Laboratorul 4

De la BD

Tematica: join - afișarea informațiilor din mai multe tabele

Cuprins

Produsul cartezian a două tabele

Pentru a pune împreună informația din mai multe tabele avem la dispoziție două modalități:

1. utilizarea operatorilor pe mulțimi (gen reuniune, intersecție, diferență) care pun împreună rezultatele a două sau a mai multor interogări - discutați la laboratorul 2
2. utilizarea produsului cartezian si, derivat, a joinului.

Toate frazele SELECT din cadrul laboratoarelor anterioare au utilizat în cadrul clauzei FROM o singură tabelă. Încercați însă, pe rand, următoarele interogări:

```
SELECT * FROM studenti;
```
--cate linii sunt returnate?

```
SELECT * FROM note;
```
--cate linii sunt returnate?

```
SELECT * FROM studenti, note;
```
--cate linii sunt returnate? de ce?

```
SELECT * FROM studenti, note WHERE prenume='Andrei';
```
--cate linii sunt returnate? de ce?

```
SELECT * FROM studenti, note, cursuri;
```
--cate linii sunt returnate? de ce?

Dacă în cadrul clauzei FROM sunt utilizate mai multe tabele, rezultatul este produsul cartezian al tuturor tabelelor. Numarul de linii returnate este produsul numarului de linii ale tabelelor.

Interogările de mai sus în care în clauza FROM se află o listă de tabele este specifică Oracle. Pentru a realiza însă produsul cartezian și în alte SGBD-uri relaționale, se utilizează următoarea sintaxă care este standard limbajului SQL (valabilă și pentru Oracle, testati-o):

```
SELECT * FROM studenti CROSS JOIN note

SELECT * FROM studenti CROSS JOIN note WHERE prenume='Andrei';

SELECT * FROM studenti CROSS JOIN note CROSS JOIN cursuri;
```

Joinul natural a două tabele

Înregistrările dintr-o tabelă își găsesc corespondentul în alt(e) tabel(e) prin intermediul unui atribut ce joacă rol de cheie străină trimițând la un atribut cu rol de cheie (proprietate de identificare unică) în tabela referențiată. De exemplu, în tabela *note* atributul *nr_matricol* are rolul de a identifica studentul din tabela *studenti* iar atributul *id_curs* identifică înregistrarea corespunzătoare din tabelul *cursuri*.

Pentru a identifica notele unui anumit student nu este suficient să solicităm informația din ambele tabele - studenti si note (vezi exemplul anterior unde solicitând notele lui Andrei s-a realizat de fapt un produs cartezian conducând la un rezultat incorect), ci trebuie să specificăm și condiția de join - și anume atributul care joacă rol de cheie străină să fie egal cu atributul care joaca rol de identificare unică în tabelul referențiat:

```
SELECT * FROM studenti, note WHERE studenti.nr_matricol = note.nr_matricol ORDER BY nume;
--cate inregistrari au fost returnate? de ce?
```

Din nou, formularea anterioară este specifică Oracle. Operația care s-a efectuat a fost joinul intern a două tabele, păstrându-se în rezultat acele linii din produsul cartezian care satisfac condiția impusă de clauza WHERE (pentru a înțelege operația de join este util să priviți joinul ca un produs cartezian urmat de selecție; în realitate sistemul tratează operația cu unul dintre algoritmii dedicați joinului - care vor fi explicati la curs). Versiunea admisă însă de standardul SQL și la care aderă cele mai multe sisteme de baze de date relaționale utilizează explicit cuvintele cheie JOIN...ON, după cum urmează:

```
SELECT * FROM studenti JOIN note ON studenti.nr_matricol = note.nr_matricol ORDER BY nume;

SELECT * FROM studenti JOIN note ON studenti.nr_matricol = note.nr_matricol WHERE
prenume='Andrei' --cate inregistrari au fost returnate? de ce?

SELECT * FROM studenti JOIN note ON studenti.nr_matricol = note.nr_matricol WHERE
prenume='Ioana'; --cate inregistrari au fost returnate? de ce?
```

Fiindcă numele atributului cu rol de cheie străină și numele atributului cu rol de identificare unică referențiat coincid în exemplul prezentat, ele au fost prefixate de numele tabelului la care aparțin. Ca alternativă, atunci când numele coloanei după care se face JOIN este identic în ambele tabele, condiţia de egalitate a acestora poate fi omisă dacă se foloseşte operaţia de Join natural:

```
SELECT nume, prenume, valoare FROM studenti NATURAL JOIN note WHERE prenume='Ioana';
```

Alias pentru tabele

De fiecare dată când într-o interogare apar mai multe tabele și ele conțin atribute cu același nume, orice referire la astfel de atribute trebuiesc prefixate de numele tabelei din care se dorește a fi citită valoarea.

Încercați pe rand

```
SELECT nume, prenume, valoare, nr_matricol FROM studenti JOIN note ON
studenti.nr_matricol=note.nr_matricol WHERE prenume = 'Ioana'; --de ce nu functioneaza ?

SELECT nume, prenume, valoare, studenti.nr_matricol FROM studenti JOIN note ON
studenti.nr_matricol = note.nr_matricol WHERE prenume = 'Ioana';
```

In cazul joinului natural, SGBD-ul ştie că între cele două coloane oricum se aplică operatorul de egalitate. In acest caz nu mai trebuie să prefixaţi numele coloanei cu numele tabelei:

```
SELECT nume, prenume, valoare, nr_matricol FROM studenti NATURAL JOIN note WHERE
prenume='Ioana';
```

Orice utilizare a unui atribut a cărui nume se regăsește în mai multe tabele referențiate în interogare trebuie prefixat de numele tabelei (exceptie fac tabelele implicate în join natural). Dacă însă numele tabelelor este incomod a fi repetat, putem să dăm aliasuri simple acestora - specificate în clauza FROM imediat dupa numele tabelei. Identificarea atributelor în acest caz se face utilizand aliasul ca și prefix:

```
SELECT nume, prenume, valoare, s.nr_matricol FROM studenti s JOIN note n ON s.nr_matricol =
n.nr_matricol;
```

ATENȚIE: introducerea de aliasuri pentru numele tabelelor necesită utilizarea acestora oriunde e necesar în cadrul interogării.

Încercați:

```
SELECT nume, prenume, valoare, studenti.nr_matricol FROM studenti s JOIN note n ON
s.nr_matricol = n.nr_matricol;
```

Join extern

Să afișăm în mod distinct numele studenților din joinul între tabelele *studenti* și *note*:

```
SELECT DISTINCT nume FROM studenti s NATURAL JOIN note n;
```

Câte înregistrări au fost returnate? Și câte nume distincte sunt în tabela studenti de fapt? De unde diferența?

Sunt cazuri în care nu toate înregistrările dintr-o tabelă au corespondent în altă tabelă. În exemplul anterior, nu toți studenții au note. În cadrul joinului anterior s-a văzut cum înregistrările fără corespondent nu fac parte din mulțimea rezultat.

In cazul în care se doreşte realizarea unui join între tabelele studenţi şi note, se poate observa că există studenţi care nu au încă note (fiind primul semestru, ei sunt înscrişi la cursurile din anul întâi dar nu au primit încă note). Spunem despre tabela note că este deficitară în informaţii deoarece nu conţine toate numerele matricole existente în tabela studenţi. Dacă se doreşte ca în rezultat să se regăsească şi studenţii fără note, se va utiliza un join extern.

Joinul extern poate fi de mai multe tipuri:

- LEFT OUTER JOIN este utilizat atunci când se doreşte preluarea tuturor informaţiilor din tabela din stânga operatorului join (deci join in acest caz nu este comutativ) şi, eventual, aceste informaţii să fie completate (dacă se poate) cu informaţii din tabela deficitară în informaţii aflată în dreapta operatorului join. De exemplu: `SELECT DISTINCT nume FROM studenti s LEFT OUTER JOIN note n ON s.nr_matricol = n.nr_matricol;`

- RIGHT OUTER JOIN este similar dar de aceasta dată tabela deficitară în informaţii se va afla în dreapta. Se vor prelua toate informaţiile din tabela din dreapta şi se vor completa (atunci când se poate) cu informaţii din tabela din stânga. Ca şi exemplu putem schimba ordinea tabelelor din LEFT OUTER JOIN-ul anterior: `SELECT DISTINCT nume FROM note n RIGHT OUTER JOIN studenti s ON s.nr_matricol = n.nr_matricol;`

Cele două versiuni de join extern se pot realiza în Oracle și cu următoarea sintaxă:

```
SELECT DISTINCT nume FROM studenti s, note n WHERE s.nr_matricol = n.nr_matricol(+);
```

Simbolul (+) se adaugă pe ramura deficitară în informații.

- FULL OUTER JOIN este reuniunea celor de mai sus: sunt afişate toate informaţiile şi completate (eventual) cu informaţii din cealalta tabelă.

Ce valori primesc în urma unui join extern înregistrările care nu au corespondent?

Alte detalii la: http://www.techonthenet.com/oracle/joins.php

Schita a tipurilor de join: http://i.stack.imgur.com/udQpD.jpg

Joinul mai multor tabele

În interogările anterioare am identificat notele tuturor studenților utilizând un join între două tabele - studenti și note. Cum aflăm însă la ce discipline au fost puse notele? Adăugăm în interogare un al treilea tabel, împreună cu condiția de join necesară:

```
SELECT nume, prenume, titlu_curs, valoare
   FROM studenti s
              JOIN note n ON s.nr_matricol=n.nr_matricol
              JOIN cursuri c ON c.id_curs=n.id_curs
```

Self-joinul - joinul unei tabele cu ea insăși

Uneori, modul în care este structurată informația în tabele ne pune în situația de a căuta corespondentul unor înregistrări chiar în același tabel. În cazul de față, un exemplu este determinarea colegilor de grupă pentru un anumit student. Oricând suntem în situația de a apela de două ori la un tabel într-o singură interogare, spunem că realizăm un *self-join*. În cazul unui self join tabela în cauză este apelată de două ori cu aliasuri distincte, ca și când ar exista două instanțe identice ale aceluiași tabel.

Exemplu:

```
SELECT s.nume || ' ' || s.prenume || ' este coleg cu ' || colegi.nume || ' ' || colegi.prenume AS "Colegi de gru
```

```
FROM studenti s JOIN studenti colegi ON s.grupa=colegi.grupa AND s.an=colegi.an
WHERE s.prenume='Andrei'
```

Cum eliminăm înregistrarea în care studentul apare coleg cu el însuși?

Exercitii

1. Afişaţi studenţii şi notele pe care le-au luat si profesorii care le-au pus acele note.
2. Afisati studenţii care au luat nota 10 la materia 'BD'. Singurele valori pe care aveţi voie să le hardcodaţi în interogare sunt valoarea notei (10) şi numele cursului ('BD').
3. Afisaţi profesorii (numele şi prenumele) impreuna cu cursurile pe care fiecare le ţine.
4. Modificaţi interogarea de la punctul 3 pentru a fi afişaţi şi acei profesori care nu au încă alocat un curs (In prealabil rulaţi din nou scriptul aflat la adresa: http://profs.info.uaic.ro/~vcosmin/BD/facultate.sql pentru a adăuga aceşti noi profesori).
5. Modificaţi interogarea de la punctul 3 pentru a fi afişate acele cursuri ce nu au alocate încă un profesor (după ce aţi rulat scriptul aflat la adresa: http://profs.info.uaic.ro/~vcosmin/BD/facultate.sql).
6. Modificaţi interogarea de la punctul 3 astfel încât să fie afişaţi atat profesorii care nu au nici un curs alocat cât şi cursurile care nu sunt încă predate de nici un profesor (după ce aţi rulat scriptul aflat la adresa: http://profs.info.uaic.ro/~vcosmin/BD/facultate.sql).
7. In tabela studenti există studenţi care s-au nascut în aceeasi zi a săptămânii. De exemplu, Cobzaru George şi Pintescu Andrei s-au născut amândoi într-o zi de marti. Construiti o listă cu studentii care s-au născut in aceeaşi zi grupându-i doi câte doi în ordine alfabetică a numelor (de exemplu in rezultat va apare combinatia Cobzaru-Pintescu dar nu va apare şi Pintescu-Cobzaru). Lista va trebui să conţină doar numele de familie a celor doi împreună cu ziua în care cei doi s-au născut. Evident, dacă există şi alţi studenti care s-au născut marti, vor apare si ei in combinatie cu cei doi amintiţi mai sus. Lista va fi ordonată în funcţie de ziua săptămânii în care s-au născut si, în cazul în care sunt mai mult de trei studenţi născuţi în aceeaşi zi, rezultatele vor fi ordonate şi după numele primei persoane din listă [pont: interogarea trebuie să returneze 10 rânduri].
8. Sa se afiseze, pentru fiecare student, numele colegilor care au luat nota mai mare ca ei la fiecare dintre cursuri. Formulati rezultatele ca propozitii (de forma "Popescu Gigel a luat nota mai mare ca Vasilescu Ionel la matera BD."). Dati un nume corespunzator coloanei [pont: interogarea trebuie să returneze 118 rânduri].
