

# Baze de date - Anul 1

## Laborator 4

---

### Limbajul de manipulare a datelor (LMD)

### Limbajul de control al datelor (LCD)

#### I. [Obiective]

- Operații asupra conținutului tabelilor bazei de date (inserare, actualizare, ștergere)
- Conceptul de tranzacție
- Controlul tranzacțiilor (permanentizare, anulare)

#### II. [LMD și 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 condiționată de înregistrări (*MERGE*);
  - suprimarea de înregistrări (*DELETE*).
- **Tranzacția** este o unitate logică de lucru, constituită dintr-o secvență de comenzi care trebuie să se execute **atomic** (ca un întreg) pentru a menține **consistența** bazei de date.
- *Server-ul Oracle* asigură consistența datelor pe baza tranzacțiilor, inclusiv în eventualitatea unei anomalii a unui proces sau a sistemului. Tranzacțiile oferă mai multă flexibilitate și control în modificarea datelor.
- Comenzile SQL care alcătuiesc **LCD** sunt:
  - *ROLLBACK* – pentru a renunța la modificările aflate în așteptare se utilizează instrucțiunea *ROLLBACK*. În urma execuției acesteia, se încheie tranzacția, se anulează modificările asupra datelor, se restaurează starea lor precedentă și se eliberează blocările asupra liniilor.
  - *COMMIT* – determină încheierea tranzacției curente și permanentizarea modificărilor care au intervenit pe parcursul acesteia. Instrucțiunea suprimă toate punctele intermediare definite în tranzacție și eliberează blocările tranzacției.

**Observație:** O comandă LDD (*CREATE*, *ALTER*, *DROP*) determină un **COMMIT implicit**.

- *SAVEPOINT* – instrucțiunea *SAVEPOINT* marchează un punct intermediar în procesarea tranzacției. În acest mod este posibilă împărțirea tranzacției în subtranzacții. Această instrucțiune nu face parte din standardul *ANSI* al limbajului *SQL*.

### III. [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 și tip, celor precizate în clauza *INTO*. În absența unei liste de coloane în clauza *INTO*, subcererea trebuie să furnizeze valori pentru fiecare atribut al obiectului destinație, respectând ordinea în care acestea au fost definite.

#### Observații (tipuri de date):

- Pentru claritate, este recomandată utilizarea unei liste de coloane în clauza *INSERT*.
- În clauza *VALUES*, valorile de tip caracter și 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 funcții.
- Adăugarea unei linii care va conține 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 șirurilor de caractere sau al datelor calendaristice se poate preciza șirul vid ('').

#### Observații (erori):

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

- Nu a fost specificată o valoare pentru o coloană *NOT NULL*;

- Există valori duplicate care încalcă o constrângere de unicitate;
- A fost încălcată constrângerea de cheie externă sau o constrângere de tip *CHECK*;
- Există o incompatibilitate în privința tipurilor de date;
- S-a încercat inserarea unei valori având o dimensiune mai mare decât a coloanei corespunzătoare.

## 2. Inserări multi-tabel

O inserare multi-tabel presupune introducerea de linii calculate pe baza rezultatelor unei subcereri, într-unul sau mai multe tabele. Acest tip de inserare este util în mediul *data warehouse*.

În absența acestui tip de inserare, ar fi necesare  $n$  operații independente *INSERT INTO...SELECT...*, unde  $n$  reprezintă numărul tabelelor destinație. Aceasta presupune  $n$  procesări ale aceleiași surse de date și, prin urmare, creșterea de  $n$  ori a timpului necesar procesului.

Sintaxa comenzii *INSERT* în acest caz poate fi:

- Pentru inserări necondiționate:

```
INSERT ALL INTO... [INTO...]  
    subcerere;
```

- Pentru inserări condiționate:

```
INSERT [ALL | FIRST]  
WHEN condiție THEN INTO...  
[WHEN condiție THEN INTO...  
[ELSE INTO ...]]  
    subcerere;
```

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

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

## IV. [Exerciții – *INSERT*]

1. Să se creeze tabelele *EMP\_pnu*, *DEPT\_pnu* (în șirul de caractere "pnu",  $p$  reprezintă prima literă a prenumelui, iar  $nu$  reprezintă primele două litere ale numelui dumneavoastră), prin copierea structurii și conținutului tabelelor *EMPLOYEES*, respectiv *DEPARTMENTS*.

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

2. Listați structura tabelelor sursă și a celor create anterior. Ce se observă?
3. Listați conținutul tabelelor create anterior.
4. Pentru introducerea constrângerilor de integritate, executați instrucțiunile LDD indicate în continuare. Prezentarea detaliată a LDD se va face în cadrul laboratorului 5.

```
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);
```

**Observație:** Ce constrângere nu am implementat?

5. Să se insereze departamentul 300, cu numele *Programare* în *DEPT\_pnu*.

Analizați cazurile, precizând care este soluția corectă și explicând erorile celorlalte variante. Pentru a anula efectul instrucțiunii(ilor) corecte, utilizați 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);*

Executați varianta care a fost corectă de două ori. Ce se obține și 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). Determinați ca efectele instrucțiunii să devină permanente.

Atenție 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 explicită de inserare). Se presupune că data angajării acestuia este cea curentă (*SYSDATE*). Salvați înregistrarea.

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;
```

Introduceți un angajat precizând pentru valoarea *employee\_id* o subcerere care returnează (codul maxim +1).

9. Se poate utiliza clauza *WITH* pentru rezolvarea problemei anterioare?
10. Creați un nou tabel, numit *EMP1\_PNU*, care va avea aceeași structură ca și *EMPLOYEES*, dar nicio înregistrare. Copiați în tabelul *EMP1\_PNU* salariații (din tabelul *EMPLOYEES*) al căror comision depășește 25% din salariu.

```
CREATE TABLE emp1_pnu AS SELECT * FROM employees WHERE 1=0;  
--DELETE FROM emp1_pnu; --necesar daca nu aveam clauza WHERE de  
mai sus  
INSERT INTO emp1_pnu  
  SELECT *  
  FROM employees  
  WHERE commission_pct > 0.25;  
SELECT employee_id, last_name, salary, commission_pct  
FROM emp_pnu;  
ROLLBACK;
```

Ce va conține tabelul *EMP1\_PNU* în urma acestei succesiuni de comenzi?

11. Inserați o nouă înregistrare în tabelul *EMP\_PNU* care să totalizeze salariile, să calculeze media comisioanelor, iar câmpurile de tip dată să conțină data curentă și câmpurile de tip caracter să conțină textul 'TOTAL'. Numele și prenumele angajatului vor corespunde utilizatorului curent (*USER*). Pentru câmpul *employee\_id* se va introduce valoarea 0, iar *manager\_id* și *department\_id* vor avea valoarea null.
12. Introduceți încă 2 înregistrări în tabel (prin orice metodă).
13. Creați 2 tabele *emp2\_pnu* și *emp3\_pnu* cu aceeași structură ca tabelul *EMPLOYEES*, dar fără înregistrări (acceptăm omiterea constrângerilor de integritate). Prin intermediul unei singure comenzi, copiați din tabelul *EMPLOYEES*:
  - în tabelul *EMP1\_PNU* salariații care au salariul mai mic decât 5000;
  - în tabelul *EMP2\_PNU* salariații care au salariul cuprins între 5000 și 10000;
  - în tabelul *EMP3\_PNU* salariații care au salariul mai mare decât 10000.

Verificați rezultatele, apoi ștergeți toate înregistrările din aceste tabele.

*INSERT ALL*

*WHEN ... THEN*

*INTO ...*

*...*

*ELSE*

*INTO ...*

*SELECT \* FROM employees;*

*SELECT \* FROM emp1\_pnu;*

*SELECT \* FROM emp2\_pnu;*

*SELECT \* FROM emp3\_pnu;*

*DELETE FROM emp1\_pnu;*

*DELETE FROM emp2\_pnu;*

*DELETE FROM emp3\_pnu;*

14. Să se creeze tabelul *EMP0\_PNU* cu aceeași structură ca tabelul *EMPLOYEES* (fără constrângeri), dar fără nicio înregistrare. Copiați din tabelul *EMPLOYEES*:
  - în tabelul *EMP0\_PNU* salariații care lucrează în departamentul 80;
  - în tabelul *EMP1\_PNU* salariații care au salariul mai mic decât 5000;
  - în tabelul *EMP2\_PNU* salariații care au salariul cuprins între 5000 și 10000;
  - în tabelul *EMP3\_PNU* salariații care au salariul mai mare decât 10000.

Dacă un salariat se încadrează în tabelul *EMP0\_PNU* atunci acesta nu va mai fi inserat și în alt tabel (tabelul corespunzător salariului său).

*INSERT FIRST*

```
WHEN ... THEN
    INTO ...
...
ELSE
    INTO ...
SELECT * FROM employees;
SELECT * FROM emp*_pnu;
```

## V. [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];
```

### Observații:

- în general, pentru identificarea unei linii se folosește o condiție ce implică cheia primară;
- dacă nu apare clauza *WHERE* atunci sunt afectate toate liniile tabelului specificat;
- cazurile în care instrucțiunea *UPDATE* nu poate fi executată sunt similare celor în care eșuează instrucțiunea *INSERT*. Acestea au fost menționate anterior.

## VI. [Exerciții – *UPDATE*]

15. Măriți salariul tuturor angajaților din tabelul *EMP\_PNU* cu 5%. Vizualizați, iar apoi anulați modificările.
16. Schimbați jobul tuturor salariaților din departamentul 80 care au comision în 'SA\_REP'. Anulați modificările.
17. Să se promoveze Douglas Grant la funcția de manager în departamentul 20, având o creștere de salariu de 1000. Se poate realiza modificarea prin intermediul unei singure comenzi?
18. Schimbați salariul și comisionul celui mai prost plătit salariat din firmă, astfel încât să devină egale cu salariul și comisionul mediu. Țineți cont de liniile introduse anterior în calculul mediei (media se va calcula raportat la numărul total de angajați).

19. Să se modifice jobul și departamentul angajatului având codul 114, astfel încât valorile acestor atribute să devină egale cu cele ale angajatului având codul 205.

## VII. [Comanda *DELETE*]

Sintaxa simplificată a comenzii **DELETE** este:

*DELETE FROM nume\_tabel*

*[WHERE conditie];*

Dacă nu se specifică nicio condiție, atunci vor fi șterse toate liniile din tabel.

## VIII. [Exerciții – *DELETE*]

20. Ștergeți toate înregistrările din tabelul *DEPT\_PNU*. Ce înregistrări se pot șterge? Anulați modificările.
21. Ștergeți angajații care nu au comision. Anulați modificările.
22. Suprimați departamentele care nu au nici un angajat. Anulați modificările.

## IX. [Exerciții – *LMD, LCD*]

23. Să se șteargă un angajat din tabelul *EMP\_PNU*. Modificările vor deveni permanente.
24. Să se mai introducă o linie în tabel.
25. Să se marcheze un punct intermediar în procesarea tranzacției.
- SAVEPOINT p*
26. Să se șteargă tot conținutul tabelului. Listați conținutul tabelului.
27. Să se renunțe la cea mai recentă operație de ștergere, fără a renunța la operația precedentă de introducere.
- ROLLBACK TO p*
28. Listați conținutul tabelului. Determinați ca modificările să devină permanente.