LABORATOR 4:
CERERI MULTIRELAŢIE (outer join, sintaxa standard)
OPERATORI PE MULŢIMI

Join

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.

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 egalitate.

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 deficientă în informaţie.Efectul acestui operator este de a lega liniile tabelului care nu este deficient în informaţie şi 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.

Pentru join, sistemul Oracle10g 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 specifică Oracle. 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_coloana, tabel_2.nume_coloana
FROM tabel_1
[CROSS JOIN tabel_2]
| [NATURAL JOIN tabel_2]
| [JOIN tabel_2 USING (nume_coloana) ]
| [JOIN tabel_2 ON (tabel_1.nume_coloana = tabel_2.nume_coloana)]
| [LEFT | RIGHT | FULL OUTER JOIN tabel_2
ON (tabel_1.nume_coloana = tabel_2.nume_coloana)];
```

Exemplul 1: CROSS JOIN

```
SELECT city, l.country_id, country_name
FROM locations l CROSS JOIN countries;

SELECT city, l.country_id, country_name
FROM locations l, countries;
```

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.

În versiunile precedente ale lui Oracle Server nu era permisă realizarea unui join fără a specifica explicit coloanele care intervin în operaţie. Momentan se oferă posibilitatea completării automate a operaţiei de join. Coloanele având acelaşi nume în cele două tabele trebuie să nu fie precedate de numele sau alias-ul tabelului corespunzător.

Exemplul 2: NATURAL JOIN

```
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.

JOIN tabel2 ON tabel1.nume_coloană = tabel2.nume_coloană efectuează un equijoin 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).

Exemplul 3: JOIN

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

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

SELECT last_name, department_name, location_id
FROM employees e, departments d
WHERE 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.

Exemplul 4: OUTER JOIN

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

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

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

Exerciţiul 5: Să se afişeze numele şi titlul job-ul tuturor angajaţilor din departamentul 'Sales'. (varianta join-ului conform standardului SQL99)

Exerciţiul 6: Să se afişeze codul departamentului, numele departamentului, numele şi job-ul tuturor angajaţilor din departamentele din 'Seattle'. De asemenea, se va lista salariul angajaţilor, în formatul '$99,999.00'.

Exerciţiul 7: Scrieti o cerere pentru a se afişa numele, luna (în litere) şi anul angajării pentru toţi salariaţii din acelaşi departament cu Gates, al căror nume conţine litera 'a'. Se va exclude Gates. Se vor da 2 soluţii pentru determinarea apariţiei literei 'A' în nume. De asemenea, pentru una din metode se va da şi varianta join-ului conform standardului SQL99.

Exerciţiul 8: Să se afişeze codul şi numele angajaţilor care au început să lucreze într-un departament în care cel puţin un alt angajat avea deja o vechime de 2 luni. Se vor afişa, de asemenea, codul şi numele departamentului respectiv.

Exerciţiul 9: Sa se afiseze numele, salariul, titlul job-ului, oraşul şi ţara în care lucrează angajatii condusi direct de King.

Exerciţiul 10: Să se afişeze numele departamentelor şi numele managerilor acestora. Se vor afişa şi departamentele care nu au manageri. Pentru acestea coloana Manager va conţine textul "necunoscut" (left outer join, 2 variante)

Exerciţiul 11: Să se afişeze numele salariaţilor şi oraşele în care lucrează. Se vor afişa şi salariaţii pentru care nu este cunoscut departamentul.

Exerciţiul 12: Să se afişeze numele salariaţilor şi informaţii despre joburile avute anterior de aceştia (titlu, salariu minim posibil). Dacă salariatul este la primul job se va afişa jobul actual.

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 si MINUS.
Toţi operatorii pe mulţimi au aceeasi precedenţa. 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. 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.
Clauza ORDER BY poate apărea numai o singură dată într-o cerere compusă (la sfârsitul cererii). În mod implicit, pentru toţi operatorii cu excepţia lui UNION ALL, rezultatul este ordonat crescător după valorile primei coloane din clauza SELECT.

Operatorul **UNION** returnează toate liniile selectate de două cereri, eliminând duplicatele. Acest operator nu ignoră valorile null şi are precedensă 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 si în cazul operatorului UNION ALL. În cererile asupra cărora se aplică UNION ALL nu poate fi utilizat cuvântul cheie DISTINCT.
Toţi ceilalţi operatori pe mulţimi elimină liniile duplicat.

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.

Exemplul 13. Să se obţină codurile departamentelor al căror nume conţine sirul "re" sau în care lucrează angajaţi având codul job-ului "SA_REP".

```
SELECT department_id "Cod departament"
FROM employees
WHERE UPPER(job_id) = 'SA_REP'
UNION
SELECT department_id
FROM departments
WHERE LOWER(department_name) LIKE '%re%';
```

Exemplul 14. Să se obţină codurile departamentelor în care nu lucrează nimeni (nu este introdus nici un salariat în tabelul employees).

```
SELECT department_id "Cod departament"
FROM departments
MINUS
SELECT DISTINCT department_id
FROM employees;
```

Exemplul 15. Se cer codurile departamentelor al căror nume conţine sirul "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%'
```

Exemplul 16. Utilizând operatorul UNION, să se listeze codul salariaţilor, numele angajaţilor, codul şi numele departamentelor. Să se ordoneze rezultatul după codul şi numele departamentului.

```
SELECT employee_id, last_name, department_id, TO_CHAR(NULL) nume
FROM employees
UNION
SELECT TO_NUMBER(NULL), null, department_id, department_name
FROM departments
ORDER BY 3, nume;
```

Exerciţiul 17. Să se aseze media venitului tuturor angajaţilor si media venitului salariaţilor angajaţi în anul 2000. Să se rotunjească mediile la două zecimale. Cele două linii rezultat vor include textele ‘medie’ respectiv ‘medie 2000’.
Observaţi modul în care este ordonat rezultatul. Să se atribuie unicei coloane rezultat un titlu potrivit.

Exerciţiul 18. Să se aseze numele departamentelor şi numele angajaţilor. Se vor afisa şi departamenntele în care nu lucrează nimeni si angajaţii care nu lucrează în nici un departament (full outer join). Se va utiliza UNION.

Exerciţiul 19. Folosind INTERSECT să se afiseze codul departamentului si numele departamentului pentru departamentele din orasul Seattle, coduse de un angajat al cărui salariu este mai mare decât 7000.

Exerciţiul 20. Să se afăşeze denumirile joburilor pe care nu le-a avut şeful departamentului „Administration”.

Exerciţiul 21. Să se afişeze codurile şi numele şefilor de departament care au mai avut cel puţin un job anterior.

Exerciţiul 22. Să se obţina o listă cu istoricul funcţiilor avute de angajaţi (se va utiliza tabelul job_history). Tabelul va conţine următoarele coloane: codul angajatului, numele angajatului, titlul jobului avut, numele departamentului în care a lucrat în timpul cât a deţinut funcţia respectivă, perioada în care a ocupat poziţia respectivă exprimată în număr întreg de luni. Coloana "perioada" va conţine null, în cazul funcţiilor deţinute în prezent de angajaţi. Ordonaţi rezultatul dupa numele angajatului.