LABORATOR 6 SQL

I. Limbajul de control al datelor (LCD). COMMIT, SAVEPOINT, ROLLBACK.

II. Limbajul de prelucrare a datelor (LMD). INSERT, UPDATE, DELETE, MERGE.

I. Limbajul de control al datelor (LCD). COMMIT, SAVEPOINT, ROLLBACK.

- **Comanda COMMIT** permanentizează modificările care au fost realizate de tranzacţia curentă (o tranzacţie este un set de comenzi LMD); comanda suprimă toate punctele intermediare definite în tranzacţie şi eliberează blocările tranzacţiei.

  Observaţie:
  Sistemul realizează COMMIT implicit:
  - la închiderea normală a unui client *Oracle* (de exemplu SQL*Plus),
  - după fiecare comandă LDD (CREATE, ALTER, DROP).

- **Comanda SAVEPOINT** marchează un punct intermediar în procesarea tranzacţiei. În acest mod este posibilă împărţirea tranzacţiei în subtranzacţii.

  Comanda *SAVEPOINT* are sintaxa următoare:

```
SAVEPOINT nume_pct_intermediar;
```

- **Comanda ROLLBACK** permite renunţarea la modificările efectuate; aceasta determină încheierea tranzacţiei, anularea modificărilor asupra datelor şi restaurarea stării lor precedente.

  Comanda ROLLBACK are sintaxa următoare:

```
ROLLBACK [TO SAVEPOINT nume_punct_salvare];
```

  Observaţii:
  - sistemul realizează ROLLBACK implicit dacă se închide anormal (defecţiune hardware sau software, pană de curent etc.);
  - nicio comanda LDD (CREATE, ALTER; DROP) nu poate fi anulată.

1. Ce efect are următoarea secvenţă de instrucţiuni?

```
CREATE TABLE test_*** AS
     SELECT * FROM departments;

SELECT * FROM test_***;

SAVEPOINT a;

DELETE FROM test_***;
```

```
INSERT INTO test_*** VALUES (300, 'Economic ',100,1000);

INSERT INTO  test_*** VALUES (350, 'Cercetare ',200,2000);

SAVEPOINT b;

INSERT INTO  test_*** VALUES (400, 'Juritic',150,3000);

SELECT * FROM  test_***;

ROLLBACK TO b;

SELECT * FROM  test_***;

ROLLBACK TO  a;

INSERT INTO  test_***
VALUES (500, 'Contabilitate',175,1500);

COMMIT;

SELECT * FROM test_***;

DROP TABLE test_***;
```

II. Limbajul de prelucrare a datelor (LMD). INSERT, UPDATE, DELETE, MERGE.

- **Comanda INSERT**

  Sintaxa comenzii ***INSERT***:

  - pentru inserarea unei singure linii

    ```
    INSERT INTO nume_tabel [(col1,col2,...)]
    VALUES (expresie1, expresie2, ...);
    ```

  - pentru inserarea liniilor rezultat ale unei comenzi SELECT

    ```
    INSERT INTO nume_tabel [(col1,col2,...)]
    comanda_SELECT;
    ```

  - pentru captarea informaţiilor inserate

```
INSERT INTO nume_tabel
VALUES listă_valori
RETURNING coloană1,coloană2, …
INTO :variabilă_host1, :variabilă_host2, …;
```

Observaţii:

- lista de coloane (dacă este precizată) trebuie să se potrivească ca număr şi tip de date cu lista de expresii;
- în loc de tabel (*nume_tabel*), inserarea se mai poate face şi prin intermediul unei vizualizări;
- coloanele omise din lista de inserare, vor primi valoarea NULL sau valoarea implicită setată la nivel de tabel;
- posibile probleme la inserare:
  - lipsa de valori pentru coloane NOT NULL,
  - nepotrivirea listei de coloane cu cea de expresii,
  - valori duplicate ce încalcă o constrângere de unicitate,
  - încălcarea unei constrângeri de integritate referenţială,
  - încălcarea unei constrângeri de tip CHECK,
  - nepotrivire de tip de date,
  - valoare prea mare pentru coloana respectivă;
- dacă se foloseşte expresia DEFAULT, atunci valoarea inserată este NULL sau valoarea implicită setată la nivel de tabel.

- **Comanda DELETE**

```
DELETE FROM nume_tabel
[WHERE conditie]
[RETURNING coloană1,coloană2, …
 INTO :variabilă_host1, :variabilă_host2, … ];
```

- **Comanda UPDATE**

```
UPDATE nume_tabel [alias]
SET    col1 = expr1[, col2=expr2]
[WHERE conditie]
[RETURNING coloană1,coloană2, …
 INTO :variabilă_host1, :variabilă_host2, … ];
```

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.

2. Creaţi tabele *emp_*** şi *dept_***, având aceeaşi structură şi date ca şi tabelele *employees*, respectiv *departments*.

```
CREATE TABLE nume_tabel AS subcerere;
```

3. Afişaţi toate înregistrările din cele două tabele create anterior.

4. Ştergeţi toate înregistrările din cele 2 tabele create anterior. Salvaţi modificările realizate.

```
DELETE FROM nume_tabel;
```

5. Listaţi structura tabelului *employees* şi comparaţi-o cu structura tabelului *emp_***. Ce observaţi?

6. Exemplificaţi câteva dintre erorile care pot să apară la inserare şi observaţi mesajul întors de sistem.

- lipsa de valori pentru coloane NOT NULL (coloana *department_name* este definită NOT NULL)

```
INSERT INTO dept_*** (department_id, location_id)
VALUES (200, 2000);
```

- nepotrivirea listei de coloane cu cea de expresii

```
INSERT INTO dept_***
VALUES (200, 2000);

INSERT INTO dept_*** (department_id, department_name,location_id)
VALUES (200, 2000);
```

- nepotrivirea tipului de date

```
INSERT INTO dept_*** (department_id, location_id)
VALUES ('D23', 2000);
```

- valoare prea mare pentru coloană

```
INSERT INTO dept_*** (department_id, location_id)
VALUES (15000, 2000);
```

7. Inseraţi în tabelul *emp_*** salariaţii (din tabelul *employees*) al căror comision depăşeşte 25% din salariu.

8. Creaţi tabelele *emp1_***, *emp2_*** şi *emp3_*** cu aceeaşi structură ca tabelul *employees*, dar fără date. Inseraţi, utilizând o singură comandă INSERT, informaţiile din tabelul *employees* astfel:

- în tabelul *emp1_*** salariaţii care au salariul mai mic sau egal decât 6000;
- în tabelul *emp2_*** salariaţii care au salariul cuprins între 6000 şi 10000;
- în tabelul *emp3_*** salariaţii care au salariul mai mare sau egal decât 10000.

Verificaţi rezultatele, apoi ştergeţi toate înregistrările din aceste tabele.

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

Clauza FIRST a comenzii INSERT determină inserarea corespunzătoare primei clauze WHEN a cărei condiţie este evaluată TRUE. Toate celelalte clauze WHEN sunt ignorate.

```
INSERT ALL/FIRST
   WHEN condiţie THEN
      INTO nume_tabel
   WHEN condiţie THEN
      INTO nume_tabel
   …
   ELSE
      INTO nume_tabel
 comandă_select;
```

9. Creaţi tabelul *emp0_*** cu aceeaşi structură ca tabelul *employees* şi fără date. Inseraţi, utilizând o singură comandă INSERT, informaţiile din tabelul *employees* astfel:

- în tabelul *emp0_*** salariaţii care lucrează în departamentul 80;
- în tabelul *emp1_*** salariaţii care au salariul mai mic sau egal decât 6000 şi nu lucrează în departamentul 80;
- în tabelul *emp2_*** salariaţii care au salariul cuprins între 6001 şi 10000 şi nu lucrează în departamentul 80;
- în tabelul *emp3_*** salariaţii care au salariul mai mare decât 10001 şi nu lucrează în departamentul 80.

10. Inseraţi o linie nouă în tabelul *dept_***,* folosind valori introduse de la tastatură.

Observaţie: Comenzile care permit interacţiunea dintre mediul *SQL*Plus* şi utilizator sunt:

ACC[EPT] *variabila* [tip] [PROMPT *text*]	Citeşte o linie de intrare şi o stochează într-o variabilă utilizator.
DEF[INE] [*variabila*] | [*variabila* = *text*]	Specifică o variabilă utilizator, căreia i se poate atribui o valoare de tipul CHAR. Fără argumente, comanda listează valorile şi tipurile tuturor variabilelor.
PROMPT [*text*]	Afişează mesajul specificat sau o linie vidă pe ecranul utilizatorului.
UNDEF[INE] *variabila*	Permite ştergerea variabilelor definite de utilizator.

Exemplu:

```
ACCEPT n PROMPT 'n='

SELECT last_name, job_id, salary
FROM   employees
WHERE  salary >&n;
```

11. Inseraţi o linie nouă în tabelul *dept_****. Salvaţi într-o variabilă de legătură codul departamentului nou introdus. Afişaţi valoarea menţinută în variabila respectivă. Anulaţi efectele tranzacţiei.

Observaţie: Variabilele de legătură (*bind variables*) se creează în SQL*Plus şi pot fi referite în SQL sau PL/SQL. Acestea au următoarele funcţionalităţi:

- permit afişarea în SQL*Plus a valorilor utilizate în blocuri PL/SQL (variabilele declarate în blocurile PL/SQL, nu pot fi afişate în SQL*Plus);
- pot fi utilizate în mai multe blocuri PL/SQL, permiţând comunicarea între acestea.

Comenzile care permit crearea şi afişarea variabilelor de legătură:

VAR[IABLE] [*variabila*]	declară o variabilă de legătură care poate fi referită în blocurile PL/SQL. Dacă nu se specifică nici un argument, comanda VARIABLE listează toate variabilele de legătură create în sesiunea curentă
PRI[NT] *variabila*	afişează valoarea unei variabile de legătură

Exemplu:

```
VARIABLE  v_nume VARCHAR2(20)

BEGIN
   SELECT last_name
   INTO   :v_nume
   FROM   employees
   WHERE  employee_id = 100;
END;
/

PRINT  v_nume
```

12. Ştergeţi toate înregistrările din tabelele *emp_*** şi *dept_****. Inseraţi în aceste tabele toate înregistrările corespunzătoare din *employees*, respectiv *departments*. Permanentizaţi tranzacţia.

13. Eliminaţi departamentele care nu au angajaţi. Anulaţi efectele tranzacţiei.

14. Ştergeţi angajaţii care nu conduc departamente sau alţi angajaţi şi care nu au avut alte joburi în trecut. Anulaţi efectele tranzacţiei.

15. Ştergeţi un angajat al cărui cod este dat de la tastatură. Menţineţi numele acestuia într-o variabilă de legătură. Afişaţi valoarea acestei variabile. Anulaţi modificările.

16. Măriţi salariul tuturor angajaţilor (din tabelul *emp_****) cu 5%. Anulaţi modificările.

17. Pentru angajaţii departamentului 80 s-a luat decizia că nu vor mai avea comision, iar comisionul pe care l-au avut până în acest moment va fi integrat în salariu. Realizaţi aceste modificări. Verificaţi rezultatul obţinut. Anulaţi efectele tranzacţiei.

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

19. Modificaţi jobul şi departamentul angajatului având codul 114, astfel încât să fie la fel cu cele ale angajatului având codul 205. Anulaţi efectele tranzacţiei.

```
UPDATE emp_***
SET (job_id, department_id) = (SELECT job_id, department_id
                               FROM emp_***
                               WHERE employee_id = 205)
WHERE employee_id = 114;
ROLLBACK;
```

20. Schimbaţi salariul şi comisionul celui mai prost plătit salariat din firmă, astfel încât să fie egale cu salariul si comisionul directorului. Anulaţi efectele tranzacţiei.

```
UPDATE emp_***
SET (salary, commission_pct) = (SELECT salary, commission_pct
                                FROM emp_***
                                WHERE manager_id IS NULL)
WHERE salary = (SELECT MIN(salary)
                FROM emp_***);
ROLLBACK;
```

21. Pentru fiecare departament să se mărească salariul celor care au fost angajaţi primii astfel încât să devină media salariilor din companie. Anulaţi efectele tranzacţiei.

22. Modificaţi valoarea emailului pentru angajaţii care câştigă cel mai mult în departamentul în care lucrează astfel încât acesta să devină iniţiala numelui concatenată cu „_“ concatenat cu prenumele. Anulaţi efectele tranzacţiei.

23. Modificaţi comanda de la exerciţiul anterior astfel încât actualizarea coloanei *email* să fie realizată doar pentru angajatul având codul 200. Menţineţi numele şi emailul acestuia în două variabile de legătură. Anulaţi efectele tranzacţiei.

24. Măriţi cu 1000 salariul unui angajat al cărui cod este introdus de la tastatură.

25. a. Ştergeţi din tabelul *emp_**** toţi angajaţii care lucrează în departamentul 80.

b. Micşoraţi cu 5% salariul celor care lucrează în departamentul 50.

c. Folosiţi comanda MERGE pentru a sincroniza datele din tabelul *emp_**** cu cele din tabelul *employees* (înregistrările şterse anterior vor fi inserate, iar înregistrările pentru care a fost modificat salariul vor fi actualizate).

Observaţie: Instrucţiunea *MERGE* permite inserarea sau actualizarea condiţionată a datelor dintr-un tabel al bazei. Instrucţiunea efectuează *UPDATE* dacă înregistrarea există deja în tabel sau *INSERT* dacă înregistrarea este nouă. În acest fel, se pot evita instrucţiunile *UPDATE* multiple.

Comanda ***MERGE*** are următoarea sintaxă simplificată:

```
MERGE INTO  nume_tabel [alias]
USING {tabel | vizualizare | subcerere} [alias]
ON (condiţie)
WHEN MATCHED THEN
     UPDATE SET
          coloana_1 = {expr_u1 | DEFAULT},…,
          coloana_n = {expr_un | DEFAULT}
WHEN NOT MATCHED THEN
     INSERT  (coloana_1,…, coloana_n)
     VALUES (expr_i1,…, expr_in);
```

```
MERGE INTO emp_***  a
  USING employees b
  ON (a.employee_id = b.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      a.first_name=b. first_name,
      a.last_name=b.last_name,
      a.email=b.email,
      a.phone_number=b.phone_number,
      a.hire_date= b.hire_date,
      a.job_id= b.job_id,
```

```
    a.salary = b.salary,
    a.commission_pct= b.commission_pct,
    a.manager_id= b.manager_id,
    a.department_id= b.department_id
WHEN NOT MATCHED THEN
  INSERT VALUES(b.employee_id, b.first_name, b.last_name, b.email,
                b.phone_number, b.hire_date, b.job_id, b.salary,
                b.commission_pct, b.manager_id, b.department_id);
```