

LABORATOR 9

LMD: INSERT UPDATE DELETE MERGE

LCD: COMMIT, ROLLBACK, SAVEPOINT

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.

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:

```
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:

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

Sistemul realizează ROLLBACK implicit

- dacă se închide anormal (defecțiune hardware sau software, pană de curent etc.);

Nici o comandă LDD (CREATE, ALTER, DROP) nu poate fi anulată.

Exemplul 1: Analizați efectul următoarei secvențe de instrucțiuni:

```
CREATE TABLE dept_***
AS SELECT * FROM departmets;

SELECT *
FROM dept_***;

SAVEPOINT a;

DELETE FROM dept_***;

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

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

```
SAVEPOINT b;
```

```
INSERT INTO dept_***  
VALUES (400,'Juridic',150,3000);
```

```
SELECT COUNT(*)  
FROM dept_***;
```

```
ROLLBACK TO b;
```

```
SELECT COUNT(*)  
FROM dept_***;
```

```
ROLLBACK TO a;
```

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

```
COMMIT;
```

```
SELECT *  
FROM dept_***;
```

Exercițiul 2: Creați tabele *emp_**** și *dept_**** (dacă nu este deja creat), având aceeași structură și date ca și tabelele *employees*, respectiv *departments*.

```
CREATE TABLE emp_*** AS  
SELECT * FROM employees;
```

Să se selecteze toate înregistrările din cele două tabele create anterior.

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

```
DELETE FROM emp_***;  
DELETE FROM dept_***;  
COMMIT;
```

Exercițiul 3: Să se listeze structura tabelului *employees* și să se compare cu structura tabelului *emp_****. Ce observații?

Sintaxa simplificată a 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;
```

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.

Exemplul 4: Să se exemplifice câteva dintre erorile care pot să apară la inserare și să se observe mesajul returnat 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);
```

Exercițiul 5: Inerați în tabelul *emp_**** salariații (din tabelul *employees*) al căror comision depășește 25% din salariu.

Exemplul 6: Creați tabele *emp1_****, *emp2_**** și *emp3_**** cu aceeași structură ca tabelul *employees*. Inerați, utilizând o singură comandă INSERT, informații din tabelul *employees*:

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

Verificați rezultatele, apoi ștergeți toate înregistrările din aceste tabele.

Obs. 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.

```
CREATE TABLE emp1_***
AS SELECT * FROM employees WHERE 1=0;
...
INSERT ALL
    WHEN salary <= 6000 THEN
        INTO emp1_***
    WHEN salary >= 6000 AND salary <= 10000 THEN
        INTO emp2_***
    ELSE
        INTO emp3_***
SELECT * FROM employees;

DELETE FROM emp1_***;
...
COMMIT;
```

Exercițiul 7: Să se creeze tabelul *emp0_**** cu aceeași structură ca tabelul *employees*. Inerați, utilizând o singură comandă INSERT, informații din tabelul *employees*:

- în tabelul *emp0_**** salariații care lucrează în departamentul 80;
- în tabelul *emp1_**** salariații care au salariul mai mic decât 6000 (care nu se regăsesc în tabelul *emp0_****);
- în tabelul *emp2_**** salariații care au salariul cuprins între 6000 și 10000 (care nu se regăsesc în tabelele *emp0_**** și *emp1_****);
- în tabelul *emp3_**** salariații care au salariul mai mare decât 10000 (care nu se regăsesc în tabelele *emp0_****, *emp1_**** și *emp2_****).

Obs.

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

Exercițiul 8: Inerați o linie nouă în tabelul *dept_****, folosind valori introduse de la tastatură.

```
VALUES (&cod, . . . .)
```

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 <i>CHAR</i> . 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

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

Exemplul 9:

```

VARIABLE v_ume VARCHAR2 (20)

BEGIN

    SELECT last_name
    INTO      :v_ume
    FROM      employees
    WHERE     employee_id = 100;

END;

/

PRINT v_ume

```

Exercițiul 10: Inserați o linie nouă în tabelul *dept_****. Salvați într-o variabilă de legătură codul departamentului nou introdus. Afișați valoarea menținută în variabila respectivă. Anulați tranzacția.

```

VARIABLE v_cod NUMBER

INSERT INTO dept_*** (department_id,
    department_name,location_id)
VALUES (200, 'dept_nou', 2000)
RETURNING department_id INTO :v_cod;

PRINT v_cod

ROLLBACK;

```

Sintaxa simplificată a comenzii **DELETE**

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

Exemplul 11: Ștergeți toate înregistrările din tabelele *emp_**** și *dept_****. Inșerați în aceste tabele toate înregistrările corespunzătoare din *employees*, respectiv *departments*. Permanentizați tranzacția.

```
DELETE FROM dept_***;
```

```
DELETE FROM emp_***;
```

```
INSERT INTO emp_***  
SELECT * FROM employees;
```

```
INSERT INTO dept_***  
SELECT * FROM departments;
```

```
COMMIT;
```

Exercițiul 12: Ștergeți angajații care nu au comision. Anulați modificările.

Exercițiul 13: Eliminați departamentele care nu au nici un angajat. Anulați modificările.

Exercițiul 14: Eliminați angajații care nu aparțin unui departament valid. Anulați modificările.

Exercițiul 15: Ștergeți un angajat al cărui cod este dat de la tastatură.

Exercițiul 16: Să se șteargă angajatul având codul 100. Să se mențină numele acestuia într-o variabilă de legătură. Afișați valoarea acestei variabile.

Sintaxa simplificată a comenzii **UPDATE**:

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

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.

Exemplul 17: Măriți salariul tuturor angajaților din tabelul *emp_**** cu 5%. Anulați modificările.

```
UPDATE emp_***  
SET salary = salary * 1.05;  
ROLLBACK;
```

Exercițiul 18: Schimbați jobul tuturor salariaților din departamentul 80 care au comision în 'SA_REP'. Anulați modificările.

Exercițiul 19: Să se promoveze Douglas Grant la manager în departamentul 20, având o creștere de salariu cu 1000\$.

Exercițiul 20: Să se modifice jobul și departamentul angajatului având codul 114, astfel încât să fie la fel cu cele ale angajatului având codul 205.

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

Exercițiul 21: Pentru fiecare departament să se mărească salariul celor care au fost angajați primii astfel încât să devină media salariilor din companie.

Exercițiul 22: Să se modifice 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 modificările.

Exercițiul 23: Să se modifice cererea de la exercițiul anterior astfel încât actualizarea coloanei *email* să fie realizată doar pentru angajatul având codul 200. Să se mențină numele și emailul acestuia în două variabile de legătură. Să se anuleze tranzacția.

Exercițiul 24: Măriți cu 1000 salariul unui angajat al cărui cod este introdus de la tastatură.

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);
```

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.

Exemplul 25: Să se șteargă din tabelul *emp_**** toți angajații care nu lucrează în departamentul 20. Să se introducă sau să actualizeze datele din tabelul *emp_**** folosind tabelul *employees*.

```
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);
```