

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 tranzacția curentă (o tranzacție este un set de comenzi LMD); comanda suprimă toate punctele intermediare definite în tranzacție și eliberează blocările tranzacției.

Observație:

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 tranzacției. În acest mod este posibilă împărțirea tranzacției în subtranzacții.

Comanda *SAVEPOINT* are sintaxa următoare:

```
SAVEPOINT nume_pct_intermediar;
```

- **Comanda ROLLBACK** permite renunțarea la modificările efectuate; aceasta determină încheierea tranzacției, anularea modificărilor asupra datelor și restaurarea stării lor precedente.

Comanda *ROLLBACK* are sintaxa următoare:

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

Observații:

- sistemul realizează ROLLBACK implicit dacă se închide anormal (defecțiune hardware sau software, până de curent etc.);
- nicio comandă LDD (CREATE, ALTER, DROP) nu poate fi anulată.

1. Ce efect are următoarea secvență de instrucțiuni?

```
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, 'Juritice',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 informațiilor inserate

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

Observații:

- lista de coloane (dacă este precizată) trebuie să se potrivească ca număr și tip de date cu lista de expresii;
- în loc de tabel (*nume_tabel*), inserarea se mai poate face și 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 referențială,
 - încălcarea unei constrângeri de tip CHECK,
 - nepotrivire de tip de date,
 - valoare prea mare pentru coloana respectivă;
- dacă se folosește 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];
```

Observații:

- de obicei pentru identificarea unei linii se folosește o condiție ce implică cheia primară;
- dacă nu apare clauza *WHERE* atunci sunt afectate toate liniile tabelului specificat.

2. Creați tabele *emp_**** și *dept_****, având aceeași structură și date ca și tabelele *employees*, respectiv *departments*.

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

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

4. Ștergeți toate înregistrările din cele 2 tabele create anterior. Salvați modificările realizate.

```
DELETE FROM nume_tabel;
```

5. Listați structura tabelului *employees* și comparați-o cu structura tabelului *emp_****. Ce observați?

6. Exemplificați câteva dintre erorile care pot să apară la inserare și observați 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. Inserați în tabelul *emp_**** salariații (din tabelul *employees*) al căror comision depășește 25% din salariu.

8. Creați tabelele *emp1_****, *emp2_**** și *emp3_**** cu aceeași structură ca tabelul *employees*, dar fără date. Inserați, utilizând o singură comandă INSERT, informațiile din tabelul *employees* astfel:

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

Verificați rezultatele, apoi ștergeți toate înregistrările din aceste tabele.

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

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

```
INSERT ALL/FIRST
  WHEN condiție THEN
    INTO nume_tabel
  WHEN condiție THEN
    INTO nume_tabel
  ...
  ELSE
    INTO nume_tabel
  comandă_select;
```

9. Creați tabelul *emp0_**** cu aceeași structură ca tabelul *employees* și fără date. Inșerați, utilizând o singură comandă *INSERT*, informațiile din tabelul *employees* astfel:

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

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

Observație: Comenzile care permit interacțiunea dintre mediul *SQL*Plus* și utilizator sunt:

ACC[EPT] <i>variabila</i> [tip] [PROMPT <i>text</i>]	Citește o linie de intrare și o stochează într-o variabilă utilizator.
DEF[INE] [<i>variabila</i>] [<i>variabila</i> = <i>text</i>]	Specifică o variabilă utilizator, căreia i se poate atribui o valoare de tipul CHAR. Fără argumente, comanda listează valorile și tipurile tuturor variabilelor.
PROMPT [<i>text</i>]	Afișează mesajul specificat sau o linie vidă pe ecranul utilizatorului.
UNDEF[INE] <i>variabila</i>	Permite ștergerea variabilelor definite de utilizator.

Exemplu:

```
ACCEPT n PROMPT 'n='

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

11. Inserați o linie nouă în tabelul *dept_****. Salvați într-o variabilă de legătură codul departamentului nou introdus. Afișați valoarea menținută în variabila respectivă. Anulați efectele tranzacției.

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

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

Comenzile care permit crearea și afișarea variabilelor de legătură:

VAR[iable] [<i>variabila</i>]	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] <i>variabila</i>	afișează valoarea unei variabile de legătură

Exemplu:

```
VARIABLE v_num VARCHAR2(20)

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

/

PRINT v_num
```

12. Ștergeți toate înregistrările din tabelele *emp_**** și *dept_****. Inserați în aceste tabele toate înregistrările corespunzătoare din *employees*, respectiv *departments*. Permanentizați tranzacția.
13. Eliminați departamentele care nu au angajați. Anulați efectele tranzacției.

14. Ștergeți angajații care nu conduc departamente sau alți angajați și care nu au avut alte joburi în trecut. Anulați efectele tranzacției.
15. Ștergeți un angajat al cărui cod este dat de la tastatură. Mențineți numele acestuia într-o variabilă de legătură. Afișați valoarea acestei variabile. Anulați modificările.
16. Măriți salariul tuturor angajaților (din tabelul *emp_****) cu 5%. Anulați modificările.
17. Pentru angajații 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. Realizați aceste modificări. Verificați rezultatul obținut. Anulați efectele tranzacției.
18. Să se promoveze Douglas Grant la manager în departamentul 20, având o creștere de salariu cu 1000\$. Anulați efectele tranzacției.
19. Modificați jobul și departamentul angajatului având codul 114, astfel încât să fie la fel cu cele ale angajatului având codul 205. Anulați efectele tranzacției.

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

20. Schimbați salariul și comisionul celui mai prost plătit salariat din firmă, astfel încât să fie egale cu salariul și comisionul directorului. Anulați efectele tranzacției.

```
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 angajați primii astfel încât să devină media salariilor din companie. Anulați efectele tranzacției.
22. Modificați valoarea emailului pentru angajații care câștigă cel mai mult în departamentul în care lucrează astfel încât acesta să devină inițiala numelui concatenată cu „_” concatenat cu prenumele. Anulați efectele tranzacției.

23. Modificați comanda de la exercițiul anterior astfel încât actualizarea coloanei *email* să fie realizată doar pentru angajatul având codul 200. Mențineți numele și emailul acestuia în două variabile de legătură. Anulați efectele tranzacției.
24. Măriți cu 1000 salariul unui angajat al cărui cod este introdus de la tastatură.
25. a. Ștergeți din tabelul *emp_**** toți angajații care lucrează în departamentul 80.
b. Micșorați cu 5% salariul celor care lucrează în departamentul 50.
c. Folosiți comanda *MERGE* pentru a sincroniza datele din tabelul *emp_**** cu cele din tabelul *employees* (înregistrările șterse anterior vor fi inserate, iar înregistrările pentru care a fost modificat salariul vor fi actualizate).

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

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

```
MERGE INTO  nume_tabel [alias]
USING {tabel | vizualizare | subcerere} [alias]
ON (condiție)
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);
```