UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiect nr. 154/323 cod SMIS – 4428 cofinanțat de prin Fondul European de Dezvoltare Regională “Investiții pentru viitorul dumneavoastră”.

Programul Operaţional Sectorial Creşterea Competitivităţii Economice - POS CCE

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Baze de date

19. Interogări la un singur tabel

Introducere

- Cererile de interogare SQL folosesc în exclusivitate comanda SELECT, fiind utilizate atât pentru interogarea obiectelor create de utilizator, cât şi a celor sistem. Sintaxa comenzii este următoarea :
- **SELECT [DISTINCT,ALL] [schema.table.]expresion expr_alias**

  FROM [schema.table@dblink] table_alias

  [WHERE condition]

  [START WITH condition][CONNECT BY condition]

  [UNION,UNION ALL,INTERSECT,MINUS][SELECT command]

  [GROUP BY expresion][HAVING condition]

  [ORDER BY expresion(position)] [ASC,DESC]

  [FOR UPDATE OF schema.table.column] [NOWAIT]

Parametrii de interogare

- Parametrii comenzii au următoarea semnificație(cei din paranteze sunt opționali):
- ***DISTINCT*** - returnează numai o înregistrare în cazul în care comanda găseşte rânduri duplicate
- ***ALL*** – returnează toate înregistrările simple şi duplicate (selectează de asemenea toate coloanele tabelelor din clauza FROM)
- ***schema.table*** – reprezintă shema de identificare a tabelei(view-lui) specificată prin *user.table_name*
- ***expresion*** – reprezintă un nume de coloană sau o expresie care poate folosi funcții sistem.
- ***expr_alias*** – este un nume alocat unei expresii care va fi folosit în formatarea coloanei (apare în antetul listei)
- ***dblink*** – reprezintă numele complet sau parțial de identificare a unei baze de date (database.domain@connection_qualifier)
- ***table_alias*** – este un nume alocat unei tabele(view) care va fi folosit în cereri corelate
- ***WHERE condition*** – reprezintă o clauză (înlănțuire de condiții) care trebuie să fie îndeplinită în criteriul de selecție a înregistrărilor

Parametrii de interogare- continuare

- ***START WITH condition*** – stabileşte criteriul de selecție pentru prima înregistrare
- ***GROUP BY expresion*** – stabileşte criteriile de grupare a înregistrărilor(numele coloanelor folosite în criteriul de grupare)
- ***CONNECT BY condition*** – stabileşte o ierarhie de selecție a înregistrărilor
- ***HAVING condition*** – restricționează înregistrările din grup la anumite condiții
- ***UNION,UNION ALL,INTERSECT,MINUS*** – combină rândurile selectate de mai multe comenzi *SELECT,* prin aplicarea anumitor restricții
- ***ORDER BY expresion(position)*** – ordonează înregistrările selectate după coloanele din expresie, sau în ordinea coloanelor specificate prin poziție
- ***FOR UPDATE OF*** – face o blocare (lock) a înregistrărilor în vederea modificării anumitor coloane
- ***NOWAIT*** – returnează controlul userului, dacă comanda aşteaptă eliberarea unei înregistrări blocată de un alt user

Interogări pe clauza WHERE

- Clauza WHERE poate compara valori de coloană , valori literale, expresii aritmetice (sau funcții) şi poate avea patru tipuri de parametri:
- nume de coloană
- operator de comparație
- operator de negație
- lista de valori

Operatori de comparație

- ***operatori logici***

Operator	Semnificație
=	egal cu
>	mai mare decât
>=	mai mare sau egal
<	mai mic decât
<=	mai mic sau egal

- ***operatori SQL***

Operator	Semnificație
BETWEEN..AND...	intre două valori(inclusiv)
IN(list)	compară cu o listă de valori
LIKE	compară cu un model de tip caracter
IS NULL	este o valoare nulă

Operatori de negație

- ***operatori logici***

Operator	Semnificație
!=	diferit de(VAX,UNIX,PC)
^=	diferit de(IBM)
<>	diferit de(toate OS)
NOT NUMECOL=	diferit de
NOT NUMECOL>	mai mic sau egal

- ***operatori SQL***

Operator	Semnificație
NOT BETWEEN	nu se află intre două valori date
NOT IN	nu se află intr-o listă dată de valori
NOT LIKE	diferit de un şir
IS NOT NULL	nu este o valoare nulă

Ordinea de precedență a operatorilor

- Dacă operatorii au precedența egală, atunci ei se evaluează de la stânga la dreapta.Precedența operatorilor logici este următoarea:

1. Operatorii de comparație şi operatorii SQL au precedența egală:
   =, >= , <>, BETWEEN...AND, IN, LIKE, IS NULL.
2. NOT (pentru a inversa rezultatul unei expresii logice, de exemplu
   SELECT ….WHERE not(sal>2000))
3. AND
4. OR.

Pentru a fi siguri de ordinea de execuție a două operații, se recomandă folosirea parantezelor rotunde.

Exemple:

- SQL>SELECT * FROM angajati ;
- SQL>SELECT id_dep,den_dep FROM departamente;
- SQL>SELECT id_ang ecuson, nume,salariu*12+nvl(comision,0) venit_anual
  FROM angajati;
- SQL>SELECT id_ang ecuson, nume,data_ang,salariu FROM angajati WHERE
  id_dep=10;
- SQL>SELECT id_ang ecuson, nume,functie, salariu*12+nvl(comision,0) venit
  FROM angajati WHERE id_ang IN (7499,7902,7876) ORDER BY nume;
- SQL>SELECT id_ang ecuson, nume, functie, data_ang FROM angajati
  WHERE data_ang LIKE '%80';
- SQL>SELECT id_ang ecuson, nume, functie, salariu FROM angajati
  WHERE (comision=0 OR comision IS NULL) AND id_dep=20
  ORDER BY nume;

Interogări cu variabile substituite în SQL*Plus

- Cererile SQL pot fi executate folosind anumiți parametri, care se mai numesc şi variabile substituite.

  1. Variabile substituite cu un singur ampersand (&)

- O astfel de variabilă se defineste **&nume_variabila** şi este un parametru care se va introduce de la tastatură în timpul execuției comenzii în care este utilizată. Parametrul cu un singur ampersand trebuie introdus de fiecare dată, chiar dacă este folosit de mai multe ori în aceeaşi comanda SQL.

Exemplu:

- **SQL> SELECT id_ang,nume,functie,salariu FROM angajati**

  WHERE id_sef=&ecuson_sef;

Interogări cu variabile substituite în SQL*Plus- continuare

2. Variabile substituite cu dublu ampersand (&&)

- Spre deosebire de variabila cu un singur ampersand, o variabilă cu dublu ampersand va fi stocată şi va putea fi apelată pe toată sesiunea de lucru.
- Definirea se face similar **&&nume_variabila** şi va fi cerută o singură dată, folosirea ei de mai multe ori, în cadrul comenzii, se face apelând-o cu **&nume_variabila**, aşa cum se vede în exemplul următor :

Exemplu:

- **SQL> SELECT nume,functie,&& venit venit_lunar FROM**

  angajati WHERE &venit>2000;

Interogări cu variabile substituite în SQL*Plus- continuare

3. Variabile definite cu ACCEPT

- Când definim o variabilă cu ampersand, totdeauna promtul va fi numele variabilei.
- Folosind comanda ACCEPT, se poate redefini promtul şi chiar se pot ascunde caracterele introduse de la tastatură(facilitate utilă în cazul unei parole).

În exemplul următor, se vor edita următoarele comenzi în fişierul **c:\temp\functia.sql** :

- **SQL>ACCEPT functie_sef CHAR PROMPT 'Introduceti functia sefului:' ;**
- **SQL>SELECT nume,salariu,comision FROM angajati WHERE**

  functie='&functie_sef';
- **SQL>UNDEFINE functie_sef;**

Fişierul se va rula în SQL*Plus :

- **SQL> @c:\temp\functia.sql**

Interogări cu variabile substituite în SQL*Plus- continuare

4. Variabile definite cu DEFINE şi resetate cu UNDEFINE

- Variabilele astfel definite cu DEFINE nu vor mai afişa promtul atunci când sunt setate şi rămân setate până când vor fi resetate cu comanda UNDEFINE.

În exemplul următor, vom seta variabila procent_prima =15%;

- **SQL> DEFINE procent_prima= 1.15;**
- **SQL>SELECT nume,salariu, salariu*&procent_prima prima FROM angajati WHERE id_dep=20;**
- **SQL> UNDEFINE procent_prima;**