

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 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.

Obs: 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.

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 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 unicatitate;
- 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. 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 operații independente *INSERT INTO...SELECT...*, unde n reprezintă numărul tabelor destinație. Aceasta presupunea 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.

Exerciții [I]

1. Să se creeze tabelele *EMP_pnu*, *DEPT_pnu* (în care șirul de caractere "pnu", *p* reprezintă prima literă a prenumelui, iar *nu* reprezintă primele două litere ale numelui), prin copierea structurii și conținutului tabelor *EMPLOYEES*, respectiv *DEPARTMENTS*.

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

2. Listați structura tabelor sursă și a celor create anterior. Ce se observă?
3. Listați conținutul tabelor create anterior.
4. Pentru introducerea **constrângerilor de integritate**, executați instrucțiunile 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*.
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.

```
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. **Creați** un nou tabel, numit *EMP1_PNU*, care va avea aceeași structură ca și *EMPLOYEES*, dar nici o înregistrare. **Copiați** în tabelul *EMP1_PNU* salariații (din tabelul *EMPLOYEES*) al căror comision depășește 25% din salariu.
9. Să se creeze un fișier (*script file*) care să permită introducerea de înregistrări în tabelul *EMP_PNU* în mod interactiv. Se vor cere utilizatorului: **codul, numele, prenumele și salariul angajatului**. Câmpul **email** se va completa automat prin concatenarea primei litere din prenume și a primelor 7 litere din nume.

Executați script-ul pentru a introduce 2 înregistrări în 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;
```

10. 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.

11. Să se creeze tabelul **EMP0_PNU** cu aceeași structură ca tabelul **EMPLOYEES** (fără constrângeri), dar fără nici o 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).

```
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];
```

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;
- cazurile în care instrucțiunea *UPDATE* nu poate fi executată sunt similare celor în care eșuează instrucțiunea *INSERT*. Acestea au fost menționate anterior.

Exerciții [II]

12. Măriți **salariul tuturor angajaților** din tabelul *EMP_PNU* cu 5%. Vizualizați, iar apoi anulați modificările.

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

SELECT * FROM emp_pnu;

ROLLBACK;
```

13. Schimbați **jobul tuturor salariaților** din departamentul 80 care au comision, în 'SA_REP'. Anulați modificările.

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

III. Comanda DELETE

Sintaxa simplificată a comenzii **DELETE** este:

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

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

Exercitii [III]

15. Ștergeți **toate înregistrările** din tabelul *DEPT_PNU*. Ce înregistrări se pot șterge? Anulați modificările.

16. Suprimați departamentele care nu au angajati. Anulați modificările.

Exerciții [LMD, LCD]

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

18. Să se marcheze un punct intermediar in procesarea tranzacției (*SAVEPOINT p*).

19. Să se șteargă din tabelul *DEPT_PNU* departamentele care au codul de departament cuprins între 160 si 200 . Listați conținutul tabelului.

20. Să se renunțe la cea mai recentă operație de ștergere, fără a renunța la operația precedentă de introducere. (*ROLLBACK TO p*). Determinați ca modificările să devină permanente.