

LABORATOR 4:

CERERI MULTIRELAȚIE (outer join, sintaxa standard)

OPERATORI PE MULȚIMI

JOIN

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 tabelului. Reamintim că pentru a realiza un join între n tabele, va fi nevoie de cel puțin n - 1 condiții 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 egalitate.

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 deficientă în informație. Efectul acestui operator este de a lega liniile tabelului care nu este deficient în informație și 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.

Pentru join, sistemul Oracle10g 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 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 și 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 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.

În versiunile precedente ale lui Oracle Server nu era permisă realizarea unui join fără a specifica explicit coloanele care intervin în operație. Momentan se 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.

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 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.

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

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 și FULL OUTER JOIN tabel_2

ON (tabel_1.num_coloană = tabel_2.num_coloană) efectuează outer join la stînga, 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 stînga și la dreapta se numește 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;
```

Exercițiul 5: Să se afișeze numele și titlul job-ului tuturor angajaților din departamentul 'Sales'. (varianta join-ului conform standardului SQL99)

Exercițiul 6: Să se afișeze codul departamentului, numele departamentului, numele și job-ul tuturor angajaților din departamentele din 'Seattle'. De asemenea, se va lista salariul angajaților, în formatul '\$99,999.00'.

Exercițiul 7: Scrieti o cerere pentru a se afișa numele, luna (în litere) și anul angajării pentru toți salariații din același departament cu Gates, al căror nume conține litera 'a'. Se va exclude Gates. Se vor da 2 soluții pentru determinarea apariției literei 'A' în nume. De asemenea, pentru una din metode se va da și varianta join-ului conform standardului SQL99.

Exercițiul 8: Să se afișeze codul și numele angajaților care au început să lucreze într-un departament în care cel puțin un alt angajat avea deja o vechime de 2 luni. Se vor afișa, de asemenea, codul și numele departamentului respectiv.

Exercițiul 9: Sa se afiseze numele, salariul, titlul job-ului, orașul și țara în care lucrează angajatii condusi direct de King.

Exercițiul 10: Să se afișeze numele departamentelor și numele managerilor acestora. Se vor afișa și departamentele care nu au manageri. Pentru acestea coloana Manager va conține textul "necunoscut" (left outer join, 2 variante)

Exercițiul 11: Să se afișeze numele salariaților și orașele în care lucrează. Se vor afișa și salariații pentru care nu este cunoscut departamentul.

Exercițiul 12: Să se afișeze numele salariaților și informații despre joburile avute anterior de aceștia (titlu, salariu minim posibil). Dacă salariatul este la primul job se va afișa jobul actual.

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. 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.

Clauza ORDER BY poate apărea numai o singură dată într-o cerere compusă (la sfârșitul cererii). În mod implicit, pentru toți operatorii cu excepția 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 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. Toți ceilalți operatori pe mulțimi elimină liniile duplicate.

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 obțină codurile departamentelor al căror nume conține sirul "re" sau în care lucrează angajați 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 obțină 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 conține sirul "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%'
```

Exemplul 16. Utilizând operatorul UNION, să se listeze codul salariaților, numele angajaților, codul și numele departamentelor. Să se ordoneze rezultatul după codul și 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;
```

Exercițiul 17. Să se așeze media venitului tuturor angajaților și media venitului salariaților angajați în anul 2000. Să se rotunjească mediile la două zecimale. Cele două linii rezultat vor include textele 'medie' respectiv 'medie 2000'.

Observați modul în care este ordonat rezultatul. Să se atribue unice coloane rezultat un titlu potrivit.

Exercițiul 18. Să se așeze numele departamentelor și numele angajaților. Se vor afișa și departamentele în care nu lucrează nimeni și angajații care nu lucrează în nici un departament (full outer join). Se va utiliza UNION.

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

Exercițiul 20. Să se afișeze denumirile joburilor pe care nu le-a avut șeful departamentului „Administration”.

Exercițiul 21. Să se afișeze codurile și numele șefilor de departament care au mai avut cel puțin un job anterior.

Exercițiul 22. Să se obțină o listă cu istoricul funcțiilor avute de angajați (se va utiliza tabelul job_history). Tabelul va conține următoarele coloane: codul angajatului, numele angajatului, titlul jobului avut, numele departamentului în care a lucrat în timpul cât a deținut funcția respectivă, perioada în care a ocupat poziția respectivă exprimată în număr întreg de luni. Coloana "perioada" va conține null, în cazul funcțiilor deținute în prezent de angajați. Ordonăți rezultatul după numele angajatului.