Baze de date - Anul 1
Laborator 3

Interogări multi-relaţie: Operaţia de join. Operatori pe mulţimi. Funcții (completare).

I. [Obiective]

În acest laborator vom continua lucrul cu interogări ***multi-relaţie*** (acestea sunt cele care regăsesc date din mai multe tabele).

În laboratorul precedent 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***.

Foarte utile în rezolvarea exerciţiilor propuse vor fi ***funcţiile SQL***, prezentate în laboratorul 2. Vom completa lista de funcții utilizate anterior cu alte cateva exemple, la finalul laboratorului.

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***. Dacă una dintre condițiile de join nu este specificată, va fi generat produsul cartezian al tabelelor respective.

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.
- ***Left | Right Outer join*** – un *outer join* este utilizat pentru a obţine în rezultat şi înregistrările care nu satisfac condiţia de *join*. O operație de *join* implementată cu ajutorul unei condiții specificate in clauza *WHERE* devine *outer-join* adăugând semnul plus inclus între paranteze **(+)** 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 – un caz particular al operației de join, ce apare atunci când este realizat *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***, ***JOIN***, ***LEFT*** | ***RIGHT*** | ***FULL OUTER JOIN***, clauzele ***USING*** şi ***ON***.

Sintaxa corespunzătoare acestor tipuri de join (în standardul *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*
{ ***USING*** (*nume_coloană*) | ***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.

- ***JOIN tabel_2 USING (nume_coloană)*** efectuează un *equijoin* pe baza coloanei cu numele specificat în sintaxă. Această clauză este utilă dacă există mai multe coloane având acelaşi nume, dar operația de *join* nu trebuie realizată după toate aceste coloane. 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 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

- ***{LEFT | RIGHT | FULL} [OUTER] JOIN tabel_2 {USING (nume_coloană) | ON (tabel_1.nume_coloană = tabel_2.nume_coloană)}*** efectuează *outer join* la stânga, dreapta, respectiv în ambele părţi pe baza egalității coloanei specificate în clauza *USING*, respectiv a 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*.

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*.

- 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. [Functii]

Completări: utilizarea alternativă a funcţiei DECODE şi a structurii CASE; din nou NVL şi NVL2; COALESCE; NULLIF

Reamintim:

- *NVL*(*a*, *b*) – întoarce *a*, dacă *a* este *NOT NULL*, altfel întoarce *b*;
- *NVL2*(*a*, *b*, *c*) – întoarce *b*, dacă *a* este *NOT NULL*, altfel întoarce *c*;
- *COALESCE* (*expr*_1, *expr*_2, …*expr*_n) – întoarce prima expresie *NOT NULL* din listă;
- *NULLIF*(*a*, *b*) – întoarce *a*, dacă *a* != *b*; altfel întoarce *NULL* ;
- *DECODE* (*expresie*, *val*_1, *val*_2, [*val*_3, *val*_4, …., *val*_2*n*-1, *val*_2*n*], [*default*]) – dacă *expresie* = *val*_1, întoarce *val*_2; dacă *expresie* = *val*_3, întoarce *val*_4; …; altfel întoarce *default*.
- *DECODE* este echivalent cu *CASE*, a cărui structură este:

CASE expresie
WHEN val_1 THEN val_2
[WHEN val_3 THEN val_4
…]
[ELSE default]
END

CASE poate avea și forma:

CASE
WHEN expr_logica_1 THEN val_2
[WHEN expr_logica_3 THEN val_4
…]
[ELSE default]
END

V. [Exerciţii - join]

1. Scrieți o cerere prin care se afișează numele, luna (în litere) şi anul angajării pentru toți salariații din același 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 dintre metode se va da şi varianta *join*-ului conform standardului *SQL3*.

Soluție: Operația cerută nu se bazează pe o relație directă (care ar fi generat o egalitate între o cheie externă și cheia primară corespunzatoare). În schimb, relația este una indirectă, ce ar putea fi exprimată ca “ANGAJAT este coleg cu ANGAJAT”. Reamintim, din cursul de proiectare a bazelor de date, că în modelele de date nu se includ astfel de relații, ce derivă din altele. În cazul nostru, faptul că doi angajați lucrează în același department determină relația de colegialitate dintre ei.

```
SELECT e1.last_name, TO_CHAR(e1.hire_date, 'MONTH') AS "Luna",
EXTRACT(YEAR FROM e1.hire_date) as "An"
FROM employees e1,employees e2
WHERE e1.department_id = e2.department_id
      AND LOWER(e1.last_name) LIKE '%a%'
      AND LOWER(e1.last_name) != 'gates'
      AND LOWER(e2.last_name) = 'gates';
```

```
SELECT e1.last_name, TO_CHAR(e1.hire_date, 'MONTH') AS "Luna",
EXTRACT(YEAR FROM e1.hire_date) as "An"
FROM employees e1
JOIN employees e2
ON (e1.department_id = e2.department_id)
WHERE LOWER(e1.last_name) LIKE '%a%'
      AND LOWER(e1.last_name) != 'gates'
      AND LOWER(e2.last_name) = 'gates';
```

```
SELECT e1.last_name, TO_CHAR(e1.hire_date, 'MONTH') AS "Luna",
EXTRACT(YEAR FROM e1.hire_date) as "An"
FROM employees e1
JOIN employees e2
ON (e1.department_id = e2.department_id)
WHERE INSTR(LOWER(e1.last_name), 'a') != 0
      AND LOWER(e1.last_name) != 'gates'
      AND LOWER(e2.last_name) = '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.

Daţi şi soluţia care utilizează sintaxa specifică *Oracle* (anterioară SQL3) pentru join.

Soluție:

```
SELECT DISTINCT e1.employee_id, e1.last_name, e1.department_id,
d.department_name
FROM employees e1, employees e2, departments d
WHERE e1.department_id = e2.department_id
      AND e1.department_id = d.department_id
      AND LOWER(e2.last_name) LIKE '%t%'
ORDER BY e1.last_name;
```

```
SELECT UNIQUE e1.employee_id, e1.last_name, e1.department_id,
d.department_name
```

```
FROM employees e1
JOIN employees e2 ON (e1.department_id = e2.department_id)
JOIN departments d ON (e1.department_id = d.department_id)
WHERE LOWER(e2.last_name) LIKE '%t%'
ORDER BY e1.last_name;
```

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

Daţi două metode de rezolvare a acestui exerciţiu.

Soluție: Tratăm inclusiv cazul în care există subalterni direcți ai lui King care nu sunt asignați unui departament (coloana *department_id* este *null*). Un astfel de angajat nu va apărea în rezultat, deoarece îl "pierdem" atunci când se realizează *join*-ul cu tabelul DEPARTMENTS.

Asfel, această variantă de cerere:

```
SELECT e.last_name, e.salary, j.job_title, l.city,
       c.country_name
FROM employees e
JOIN employees m on (e.manager_id = m.employee_id)
JOIN jobs j on (e.job_id = j.job_id)
JOIN departments d on (e.department_id = d.department_id)
JOIN locations l ON (l.location_id = d.location_id)
JOIN countries c ON (c.country_id = l.country_id)
WHERE m.last_name = 'King';
```

va putea returna un rezultat diferit de următoarea (pe care o considerăm **soluția corectă** a acestui exercițiu):

```
SELECT e.last_name, e.salary, j.job_title, l.city,
       c.country_name
FROM employees e
JOIN employees m on (e.manager_id = m.employee_id)
JOIN jobs j on (e.job_id = j.job_id)
LEFT JOIN departments d on (e.department_id = d.department_id)
LEFT JOIN locations l ON (l.location_id = d.location_id)
LEFT JOIN countries c ON (c.country_id = l.country_id)
WHERE m.last_name = 'King';
```

Observații:

- Nu am pus problema pierderii de rezultate la *join*-ul cu tabelul JOBS, deoarece atributul *job_id* are constrângerea NOT NULL. Ne putem convinge de acest lucru cu ajutorul comenzii "describe jobs;". Prin urmare, coloana *job_id* din EMPLOYEES nu va fi niciodată *null*, deci nu se va pierde nicio înregistrare în acel *join*.
- De ce sunt necesare ultimele două operații LEFT JOIN de mai sus ? Pentru a păstra linia "câștigată" la primul LEFT JOIN, operația se propagă la toate *join*-urile care îi urmează. Altfel, doar cu primul LEFT JOIN în cerere, rezultatul va fi același cu cel de la varianta anterioară (cea incorectă).

Temă: A doua metodă (transformarea cererii astfel încât operațiile de join să fie specificate în clauza WHERE)

4. Să se afișeze codul departamentului, numele departamentului, numele și 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.

Soluție:

```
SELECT d.department_id, d.department_name, e.last_name,
       j.job_title, TO_CHAR(e.salary, '$99,999.00') Salariu
FROM departments d
JOIN employees e ON (e.department_id = d.department_id)
JOIN jobs j ON (e.job_id = j.job_id)
WHERE LOWER(d.department_name) LIKE '%ti%'
ORDER BY d.department_name, e.last_name;
```

De ce nu mai folosim OUTER JOIN? Fiindcă avem o condiție (LIKE) legată de coloana *department_name*, nu ar avea sens obținerea înregistrărilor pentru care *department_id* este *null* (în tabelul EMPLOYEES). Aceste linii ar conține valoarea *null* și pentru *department_name*, deci cu siguranță nu ar îndeplini condiția cerută.

5. Să se afișeze numele angajaților, numărul departamentului, numele departamentului, oraşul și job-ul tuturor salariaților al căror departament este localizat în Oxford.

Soluție:

```
SELECT e.last_name, department_id, department_name, city ,
       job_title
FROM employees e
JOIN departments d USING(department_id)
JOIN locations l USING(location_id)
JOIN jobs j USING(job_id)
WHERE LOWER(l.city) = 'oxford';
```

Motivul pentru care nu folosim OUTER JOIN nici la această problemă este similar cu cel de la problema anterioară: deoarece specificăm o condiție de egalitate pentru coloana *city* (din tabelul LOCATIONS), nu are sens să regăsim în rezultat linii pentru care *department_id* este *null*, deoarece acestea ar conține valori *null* și pe coloanele *location_id* și *city*.

6. Să se modifice cererea de la problema 2 astfel încât să afișeze codul, numele și salariul tuturor angajaților care caștigă mai mult decât salariul mediu pentru job-ul corespunzător și lucrează într-un departament cu cel puțin unul dintre angajații al căror nume conține litera “t”. Vom considera salariul mediu al unui job ca fiind egal cu media aritmetică a limitelor sale admise (specificate în coloanele *min_salary*, *max_salary* din tabelul JOBS).

Soluție:

```
SELECT UNIQUE e1.employee_id, e1.last_name, e1.salary
FROM employees e1
JOIN employees e2 ON (e2.department_id = e1.department_id)
```

```
JOIN jobs j ON (j.job_id = e1.job_id)
WHERE LOWER(e2.last_name) LIKE '%t%' --INSTR(LOWER(e2.last_name),
't') != 0)
AND e1.salary > (j.min_salary + j.max_salary) / 2
ORDER BY e1.last_name;
```

7. Să se afişeze numele salariaţilor şi numele departamentelor în care lucrează. Se vor afişa şi salariaţii care nu au asociat un departament. (outer join, 2 variante).

Soluție:

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

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

8. Să se afişeze numele departamentelor şi numele salariaţilor care lucrează în ele. Se vor afişa şi departamentele care nu au salariaţi. (outer join, 2 variante)

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

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

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

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

Soluție:

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

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

Ce observați, comparând rezultatele celor două metode de mai sus? Din cauză că există perechi egale de (*last_name*, *department_name*) și deoarece operatorul UNION elimină duplicatele, rezultatul primei cereri conține (în mod eronat) mai puține linii. Pentru ca doi angajati având același nume și departament să fie considerați diferiți, în prima metodă de

mai sus introducem o coloană care să le asigure unicitatea (chiar cheia primara – *employee_id*) :

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

VI. [Exerciţii - operatori pe mulţimi]

10. 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?

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

Soluție:

```
SELECT D.DEPARTMENT_ID
FROM DEPARTMENTS D
WHERE LOWER(D.DEPARTMENT_NAME) LIKE '%re%'
UNION
SELECT DEPARTMENT_ID
FROM EMPLOYEES E
WHERE E.JOB_ID='SA_REP';
```

11. Să se obțină codurile departamentelor în care nu lucrează nimeni (nu este introdus nici un salariat în tabelul *employees*). Se cer două soluţii (*MINUS*, *NOT IN*).

Observație: 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.

Comentați necesitatea tratării valorilor *null* în varianta utilizării operatorului *NOT IN*.

Soluție:

```
SELECT d.department_id
FROM departments d
MINUS
SELECT UNIQUE department_id
FROM employees;
```

```
SELECT department_id
FROM departments
WHERE department_id NOT IN
      (
      SELECT d.department_id
      FROM departments d
      JOIN employees e ON(d.department_id=e.department_id)
      );
```

A doua varianta utilizeaza o subcerere necorelata. Aceste subcereri vor fi prezentate in continuarea laboratorului.

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

Soluție:

```
SELECT d.department_id
FROM departments d
WHERE LOWER(d.department_name) LIKE '%re%'
INTERSECT
SELECT e.department_id
FROM employees e
WHERE e.job_id = 'HR_REP';
```

13. Să se determine codul angajaţilor, codul job-urilor şi numele celor al căror salariu este mai mare decât 3000 sau este egal cu media dintre salariul minim şi cel maxim pentru job-ul respectiv.

Soluție:

```
SELECT e.employee_id, e.job_id, e.last_name
FROM employees e
WHERE (e.salary > 3000)
UNION
SELECT e.employee_id, e.job_id, e.last_name
FROM employees e
JOIN jobs j ON (j.job_id = e.job_id)
WHERE e.salary = (j.min_salary + j.max_salary) / 2;
```

Cererea anterioară utilizează un operator pe mulțimi. Evident, există și această variantă mai simplă:

```
SELECT e.employee_id, e.job_id, e.last_name
FROM employees e
JOIN jobs j ON (e.job_id = j.job_id)
WHERE e.salary > 3000 OR e.salary = (j.min_salary + j.max_salary)
/ 2;
```

VII. [Exerciții - funcții]

14. Să se afişeze informaţii despre departamente, în formatul următor: „Departamentul <*department_name*> este condus de {<*manager_id*> | *nimeni*} şi {are salariaţi | nu are salariati}“. (Se vor utiliza *NVL*, *NVL2*, *outer join*, *CASE*). **(Temă)**

15. Să se afişeze numele, prenumele angajaţilor şi lungimea numelui pentru înregistrările în care aceasta este diferită de lungimea prenumelui. (Se va utiliza *NULLIF*). **(Temă)**

16. Să se afişeze numele, data angajării, salariul şi o coloană reprezentând salariul după ce se aplică o mărire, astfel: pentru salariaţii angajaţi în 1989 creşterea este de 20%, pentru cei angajaţi în 1990 creşterea este de 15%, iar salariul celor angajaţi în anul 1991 creşte cu 10%. Pentru salariaţii angajaţi în alţi ani valoarea nu se modifică. (*DECODE* si cele 2 forme ale lui *CASE*). **(Temă)**