de timp în care a lucrat pe jobul respectiv.
29. Modificați cererea de la punctul 28 astfel încât să se afișeze și numele angajatului, respectiv codul jobului său curent.
30. Modificați cererea de la punctul 29 astfel încât să se afișeze și numele jobului său curent.
31. Modificați cererea de la punctul 30 astfel încât să se afișeze informațiile cerute doar pentru angajații care au lucrat în trecut pe același job pe care lucrează în prezent.
32. Modificați cererea de la punctul 31 astfel încât să se afișeze în plus numele departamentului în care a lucrat angajatul în trecut, respectiv numele departamentului în care lucrează în prezent.
33. Modificați cererea de la punctul 32 încât să se afișeze informațiile cerute doar pentru angajații care au lucrat în trecut pe același job pe care lucrează în prezent, dar în departamente diferite.

## FACULTATIV

Comparați sintaxa Oracle și cu **standardul SQL3 (SQL 1999)**.

**Observație:** Sistemul Oracle oferă pentru join și o sintaxă specifică, în conformitate cu standardul SQL3.

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]
| [[INNER] JOIN tabel_2 USING (nume_coloană) ]
```

```
| [[INNER] JOIN tabel_2
      ON (tabel_1.num_coloană = tabel_2.num_coloană) ]
| [LEFT | RIGHT | FULL OUTER JOIN tabel_2
      USING (num_coloană) |
      ON (tabel_1.num_coloană = tabel_2.num_coloană)];
```

Tipuri de *join*:

- **CROSS JOIN** – realizează produsul cartezian a două tabele;

```
SELECT a.region_name, b.region_name
FROM   regions a
CROSS JOIN regions b;

SELECT a.region_name, b.region_name
FROM   regions a, regions b;
```

- **NATURAL JOIN** – se bazează pe coloanele cu același nume din cele două tabele și selectează liniile care au aceleași valori pentru aceste coloane;

- Dacă tipurile de date ale coloanelor cu nume identice sunt diferite, va fi returnată o eroare.
- Coloanele comune nu trebuie calificate cu aliasuri.
- Oracle oferă posibilitatea completării automate a operației de join.
- Coloanele având același nume în cele două tabele trebuie să nu fie precedate de numele sau alias-ul tabelului corespunzător.

```
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;
```

- **[INNER] JOIN ... USING** – se bazează pe coloanele cu același nume din cele două tabele și selectează liniile care au aceleași valori pentru aceste coloane;

- Dacă cele două tabele conțin coloane cu același nume, dar cu tipurile de date diferite, atunci în loc de NATURAL JOIN se utilizează JOIN cu clauza USING.
- Similar cu NATURAL JOIN, coloana de legătură nu trebuie calificată cu alias oriunde în cadrul cererii.
- Acest tip de JOIN se mai folosește atunci când cele două tabele au mai multe coloane cu nume comun, dar legătura dintre ele trebuie realizată doar utilizând câteva dintre acestea.

```
SELECT last_name, department_name, location_id
```

```
FROM employees
JOIN departments USING (department_id);

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

SELECT last_name, department_name, location_id
FROM employees e, departments d
WHERE e.department_id=d.department_id;
```

- **[INNER] JOIN ... ON** – realizează compunerea a două tabele pe baza unei condiții specificate;
  - Dacă se dorește particularizarea, specificarea și/sau evidențierea clară a condiției de join se utilizează clauza ON. Aceasta separă condiția de JOIN de alte condiții, făcând codul mai lizibil.
  - Într-o înlănțuire de JOIN-uri acestea sunt evaluate de la stânga la dreapta.

```
SELECT department_name, city
FROM departments d
JOIN locations l ON (d.location_id = l.location_id);

SELECT department_name, city
FROM departments d
INNER JOIN locations l ON (d.location_id = l.location_id);

SELECT department_name, city
FROM departments d, locations l
WHERE d.location_id = l.location_id;
```

- **LEFT | RIGHT | FULL OUTER JOIN ... USING | ON** – realizează compunerea externă a două tabele.

- **left outer join**

```
SELECT employee_id, last_name, department_name
FROM employees
LEFT OUTER JOIN departments
USING (department_id);

SELECT employee_id, last_name, department_name
FROM employees e
LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);

SELECT employee_id, last_name, department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id(+);
```

- **right outer join**

```
SELECT employee_id, last_name, department_name
FROM employees
RIGHT OUTER JOIN departments
USING (department_id);
```

```
SELECT employee_id, last_name, department_name
FROM employees e
RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

```
SELECT employee_id, last_name, department_name
FROM employees e, departments d
WHERE e.department_id (+) = d.department_id;
```

- **full outer join**

```
SELECT employee_id, last_name, department_name
FROM employees
FULL OUTER JOIN departments
USING (department_id);
```

```
SELECT employee_id, last_name, department_name
FROM employees e
FULL OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

```
SELECT employee_id, last_name, department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id (+)
UNION
```

```
SELECT employee_id, last_name, department_name
FROM employees e, departments d
WHERE e.department_id (+) = d.department_id;
```

**34.** Dați o alternativă de rezolvare a exercițiilor anterioare implementând operația de join cu standardul SQL3.