=Variabile de substituție SQL\*Plus=

Variabilele de substituție au rolul de a suplini constante, condiții utilizate în clauza WHERE, nume de campuri utilizate în clauza ORDER BY, nume de tabele sau chiar întregi fraze SELECT.

Variabilele de substituție sunt precedate de simbolul & iar la execuția interogării SQLPlus va solicita de la utilizator valorile necesare.

Exemplu 1:

SELECT \*

FROM studenti

WHERE an=&an AND nume LIKE '&nume'

Exemplu 2:

SELECT nume, &camp\_afisare

FROM studenti

WHERE &conditie2

ORDER BY &camp\_sortare;

Executați interogările de mai sus de mai multe ori inserand diverse valori pentru variabilele de substituție. Verificați rezultatul.

Variabilele de substituție pot fi predefinite și instanțiate, caz în care la execuția interogării nu se mai solicită valoarea:

DEFINE camp=prenume

SELECT nume, &camp

FROM studenti

ORDER BY &camp;

Variabila este stocată pe parcursul sesiunii curente până la eliminarea ei cu comanda UNDEFINE sau până la închiderea sesiunii.

UNDEFINE camp

Comenzile DEFINE si UNDEFINE sunt specifice utilitarului SQL\*Plus și nu limbajului SQL.

=Comenzi DML - Modificare date=

Până acum s-au utilizat în cadrul laboratoarelor doar comenzi de interogarea a datelor - adică fraza SELECT - care sunt parte tot din cadrul componentei DML (Data Manipulation Language) a limbajului SQL.

În această secțiune exemplificăm comenzile de modificare a datelor.

==Adăugarea înregistrărilor==

Inserarea de noi înregistrări în baza de date se realizează cu ajutorul comenzii INSERT care poate lua două forme:

* INSERT INTO nume\_tabel [ (lista\_coloane) ] VALUES (lista\_valori)

Ex: INSERT INTO cursuri VALUES(30, 'Inv. automata', 3, 1, 5);

Ex: INSERT INTO cursuri (ID\_curs, titlu\_curs) VALUES(31, 'NoSQL');

* INSERT INTO nume\_tabel [ (lista\_coloane) ] fraza\_select

Ex: INSERT INTO note SELECT nr\_matricol, 30, 10, SYSDATE FROM student WHERE an=3;

===SECVENTE===

De multe ori suntem puși în situația de a crea atribute sintetice care să joace rol de cheie, deci să ia valori unice în cadrul unei tabele. Secvențele sunt obiecte care ne ajută să generăm automat numere întregi unice în momentul inserării înregistrilor.

Sintaxa creării unei secvențe:

CREATE SEQUENCE nume\_secventa

[INCREMENT BY n] -- pasul

[START WITH n1] -- valoarea de start

[MAXVALUE n\_max | NOMAXVALUE] -- valoarea de oprire

[MINVALUE n\_min | NOMINVALUE] -- valoarea minima (listele pot fi siruri descrescatoare)

[CYCLE | NOCYCLE] -- daca se continua generarea cand se ajunge la min/max

Comanda NEXTVAL generează următorul număr din șir iar CURRVAL obține valoarea curentă. NEXTVAL trebuie executată înainte ca lista să conțină vreun număr.

Exemplu:

CREATE SEQUENCE s1 INCREMENT BY 2 START WITH 10 MAXVALUE 15;

SELECT s1.NEXTVAL FROM DUAL; --executati in mod repetat comanda; cand ridica eroare?

===Exercitii===

Cum poate fi utilizată o secvență la inserare?

Răspundeți creând o secvență care sa vă ajute sa inserați noi cursuri cu id unic, cu intrari consecutive crescătoare cu pasul 1. Inserati 3 cursuri noi cu id-ul generat de secventa.

==Actualizarea înregistrărilor==

* UPDATE nume\_tabel [alias] SET nume\_coloana1 = valoare1 [, nume\_coloana2 = valoare2...] [WHERE conditie]

Ex: UPDATE studenti SET bursa=bursa\*0.15 WHERE bursa<300;

* UPDATE nume\_tabel [alias] SET (nume\_coloana1 [,nume\_coloana2]...) = (subinterogare) [WHERE conditie

===Exercitii===

#Actualizati valoarea bursei pentru studentii care au măcar o notă de 10. Acestia vor primi ca bursa 500RON.

#Toti studentii primesc o bursa egala cu 100\*media notelor lor. Efectuati modificarile necesare.

==Ștergerea înregistrărilor==

* DELETE FROM nume\_tabel [WHERE conditie]

Ex: DELETE FROM note WHERE valoare < 5;

===Exercitii===

Stergeti toti studentii care nu au nici o nota.

=Comenzi DDL - creare/modificare structuri de date=

Aceste comenzi sunt menite a crea sau a șterge tabele (sau alte tipuri de obiecte), precum și a modifica structura tabelelor (obiectelor) existente. Comenzile necesare fac parte din componenta DDL (Data Definition Language) a limbajului SQL.

==CREATE TABLE AS==

Comenzile de creare a tabelelor au fost pe larg explicate la curs, cu diversele constrângeri care pot fi impuse asupra datelor la momentul creării. Aici tratăm doar comanda CREATE cu subinterogări.

CREATE TABLE nume\_tabel [(col1, col2,...)] [AS] fraza\_select;

Comanda este similară cu cea dedicată creării viewului, diferența fiind că la crearea unui tabel, acesta este umplut cu datele returnate de interogare.

===Exercitii===

Executati comanda ROLLBACK. Creati apoi o tabelă care să stocheze numele, prenumele, bursa si mediile studentilor.

==ALTER TABLE==

Structura tabelelor create deja poate fi modificată, chiar dacă tabelul conține deja date. Modificările pot să vizeze adăugarea de noi coloane, ștergerea unor coloane existente, modificarea definiției unei coloane (tipul și constrângerile impuse), redenumirea unei coloane, redenumirea tabelului, adăugarea sau ștergerea de constângeri (chei candidat, cei primare, chei străine, constrângeri NOT NULL, CHECK).

Găsiți în continuare comenzile necesare:

* ALTER TABLE nume\_tabel ADD (col1 definitie\_col1 [,col2 definitie\_col2....])

Ex: ALTER TABLE cursuri ADD (abreviere CHAR(2) NULL, descriere VARCHAR(40) DEFAULT 'curs obligatoriu');

* ALTER TABLE nume\_tabel DROP COLUMN col

Ex: ALTER TABLE cursuri DROP COLUMN descriere;

* ALTER TABLE nume\_tabel MODIFY (col1 definitie\_col1 [,col2 definitie\_col2....])

Ex: ALTER TABLE profesori MODIFY (nume VARCHAR(20), prenume VARCHAR(20));

* ALTER TABLE nume\_tabel RENAME COLUMN col\_veche TO col\_noua

Ex: ALTER TABLE note RENAME COLUMN valoare TO nota;

* ALTER TABLE nume\_vechi\_tabel RENAME TO nume\_nou\_tabel

Ex: ALTER TABLE profesori RENAME TO cadre\_didactice;

* ALTER TABLE nume\_tabel ADD CONSTRAINT nume\_constrangere definitie\_constrangere;

Ex: ALTER TABLE studenti ADD CONSTRAINT pk\_studs PRIMARY KEY (nr\_matricol);

Ex: ALTER TABLE note ADD CONSTRAINT fk\_studs FOREIGN KEY (nr\_matricol)

REFERENCES studenti(nr\_matricol) ON DELETE CASCADE;

* ALTER TABLE nume\_tabel DROP CONSTRAINT nume\_constrangere;

Ex: ALTER TABLE note DROP CONSTRAINT fk\_studs;

===Exercitii===

Executati din nou scriptul de creare de aici: http://profs.info.uaic.ro/~vcosmin/BD/facultate.sql

#Adăugați constrângerile de tip cheie primară pentru tabelele Studenti, Profesori, Cursuri.

#Adăugați constrângerile referențiale pentru tabelele Note și Didactic. La ștergerea unui profesor din tabela Profesori, în tabela Didactic id-ul profesorului șters va deveni null. La stergerea unui curs din tabela Cursuri, in tabela Didactic va fi stearsă înregistrarea care referențiază cursul șters. Scrieți comenzi de ștergere înregistrări pentru tabelele referențiate și studiați comportamentul.

#Impuneți constrângerea ca un student să nu aibă mai mult de o notă la un curs.

#Impuneți constrângerea ca valoarea notei să fie cuprinsă între 1 și 10.

=Tranzacții=

O tranzacție reprezintă un grup de comenzi de modificare de date (DML) care trebuiesc executate împreună, pentru a garanta consistența datelor. Eșuarea oricărei comenzi din cadrul unei tranzacții determină revenirea la starea inițială, dinaintea tranzacției.

O tranzacție începe de la prima comandă DML executată și se încheie la întâlnirea uneia dintre comenzile COMMIT și ROLLBACK, la întâlnirea unei comenzi DDL, la închiderea sesiunii sau la o eroare a sistemului.

Comanda ROLLBACK încheie tranzacția readucând baza de date la starea de dinaintea începerii tranzacției. O cădere a sistemului rezultă tot într-o comandă de tip ROLLBACK, adică revenirea la starea de dinaintea începerii tranzacției; este modalitatea prin care este protejată integritatea datelor. Restul situațiilor enumerate mai sus care determină încheierea unei tranzacții, fac ca modificările efectate asupra datelor să fie permanente, fără posibilitatea revenirii la o stare anterioară.

Imediat ce o tranzacție s-a încheiat, prima comandă DML lansată marchează nceputul uneia noi.

În cadrul acestui laborator ați executat comanda ROLLBACK înainte de a experimenta cu comanda CREATE TABLE, adică înainte de prima comandă de tip DDL lansată. Rezultatul a fost revenirea bazei de date la starea existentă la începutul sesiunii de lucru. Dacă nu ar fi fost lansată comanda ROLLBACK, modificările efectuate asupra datelor ar fi devenit permanente în momentul executării comenzii DDL (așa cum s-a specificat mai sus, orice comandă DDL încheie tranzacția marcând modificările ca fiind permanente).

Pe parcursul unei tranzacții pot fi adăugați niște marcatori/indicatori cu ajutorul cărora să putem reveni la stări intermediare. Adăugarea unui indicator se realizează cu ajutorul comenzii SAVEPOINT nume\_indicator iar revenirea la starea bazei de date din acel moment se realizează cu ajutorul comenzii ROLLBACK TO SAVEPOINT nume\_indicator.

Orice comandă DDL este considerată a forma o tranzacție.

===Exerciții===

# Asigurati-va că începeți o nouă tranzacție.

# Ștergeți toate înregistrările din tabela Profesori.

# Inserați un profesor.

# Marcați starea curentă ca 's1'.

# Schimbați numele profesorului inserat.

# Vizualizați datele.

# Reveniți la starea s1.

# Vizualizați datele.

# Reveniția la starea de dinaintea ștergerii.