Tutoriat 2

Outer join; Operatori de multimi; Subcereri

Outer Join

Dupa cum am invatat la primul tutoriat join-ul ne ajuta sa „unim” datele din 2 sau mai multe tabele dupa una sau mai multe coloane. Acest lucru este foarte util cand vrem sa gasim informatii partajate in mai multe tabele. Totusi, una dintre problemele join-ului clasic (numit si INNER JOIN) este aceea ca, daca vreau sa unesc datele din 2 tabele dupa o coloana, atunci vom avea acces numai la liniile ce au valori pe acea coloana in ambele tabele.

Sa luam ca exemplu urmatorele 2 tabele :

PERSOANA

ID(PK)	NUME	ID_MASINA(FK)
1	Andrei	2
2	Ana	1
3	Maria	4
4	Ion	3
5	Raluca	(null)
6	Alex	(null)

MASINA

ID(PK)	MARCA
1	Dacia
2	BMW
3	Audi
4	Honda
5	Lexus

Daca dorim sa afisam numele fiecarei persoana si ce marca de masina foloseste codul si rezultatul vor arata astfel:

```
1.  SELECT p.nume, m.marca
2.  FROM persoana p, masina m
3.  WHERE p.id_masina = m.id;
```

p.nume	m.marca
Andrei	BMW
Ana	Dacia
Maria	Honda
Ion	Audi

Acest rezultat este corect, insa, daca dorim sa afisam toate persoanele si masina fiecaruia iar in cazul in care persoana nu are masina sa afisam (null) atunci trebuie sa ne folosim de un tip de join numit OUTER JOIN.

Acest tip de join se foloseste cand dorim sa afisam toate datele dintru-un tabel, chiar daca acestea nu satisfac conditia din join. El se semnaleaza punand semnul (+) in conditia de join, in dreapta tabelului care nu are destule inforamatii adica tabelul cu care dorim sa facem OUTER JOIN. In momentul acela toate liniile din celalalt tabelul vor fi afisate si, daca tabelul cu care se face OUTER JOIN nu are valori pentru anumite linii din celalat tabel atunci acestea vor fi (null). Sa vedem cum arata OUTER JOIN pentru exemplul nostru:

```
1.  SELECT p.nume, m.marca
2.  FROM persoana p, masina m
3.  WHERE p.id_masina = m.id(+);
```

p.nume	m.marca
Andrei	BMW
Ana	Dacia
Maria	Honda
Ion	Audi
Raluca	(null)
Alex	(null)

Deoarece Raluca si Alex nu au masina acestia nu au fost afisati prima data deoarece conditia de join nu era respectata, insa, acum sunt afisati chiar daca nu exista nici-o linie corespunzatoare pentru ei in tabelul MASINA. Valorile coloanei selectate din tabelul MASINA fiind (null) pt ei. Acest tip de join se numeste RIGHT OUTER JOIN deoarece OUTER JOIN-ul are loc pe tabelul din partea dreapta a join-ului ((+) e pus la tabelul din dreapta). Un alt tip asemanator de join este LEFT OUTER JOIN unde se va face aceasi operatie dar pe tabelul din partea stanga:

```
1.  SELECT p.nume, m.marca
2.  FROM persoana p, masina m
3.  WHERE p.id_masina(+) = m.id;
```

p.nume	m.marca
Andrei	BMW
Ana	Dacia
Maria	Honda
Ion	Audi
(null)	Lexus

Si astfel putem observa toate persoanele ce marci de masina folosesc si, la final, toate marcile de masina ce nu sunt folosite. Acest lucru are loc deoarece in momentul acesta programul se uita pe fiecare linie din tabelul MASINA si, daca nu are nici-o legatura cu tabelul PERSOANA, o afiseaza punand valoarea din coloana tabelului PERSOANA ca fiind (null).

IMPORTANT: RIGHT OUTER JOIN si LEFT OUTER JOIN au efecte diferite asupra unui join, in functie de tabelul langa care sunt puse.

Ultimul tip de join este FULL OUTER JOIN ce este combinatia intre LEFT si RIGHT OUTER JOIN:

```
1.  SELECT p.nume, m.marca
2.  FROM persoana p, masina m
3.  WHERE p.id_masina(+) = m.id(+);
```

p.nume	m.marca
Andrei	BMW
Ana	Dacia
Maria	Honda
Ion	Audi
Raluca	(null)
Alex	(null)
(null)	Lexus

FULL OUTER JOIN imi afiseaza atat rezultatele unui RIGHT OUTER JOIN cat si cele ale unui LEFT OUTER JOIN, el fiind defapt cele 2 tipuri de join-uri facute impreuna.

Operatori de Multimi

In SQL tabelul rezultat dintr-o cerere poate fi vazuta ca o multime. Daca 2 cereri returneaza tabele identice (cu exact aceleasi coloane) atunci pe acestea pot fi aplicate urmatorii operatori de multime:

UNION – Reuniunea, imi uneste toate rezultatele celor 2 cereri, eliminand duplicatele.

UNION ALL – La fel ca UNION dar fara a elimina duplicatele.

INTERSECT – Intersectia, imi afiseaza numai rezultatele ce se gasesc in ambele cereri.

MINUS – Diferenta, imi afiseaza rezultatele ce se gasesc in prima dar nu si in a doua cerere.

Pentru a face o operatie de multimi sintaxa este :

```
1.  Cerere1
2.  OPERATOR_DE_MULTIMI
3.  Cerere2;
```

Sa luam de exemplu urmatorul tabel ce contine numele, specia si varsta mai multor animale:

ANIMAL

ID(PK)	NUME	SPECIE	VARSTA
1	Alfie	Caine	3
2	Bobo	Caine	7
3	Bruno	Caine	5
4	Bella	Pisica	2
5	Alfie	Pisica	4
6	Kira	Pisica	1

Sa presupunem ca vrem sa afisam numele tuturor cainilor mai batrani de 5 ani si numele pisicilor mai tinere de 5 ani:

```
1.  SELECT nume
2.  FROM animal
3.  WHERE specie = 'Caine'
4.    AND varsta > 5
5.  UNION
6.  SELECT nume
7.  FROM animal
8.  WHERE specie = 'Pisica'
9.    AND varsta < 5;
```

nume
Bobo
Bella
Kira

Pentru rezolvare am creat 2 cereri ce intorc numele cainilor, respectiv al pisicilor si ne-am folosit de operatorul UNION pentru a reuni cele 2 rezultate, astfel ajungand la rezultatul final.

In contiunare sa afisam toate numele comune de pisici si caini:

```
1.  SELECT nume
2.  FROM animal
3.  WHERE specie = 'Caine'
4.  INTERSECT
5.  SELECT nume
6.  FROM animal
7.  WHERE specie = 'Pisica'
```

nume
Alfie

Pentru rezolvare am intors toate numele de caine intr-o cerere, toate numele de pisica in alta cerere si, folosind operatorul INTERSECT, am intersectat cele 2 multimi ramanand astfel numai numele comune.

In final sa afisam toate numele de caini ce nu sunt si nume de pisica:

```
1.  SELECT nume
2.  FROM animal
3.  WHERE specie = 'Caine'
4.  MINUS
5.  SELECT nume
6.  FROM animal
7.  WHERE specie = 'Pisica'
```

nume
Bobo
Bruno

Pentru rezolvare am luat din nou cele 2 multimi de nume de data aceasta facand diferenta dintre cele 2 multimi folosind operatorul MINUS.

Subcereri

Subcererea in SQL reprezinta o comanda SELECT in interiorul unei alte comenzi SQL. Acestea ne ajuta in cazul cererilor complexe intrucat sunt o modalitate perfecta de a imparti cererea in mai multe bucati mici, astfel ea fiind mai simpla de scris. Ne putem gandi la o subcerere ca la o functie ce returneaza una sau mai multe valori.

Subcererile pot returna valori single-line sau multi-line. In functie de tipul returnat se pot folosi diversi operatori:

- In cazul valorilor single-line se pot folosi operatorii: =, <, <=, >, >=, <>.
- In cazul valorilor multi-line se pot folosi operatorii: IN (valoarea se afla in rezultatul subereri) , ANY (asemanator cu IN), ALL(toate rezulatetle din subcerere sunt echivalente cu valoarea).

ATENTIE: in cazul in care folosim operatorii single line trebuie sa ne asiguram ca rezultatul subcererii este in toate cazurile de tip single-line, altfel va aparea o eroare. Operatorii multi-line se pot folosi si in cazul in care rezultatul subcererii este single-line.

Sa luam ca exemplu tabelul EMPLOYEES si sa scriem o cerere ce afiseaza numele tuturor angajatilor din acelasi departament cu Sigal Tobias folosind subcereri:

```
1.  SELECT last_name
2.  FROM employees
3.  WHERE department_id = ( SELECT department_id
4.                          FROM employees
5.                          WHERE upper(first_name) = 'SIGAL'
6.                            AND upper(last_name) = 'TOBIAS');
```

last_name
Raphaely
Khoo
Baida
Tobias
Himuro
Colmenares

Pentru rezolvare ne-am folosit de o subcerere in clauza WHERE pentru a afla departamentul in care lucreaza Sigal Tobias si apoi am afisat, folosind un operator single-line, toti angajatii ce lucreaza in acelasi departament cu el.

O astfel de subcerere in care tabelele folosite in cerere si in subcerere sunt total independente (tabelele din cerere nu sunt folosite in subcerere) se numeste o subcerere necorelata(nesincornizata).

In continuare sa afisam numele fiecarui angajat ce are macar o ruda(last_name identic) ce lucreaza intr-un alt departament:

```
1.  SELECT e1.first_name
2.  FROM employees e1
3.  WHERE e1.last_name IN ( SELECT e2.last_name
4.                          FROM employees e2
5.                          WHERE e2.department_id <> e1.department_id);
```

first_name
Janette
Steven
Winston
Jonathan

Pentru rezolvare ne-am folosit de o subcerere ce returneaza numele de familie al tuturor angajatiilor ce nu lucreaza in acelasi departament cu angajatul curent, daca printre aceste nume se afla si numele angajatului curent atunci il afisez.

Aceasta subcerea se numeste corelata(sincronizata) deoarece ma folosesc de e1 (ce este declarat in cerere) in interiorul subcererii. La nivel functional nu exista nici-o diferenta dintre o subcerere corelata si una necorelata.