Baze de date-Anul 1
Laborator 7 SQL

Limbajul de manipulare a datelor (LMD)
Limbajul de control al datelor (LCD)

- Comenzile SQL care alcătuiesc **LMD** permit:
  - regăsirea datelor (*SELECT*);
  - adăugarea de noi înregistrări (*INSERT*);
  - modificarea valorilor coloanelor din înregistrările existente (*UPDATE*);
  - adăugarea sau modificarea condiţionată de înregistrări (*MERGE*);
  - suprimarea de înregistrări (*DELETE*).

- **Tranzacţia** este o unitate logică de lucru, constituită dintr-o secvenţă de comenzi care trebuie să se execute atomic (ca un întreg) pentru a menţine consistenţa bazei de date.
- *Server*-ul *Oracle* asigură consistenţa datelor pe baza tranzacţiilor, inclusiv în eventualitatea unei anomalii a unui proces sau a sistemului. Tranzacţiile oferă mai multă flexibilitate şi control în modificarea datelor.
- Comenzile SQL care alcătuiesc **LCD** sunt:
  - **ROLLBACK** – pentru a renunţa la modificările aflate în aşteptare se utilizează instrucţiunea *ROLLBACK*. În urma execuţiei acesteia, se încheie tranzacţia, se anulează modificările asupra datelor, se restaurează starea lor precedentă şi se eliberează blocările asupra liniilor.
  - **COMMIT** - determină încheierea tranzacţiei curente şi permanentizarea modificărilor care au intervenit pe parcursul acesteia. Instrucţiunea suprimă toate punctele intermediare definite în tranzacţie şi eliberează blocările tranzacţiei.

    Obs: O comandă **LDD (*CREATE*, *ALTER*, *DROP*)** determină un **COMMIT** implicit.
  - **SAVEPOINT** - Instrucţiunea *SAVEPOINT* marchează un punct intermediar în procesarea tranzacţiei. În acest mod este posibilă împărţirea tranzacţiei în subtranzacţii. Această instrucţiune nu face parte din standardul *ANSI* al limbajului *SQL*.

I. Comanda *INSERT*

1. Inserări mono-tabel

Comanda INSERT are următoarea sintaxă simplificată:

INSERT INTO *obiect* [***AS*** *alias*] [(*nume_coloană* [, *nume_coloană* …])]
{***VALUES*** ({*expr* | ***DEFAULT***} [, {*expr* | ***DEFAULT***} …])
| *subcerere*}

Subcererea specificată în comanda *INSERT* returnează linii care vor fi adăugate în tabel.

Dacă în tabel se introduc linii prin intermediul unei subcereri, coloanele din lista *SELECT* trebuie să corespundă, ca **număr şi tip**, celor precizate în clauza *INTO*. În absenţa unei liste de coloane în clauza *INTO*, subcererea trebuie să furnizeze valori pentru fiecare atribut al obiectului destinaţie, respectând ordinea în care acestea au fost definite.

Observaţii (tipuri de date):

- Pentru claritate, este recomandată utilizarea unei liste de coloane în clauza *INSERT*.
- În clauza *VALUES*, valorile de tip caracter şi dată calendaristică trebuie incluse între apostrofuri. Nu se recomandă includerea între apostrofuri a valorilor numerice, întrucât aceasta ar determina conversii implicite la tipul *NUMBER*.
- Pentru introducerea de valori speciale în tabel, pot fi utilizate funcţii.

Adăugarea unei linii care va conţine valori *null* se poate realiza în mod:

- implicit, prin omiterea numelui coloanei din lista de coloane;
- explicit, prin specificarea în lista de valori a cuvântului cheie *null*

În cazul şirurilor de caractere sau al datelor calendaristice se poate preciza şirul vid (' ').

Observaţii (erori):

Server-ul *Oracle* aplică automat toate tipurile de date, domeniile de valori şi constrângerile de integritate. La introducerea sau actualizarea de înregistrări, pot apărea erori în următoarele situaţii:

- nu a fost specificată o valoare pentru o coloană *NOT NULL*;
- există valori duplicat care încalcă o constrângere de unicitate;
- a fost încălcată constrângerea de cheie externă sau o constrângere de tip *CHECK*;
- există o incompatibilitate în privinţa tipurilor de date;
- s-a încercat inserarea unei valori având o dimensiune mai mare decât a coloanei corespunzătoare.

2. Inserari multi-tabel

O inserare multi-tabel presupune introducerea de linii calculate pe baza rezultatelor unei subcereri, într-unul sau mai multe tabele.

Pentru o astfel de inserare, în versiunile anterioare lui *Oracle9i* erau necesare *n* operaţii independente *INSERT INTO…SELECT…*, unde *n* reprezintă numărul tabelelor destinaţie. Aceasta presupunea *n* procesări ale aceleiaşi surse de date şi, prin urmare, creşterea de *n* ori a timpului necesar procesului.

Sintaxa comenzii *INSERT* în acest caz poate fi:

- Pentru inserări necondiţionate:

INSERT ALL INTO… [***INTO***…]
subcerere;

- Pentru inserări condiţionate:

INSERT [***ALL*** | ***FIRST***]
WHEN *condiţie* ***THEN INTO***…
[***WHEN*** *condiţie* ***THEN INTO***…
[***ELSE INTO*** …]]
subcerere;

- *ALL* determină evaluarea tuturor condiţiilor din clauzele *WHEN*. Pentru cele a căror valoare este *TRUE*, se inserează înregistrarea specificată în opţiunea *INTO* corespunzătoare.

- *FIRST* determină inserarea corespunzătoare primei clauze *WHEN* a cărei condiţie este evaluată *TRUE*. Toate celelalte clauze *WHEN* sunt ignorate.

Exerciţii [I]

1. Să se creeze tabelele *EMP_pnu*, *DEPT_pnu* (în care şirul de caractere “pnu”, *p* reprezintă prima literă a prenumelui, iar *nu* reprezintă primele două litere ale numelui), prin copierea structurii şi conţinutului tabelelor *EMPLOYEES*, respectiv *DEPARTMENTS*.

*CREATE TABLE EMP_pnu AS SELECT * FROM employees;*
*CREATE TABLE DEPT_pnu AS SELECT * FROM departments;*

2. Listaţi structura tabelelor sursă şi a celor create anterior. Ce se observă?

3. Listaţi conţinutul tabelelor create anterior.

4. Pentru introducerea **constrângerilor de integritate**, executaţi instrucţiunile LDD indicate în continuare.

ALTER TABLE emp_pnu
ADD CONSTRAINT pk_emp_pnu PRIMARY KEY(employee_id);

ALTER TABLE dept_pnu
ADD CONSTRAINT pk_dept_pnu PRIMARY KEY(department_id);

ALTER TABLE emp_pnu
ADD CONSTRAINT fk_emp_dept_pnu
FOREIGN KEY(department_id) REFERENCES dept_pnu(department_id);

Obs: Ce constrângere nu am implementat?

5. Să se insereze **departamentul 300**, cu numele ***Programare*** în ***DEPT_pnu***.
   Analizaţi cazurile, precizând care este soluţia corectă şi explicând erorile celorlalte variante. Pentru a anula efectul instrucţiunii(ilor) corecte, utilizaţi comanda *ROLLBACK*.

```
--a)
INSERT INTO DEPT_pnu
VALUES (300, 'Programare');

--b)
INSERT INTO DEPT_pnu (department_id, department_name)
VALUES (300, 'Programare');

--c)
INSERT INTO DEPT_pnu (department_name, department_id)
VALUES (300, 'Programare');
```

--d)
INSERT INTO DEPT_pnu (department_id, department_name, location_id)
VALUES (300, 'Programare', null);
--e)
INSERT INTO DEPT_pnu (department_name, location_id)
VALUES ('Programare', null);
Executaţi varianta care a fost corectă de două ori. Ce se obţine şi de ce?

6. Să se insereze un angajat corespunzător departamentului introdus anterior în tabelul *EMP_pnu*, precizând valoarea *NULL* pentru coloanele a căror valoare nu este cunoscută la inserare (metoda implicită de inserare). Determinaţi ca efectele instrucţiunii să devină permanente.

Atenţie la constrângerile *NOT NULL* asupra coloanelor tabelului!

7. Să se mai introducă un angajat corespunzător **departamentului 300**, precizând după numele tabelului lista coloanelor în care se introduc valori (metoda explicita de inserare). Se presupune că data angajării acestuia este cea curentă (*SYSDATE*). Salvaţi înregistrarea.

   INSERT INTO emp_pnu (hire_date, job_id, employee_id, last_name, email, department_id)

   VALUES (sysdate, 'sa_man', 278, 'nume_278', 'email_278', 300);

   COMMIT ;

8. Este posibilă introducerea de înregistrări prin intermediul subcererilor (specificate în locul tabelului). Ce reprezintă, de fapt, aceste subcereri? Să se analizeze următoarele comenzi *INSERT*:

INSERT INTO emp_pnu (employee_id, last_name, email, hire_date, job_id, salary,
commission_pct)
VALUES (252, 'Nume252', 'nume252@emp.com',SYSDATE, 'SA_REP', 5000, NULL);

SELECT employee_id, last_name, email, hire_date, job_id, salary, commission_pct
FROM emp_pnu
WHERE employee_id=252;

ROLLBACK;

INSERT INTO
(SELECT employee_id, last_name, email, hire_date, job_id, salary,
commission_pct
FROM emp_pnu)
VALUES (252, 'Nume252', 'nume252@emp.com',SYSDATE, 'SA_REP', 5000, NULL);

```
SELECT employee_id, last_name, email, hire_date, job_id, salary, commission_pct
FROM emp_pnu
WHERE employee_id=252;

ROLLBACK;
```

Încercaţi dacă este posibilă introducerea unui angajat, precizând pentru valoarea *employee_id* o subcerere care returnează (codul maxim +1).

```
INSERT INTO (SELECT employee_id, last_name, hire_date, job_id, email
             FROM emp_pnu)
        VALUES (  (SELECT max(employee_id) + 1
                   FROM  emp_pnu
                   ),  'nume_nou', sysdate, 'sa_man',  'email@pnu.com'
                );

SELECT * FROM emp_pnu;

ROLLBACK;
```

9. **Creaţi** un nou tabel, numit *EMP1_PNU*, care va avea aceeaşi structură ca şi *EMPLOYEES*, dar nici o înregistrare. **Copiaţi** în tabelul *EMP1_PNU* salariaţii (din tabelul *EMPLOYEES*) al căror comision depăşeşte 25% din salariu.

10. **Inseraţi** o nouă înregistrare în tabelul *EMP_PNU* care să totalizeze salariile, să facă media comisioanelor, iar câmpurile de tip dată să conţină data curentă şi câmpurile de tip caracter să conţină textul 'TOTAL'. Numele şi prenumele angajatului să corespundă utilizatorului curent (*USER*). Pentru câmpul *employee_id* se va introduce valoarea 0, iar pentru *manager_id* şi *department_id* se va da valoarea null.

```
INSERT INTO emp_pnu
    SELECT 0,USER,USER, 'TOTAL', 'TOTAL',SYSDATE,
        'TOTAL', SUM(salary), ROUND(AVG(commission_pct)), null, null
    FROM employees;

SELECT * FROM emp_pnu;
ROLLBACK;
```

11. Să se creeze un fişier (*script file*) care să permită introducerea de înregistrări în tabelul *EMP_PNU* în mod interactiv. Se vor cere utilizatorului: **codul**, **numele**, **prenumele** si **salariul angajatului**. Câmpul ***email*** se va completa automat prin concatenarea primei litere din prenume şi a primelor 7 litere din nume.

Executati script-ul pentru a introduce 2 inregistrari in tabel.

INSERT INTO emp_pnu (employee_id, first_name, last_name, email, hire_date, job_id, salary)

VALUES(&cod, '&&prenume', '&&nume', substr('&prenume',1,1) || substr('&nume',1,7), sysdate, 'it_prog',&sal);

UNDEFINE prenume;

UNDEFINE nume;

12. Creaţi 2 tabele ***emp2_pnu*** şi ***emp3_pnu*** cu aceeaşi structură ca tabelul *EMPLOYEES*, dar fără înregistrări (acceptăm omiterea constrângerilor de integritate). Prin intermediul unei singure comenzi, copiaţi din tabelul *EMPLOYEES*:

- în tabelul *EMP1_PNU* salariaţii care au salariul mai mic decât 5000;
- în tabelul *EMP2_PNU* salariaţii care au salariul cuprins între 5000 şi 10000;
- în tabelul *EMP3_PNU* salariaţii care au salariul mai mare decât 10000.

Verificaţi rezultatele, apoi ştergeţi toate înregistrările din aceste tabele.

13. Să se creeze tabelul ***EMP0_PNU*** cu aceeaşi structură ca tabelul *EMPLOYEES* (fără constrângeri), dar fără nici o înregistrare. Copiaţi din tabelul *EMPLOYEES*:

- în tabelul *EMP0_PNU* salariaţii care lucrează în departamentul 80;
- în tabelul *EMP1_PNU* salariaţii care au salariul mai mic decât 5000;
- în tabelul *EMP2_PNU* salariaţii care au salariul cuprins între 5000 şi 10000;
- în tabelul *EMP3_PNU* salariaţii care au salariul mai mare decât 10000.

Dacă un salariat se încadrează în tabelul emp0_pnu atunci acesta nu va mai fi inserat şi în alt tabel (tabelul corespunzător salariului său).

*CREATE TABLE emp0_pnu AS SELECT * FROM employees;*

DELETE FROM emp0_pnu;

```
INSERT FIRST
        WHEN department_id = 80 THEN
           INTO emp0_pnu
        WHEN salary < 5000 THEN
           INTO emp1_pnu
        WHEN salary > = 5000 AND salary <= 10000 THEN
           INTO emp2_pnu
        ELSE
           INTO emp3_pnu
SELECT * FROM employees;

SELECT * FROM emp0_pnu;
SELECT * FROM emp1_pnu;
SELECT * FROM emp2_pnu;
SELECT * FROM emp3_pnu;
```

II. Comanda UPDATE

Sintaxa simplificată a comenzii ***UPDATE*** este:

UPDATE nume_tabel [alias]
SET col1 = expr1[, col2=expr2]
[WHERE conditie];

sau

UPDATE nume_tabel [alias]
SET (col1,col2,...) = (subcerere)
[WHERE conditie];

Observaţii:

- de obicei pentru identificarea unei linii se foloseşte o condiţie ce implică cheia primară;
- dacă nu apare clauza *WHERE* atunci sunt afectate toate liniile tabelului specificat;
- cazurile în care instrucţiunea *UPDATE* nu poate fi executată sunt similare celor în care eşuează instrucţiunea *INSERT*. Acestea au fost menţionate anterior.

Exerciţii [II]

14. Măriţi **salariul tuturor angajaţilor** din tabelul *EMP_PNU* cu 5%. Vizualizati, iar apoi anulaţi modificările.

```
UPDATE emp_pnu
SET salary = salary * 1.05;

SELECT * FROM emp_pnu;
ROLLBACK;
```

15. Schimbaţi **jobul tuturor salariaţilor** din departamentul 80 care au comision, în 'SA_REP'. Anulaţi modificările.

16. Să se promoveze Douglas Grant la manager în departamentul 20, având o creştere de salariu cu 1000$.

17. Schimbaţi **salariul** şi **comisionul** celui mai prost plătit salariat din firmă, astfel încât să fie egale cu salariul si comisionul şefului său.

```
UPDATE  emp_pnu e
SET  (salary,commission_pct) = ( SELECT salary,commission_pct
                                 FROM emp_pnu
                                WHERE employee_id = e.manager_id)
WHERE salary = (SELECT min(salary)
                FROM  emp_pnu);
ROLLBACK;
```

18. Să se **modifice adresa de e-mail** pentru angajaţii care câştigă cel mai mult în departamentul în care lucrează astfel încât acesta să devină iniţiala numelui concatenată cu prenumele. Dacă nu are prenume atunci în loc de acesta apare caracterul '.' . Anulaţi modificările.

19. Pentru fiecare departament **să se mărească salariul** celor care au fost angajaţi primii astfel încât să devină media salariilor din companie.

```
UPDATE emp_pnu d
SET salary = (SELECT avg(salary)
              FROM emp_pnu
             )
WHERE hire_date = (SELECT min(hire_date)
                   FROM emp_pnu
                   WHERE department_id = d.department_id
                  );

ROLLBACK;
```

III. Comanda DELETE

Sintaxa simplificată a comenzii ***DELETE*** este:

```
DELETE FROM nume_tabel
[WHERE conditie];
```

Daca nu se specifica nici o conditie, vor fi şterse toate liniile din tabel.

Exercitii [III]

20. Ştergeţi **toate înregistrările** din tabelul *DEPT_PNU*. Ce înregistrări se pot şterge? Anulaţi modificările.

21. Suprimaţi departamentele care nu au angajati. Anulaţi modificările.

Exerciţii [LMD, LCD]

22. Să se mai introducă o linie in tabelul ***DEPT_PNU***.

23. Să se marcheze un punct intermediar in procesarea tranzacţiei (*SAVEPOINT p).*

24. Să se şteargă din tabelul *DEPT_PNU* departamentele care au codul de departament cuprins intre 160 si 200 . Listaţi conţinutul tabelului.

25. Să se renunţe la cea mai recentă operaţie de ştergere, fără a renunţa la operaţia precedentă de introducere. (*ROLLBACK TO p).* Determinaţi ca modificările să devină permanente.