EXTRAGEREA DATELOR DIN MAI MULT DE O TABELA

Join

Join-ul este folosit când o cerere SQL necesită date din mai multe tabele din baza de date.

Articolele dintr-o tabelă pot fi unite cu articolele din altă tabelă în functie de valorile comune existente în coloanele corespunzătoare.

Vom prezenta:

1. Equi-join
2. Non-equi-join

Equi-join

Pentru a lista angajații si departamentele unde aceştia lucrează, vom compara valorile din coloana department_id a angajatului cu aceleaşi valori din department_id din tabela departments. Relația dintre tabela employees si departments este un equi-join, în care valorile din coloana department_id din ambele tabele sunt egale. (Operatorul de comparatie folosit este =)

O condiție de join este specificată în clauza WHERE:

```
SELECT column(s)
FROM tables
WHERE join condition is ...
```

Pentru a face join pe cele două tabele introducem:

```
select first_name, last_name, department_name, salary
  from employees, departments
  where employees.department_id = departments.department_id
  order by salary;
```

Vom observa că acum fiecare angajat are listat numele departamentului lui. Liniile din employees sunt combinate cu liniile din departments şi sunt întoarse doar liniile pentru care valorile ***employees.department_id*** şi ***departments.department_id*** sunt egale.

Observați că în condiția de join se specifică **numele coloanei precedat de numele tabelei**. Aceasta este o necesitate când numele coloanelor sunt aceleaşi în ambele tabele. Este necesar să specificăm exact care coloane sunt referite.

Această necesitate este de asemenea aplicată coloanelor care pot fi ambigue în clauzele SELECT sau ORDER BY.

Pentru a recunoaşte diferenţele dintre coloana department_id din employees şi coloana department_id din departments, introducem:

```
    select        first_name,        last_name,        department_name,
departments.department_id
        from employees, departments
        where employees.department_id = departments.department_id
        order by departments.department_id;
```

Observaţi că fiecărui număr de departament din tabela departments i se face join pentru a se potrivi cu numerele de departament din tabela employees. De exemplu, doi angajaţi lucrează în departamentul 110 - Accounting. Prin urmare ACCOUNTING este afisat pentru fiecare angajat din acel departament.

Folosirea alias-urilor de tabela

Etichete temporare (sau alias-uri) pot fi folosite în clauza FROM. Aceste nume temporare sunt valide doar în instrucţiunea SELECT curenta. Alias-urile de tabele trebuie de asemenea să fie specificate în clauza SELECT.

Alias-urile de tabela sunt folosite în urmatorul context:

```
select e.first_name, e.last_name, d.department_name, d.department_id
        from employees e, departments d
        where e.department_id = d.department_id
        order by d.department_id;
```

Alias-urile de tabele pot fi de lungime de maxim 30 de caractere.

În absenţa unei condiţii WHERE, fiecare linie din employees este unită în ordine cu fiecare linie din departments.

Când o condiţie de join este invalidă sau este omisă, rezultatul este omis şi toate combinaţiile de linii vor fi listate.

Un produs tinde să genereze un număr mare de linii şi rezultatul său este rar folosit. Trebuie inclusă o condiţie de join într-o clauza WHERE, în afară de cazul în care este necesară combinarea tuturor liniilor din toate tabelele.

Non-Equi-Join

Relaţia dintre tabelele employees si job_grades este un **non-equi-join**, în care nici o coloana din employees nu corespunde direct cu o coloana din job_grades. Relaţia este

obținută folosind un operator, altul decât operatorul de egalitate (=).Pentru a evalua gradația unui anajat, salariul lui trebuie să fie între salariul minim şi salariul maxim. Operatorul BETWEEN este folosit pentru a construi condiția, introducem:

```
select e.first_name, e.last_name, j.grade_level
          from employees e, job_grades j
          where e.salary between j.lowest_sal and j.highest_sal;
```

Unde lowest_sal, highest_sal, grade_level sunt câmpuri în tabela job_grades.

Reguli pentru join-ul tabelelor

Pentru a face join pe trei tabele este necesar sa construim două conditii de join. Pentru a face join pe patru tabele sunt necesare trei conditii de join.

O regulă simplă este:

numarul minim de condiții de join = numarul de tabele - 1

Sintaxa

```
SELECT [DISTINCT] {[tabela].* | expresie [alias], ...}
FROM tabela [alias], ...
WHERE [condiție de join] ...
[AND condiție de linie] ...
[OR alta condiție de linie]
GROUP BY {expresie | coloana}
HAVING {condiție de grup}
ORDER BY {expresie | coloana} [ASC | DESC]
```

Observatii

- Se pot specifica condiții de join împreună cu alte conditii (non join);
- De asemenea trebuie să fiți atenți la precedența operatorilor când folosiți OR.

Join-ul unei tabele cu ea însăşi

Este posibilă folosirea etichetelor de tabele (alias-urilor) pentru a face join unei tabele cu ea însăşi, ca si cum ar fi două tabele separate. Aceasta permite ca articolelor dintr-o tabela să li se facă join cu articolele din aceeaşi tabelă.

Următoarea cerere listează toți angajații care câştigă mai puțin ca sefii lor :

```
select    e.first_name, e.last_name, e.salary "salariu angajat",
          m.first_name, m.last_name, m.salary "salariu manager"
```

```
from employees e, employees m
where e.manager_id=m.employee_id
and e.salary < m.salary;
```

Observați că FROM se referă la employees de două ori si ca urmare employees are asociat câte un alias pentru ambele cazuri - E si M. Este util ca alias- urile asociate să aibă înțeles, de exemplu E înseamnă angajați (employees) şi M înseamnă şefi (managers).

Clauza join poate fi exprimata:

"unde numărul sefului angajatului este acelaşi cu numărul de angajat al sefului".

Operatori de multimi

In cadrul acestui capitol vor fi discutate reuniunea, intersecția şi diferența.

Reuniunea, intersecția si diferența sunt folositoare în construcția cererilor care se referă la tabele diferite. Ele combină rezultatele a două sau mai multe declarații SELECT în unul singur. O cerere poate fi formată din două sau mai multe declarații SQL înlănțuite prin operatori de mulțimi. Operatorii de mulțimi sunt numiți join-uri verticale, deoarece join-ul nu se face în raport cu articolele din cele două tabele, ci în raport cu coloanele.

Reuniunea

Pentru a lista toate articolele diferite generate de fiecare din cereri, introducem:

UNION ALL

Pentru a lista toate articolele (inclusiv duplicatele) generate de fiecare din cereri, introducem:

```
select first_name, last_name, department_id
        from employees
        where department_id=10
UNION ALL
select first_name, last_name, department_id
        from employees
        where department_id=50;
```

Intersecția

Pentru a lista doar articolele generate de ambele cereri, introducem:

```
select job_id ,department_id
from employees
where department_id=110
intersect
select job_id ,department_id
from employees
where department_id=90;
```

Diferența

Pentru a lista toate articolele generate de prima cerere, care nu sunt în a două cerere, introducem:

```
select job_id ,department_id
from employees
where department_id=110
minus
select job_id ,department_id
from employees
where department_id=90;
```

Este posibil să se construiască cereri cu mai mulți operatori de mulțimi. Dacă sunt folosiți mai mulți operatori de mulțimi, ordinea execuției pentru declarațiile SQL este de sus în jos. Parantezele pot fi folosite pentru a schimba ordinea execuției.

ORDER BY

ORDER BY poate fi folosită o singură dată într-o cerere ce foloseste operatori de mulțimi. Dacă este folosită, clauza ORDER BY trebuie plasată la sfârşitul cererii. Coloanele din ORDER BY pot să fie referite prin pozițiile relative din lista din SELECT.

```
     select first_name, last_name, department_id
        from employees
        where department_id=10
UNION
select first_name, last_name, department_id
        from employees
        where department_id=50
order by 2;
```

Observati că în clauza ORDER BY un număr (2) este folosit pentru a indica poziția coloanei last_name în lista din SELECT. Aceasta înseamnă că liniile vor fi sortate în ordine ascendenă a numelui angajaților.

Reguli pentru folosirea operatorilor de mulțimi

1. Declarațiile SELECT trebuie să aibă acelaşi număr de coloane.

2. Coloanele corespunzătoare trebuie să aibă acelaşi tip (corespondența este poziţională).
3. Liniile duplicate sunt automat eliminate (nu poate fi folosit DISTINCT).
4. Numele coloanelor din prima cerere apar în rezultat.
5. Clauza ORDER BY apare la sfârşitul declarației.
6. Blocurile de cerere sunt executate de sus în jos.
7. Operatorii de mulțimi multipli pot fi folosiți cu paranteze, dacă este necesară schimbarea ordinii de execuție.

Exerciții

1. Afişați numele tuturor angajaților şi numele departamentului lor, în ordinea numelui departamentelor.
2. Afişați numele şi departamentul angajaților al căror salariu lunar este mai mare ca 1500.
3. Afişați angajații şi tipul de job.
4. Listați doar angajații cu gradatia C.
5. Listați toți angajații din Toronto.
6. Afişați numele angajaților, funcția, salariul, gradația şi numele departamentului pentru toți angajații din companie în afară de President. Sortați după salariu, afişând mai întâi salariul cel mai mare.
7. Afişați detalii ca la exercițiul 6 pentru angajații care câştigă mai mult de 36000$ **pe an** ordonate dupa funcție.
8. Afişați departamentul care nu are nici un angajat.
9. Afişați toti angajații (nume si cod) împreună cu numele şi codul şefului.
10. Modificați soluția de la intrebarea 9 pentru a afisa şi pe KING care nu are sef.
11. Găsiți funcțiile care au fost ocupate (hire_date intre ianuarie şi iulie) în prima jumătate a anului 1993 şi în timpul aceleiaşi perioade în 1994 (union).
12. Găsiți toți angajații care au venit in companie înaintea şefilor lor.