Baze de date-Anul 2
Laborator 7 SQL

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 şi control în modificarea datelor.
- Comenzile SQL care alcătuiesc **LCD** sunt:
  - ROLLBACK – pentru a renunța la modificările aflate în aşteptare se utilizează instrucțiunea *ROLLBACK*. În urma execuției acesteia, se încheie tranzacția, se anulează modificările asupra datelor, se restaurează starea lor precedentă şi se eliberează blocările asupra liniilor.
  - COMMIT - determină încheierea tranzacției curente şi permanentizarea modificărilor care au intervenit pe parcursul acesteia. Instrucțiunea suprimă toate punctele intermediare definite în tranzacție şi 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 şi 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 şi 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 şirurilor de caractere sau al datelor calendaristice se poate preciza şirul vid ('').

Observații (erori):

Server-ul *Oracle* aplică automat toate tipurile de date, domeniile de valori şi 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 duplicat 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. Inserari 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, introdus de *Oracle9i*, este util în mediul *data warehouse*.

Pentru o astfel de inserare, în versiunile anterioare lui *Oracle9i* erau necesare *n* operații independente *INSERT INTO…SELECT…*, unde *n* reprezintă numărul tabelelor destinație. Aceasta presupunea *n* procesări ale aceleiaşi surse de date şi, prin urmare, creşterea 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 şirul de caractere “pnu”, *p* reprezintă prima literă a prenumelui, iar *nu* reprezintă primele două litere ale numelui dumneavoastră), prin copierea structurii şi 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 şi 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 4.

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 şi 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 şi 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.

INSERT INTO EMP_pnu
VALUES (250, ‘Prenume’, ‘Nume’, null, null, …, 300);
COMMIT;

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

```
INSERT INTO  EMP_pnu (employee_id, first_name, last_name, …, department_id)
    VALUES (251, 'Prenume', 'Nume', …, 300);
COMMIT;
```

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;
```

Încercați dacă este posibilă introducerea unui angajat, precizând pentru valoarea *employee_id* o subcerere care returnează (codul maxim +1).

9. Creați un nou tabel, numit *EMP1_PNU*, care va avea aceeaşi structură ca şi *EMPLOYEES*, dar nici o înregistrare. Copiați în tabelul *EMP1_PNU* salariații (din tabelul *EMPLOYEES*) al căror comision depăşeşte 25% din salariu.

```
CREATE TABLE emp1_pnu AS SELECT * FROM employees;
DELETE FROM emp1_pnu;
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 contine tabelul *EMP1_PNU* in urma acestei succesiuni de comenzi?

10. Inserați o nouă înregistrare în tabelul *EMP_PNU* care să totalizeze salariile, să facă media comisioanelor, iar câmpurile de tip dată să conțină data curentă şi câmpurile de tip caracter să conțină textul 'TOTAL'. Numele şi prenumele angajatului să corespundă utilizatorului curent (*USER*). Pentru câmpul *employee_id* se va introduce valoarea 0, iar pentru *manager_id* şi *department_id* se va da valoarea null.

```
INSERT INTO emp_pnu
      SELECT 0,USER,USER, 'TOTAL', 'TOTAL',SYSDATE,
           'TOTAL', SUM(salary), ROUND(AVG(commission_pct)), null, null
      FROM employees;
```

11. Să se creeze un fişier (*script file*) care să permită introducerea de înregistrări în tabelul *EMP_PNU* în mod interactiv. Se vor cere utilizatorului: codul, numele, prenumele si salariul angajatului. Câmpul *email* se va completa automat prin concatenarea primei litere din prenume şi a primelor 7 litere din nume.

```
REM setari
REM comenzi ACCEPT
INSERT INTO emp_pnu
   VALUES (&…);
REM suprimarea variabilelor utilizate
REM anularea setarilor, prin stabilirea acestora la valorile implicite
```

Executati script-ul pentru a introduce 2 inregistrari in tabel.

12. Creați 2 tabele *emp2_pnu* şi *emp3_pnu* cu aceeaşi 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 şi 10000;
- în tabelul *EMP3_PNU* salariații care au salariul mai mare decât 10000.

Verificați rezultatele, apoi ştergeți toate înregistrările din aceste tabele.

```
INSERT ALL
     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 emp1_pnu;
SELECT * FROM emp2_pnu;
SELECT * FROM emp3_pnu;

  DELETE FROM emp1_pnu;
  DELETE FROM emp2_pnu;
  DELETE FROM emp3_pnu;
```

13. Să se creeze tabelul *EMP0_PNU* cu aceeaşi 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 şi 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 şi în alt tabel (tabelul corespunzător salariului său).

```
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 emp*_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 foloseşte 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 eşuează instrucțiunea *INSERT*. Acestea au fost menționate anterior.

Exerciții [II]

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

```
UPDATE emp_pnu
SET salary = salary * 1.05;
SELECT * FROM emp_pnu;
ROLLBACK;
```

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

16. Să se promoveze Douglas Grant la manager în departamentul 20, având o creştere de salariu cu 1000$. Se poate realiza modificarea prin intermediul unei singure comenzi?

17. Schimbați salariul şi comisionul celui mai prost plătit salariat din firmă, astfel încât să fie egale cu salariul si comisionul şefului său.

18. Să se modifice adresa de e-mail pentru angajații care câştigă cel mai mult în departamentul în care lucrează astfel încât acesta să devină inițiala numelui concatenată cu prenumele. Dacă nu are prenume atunci în loc de acesta apare caracterul '.'. Anulați modificările.

19. Pentru fiecare departament să se mărească salariul celor care au fost angajați primii astfel încât să devină media salariilor din companie. Țineți cont de liniile introduse anterior.

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

21. Creați un script prin intermediul caruia sa fie posibilă actualizarea în mod interactiv de înregistrări ale tabelului *dept_pnu*. Se va cere codul departamentului care urmează a fi actualizat, se va afişa linia respectivă, iar apoi se vor cere valori pentru celelalte câmpuri.

III. Comanda DELETE

Sintaxa simplificată a comenzii ***DELETE*** este:

DELETE FROM nume_tabel
[WHERE conditie];

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

Exercitii [III]

22. Ştergeți toate înregistrările din tabelul *DEPT_PNU*. Ce înregistrări se pot şterge? Anulați modificările.
23. Ştergeți angajații care nu au comision. Anulați modificările.
24. Suprimați departamentele care un au nici un angajat. Anulați modificările.
25. Să se creeze un fişier *script* prin care se cere utilizatorului un cod de angajat din tabelul *EMP_PNU*. Se va lista inregistrarea corespunzatoare acestuia, iar apoi linia va fi suprimată din tabel.

Exerciții [LMD, LCD]

26. Să se şteargă un angajat din tabelul *EMP_PNU* prin intermediul *script*-ului creat la problema 25. Modificările să devină permanente.
27. Să se mai introducă o linie in tabel, rulând încă o dată fişierul creat la exercițiul 11.
28. Să se marcheze un punct intermediar in procesarea tranzacției.

SAVEPOINT p

29. Să se şteargă tot conținutul tabelului. Listați conținutul tabelului.
30. Să se renunțe la cea mai recentă operație de ştergere, fără a renunța la operația precedentă de introducere.

ROLLBACK TO p

31. Listați conținutul tabelului. Determinați ca modificările să devină permanente.

IV. Comanda MERGE

Instrucțiunea *MERGE* permite inserarea sau actualizarea condiționată a datelor dintr-un tabel al bazei de date. Sintaxa ei simplificată este următoarea:

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*);

Instrucțiunea efectuează:

- *UPDATE* dacă înregistrarea există deja în tabel
- *INSERT* dacă înregistrarea este nouă.

În acest fel, se pot evita instrucțiunile *UPDATE* multiple.

Exerciții [IV]

32. Să se şteargă din tabelul *EMP_PNU* toți angajații care câştigă comision. Să se introducă sau să se actualizeze datele din tabelul *EMP_PNU* pe baza tabelului *employees*.

```
MERGE INTO emp_pnu  x
    USING employees e
    ON (x.employee_id = e.employee_id)
    WHEN MATCHED THEN
UPDATE SET
   x.first_name=e. first_name,
   x.last_name=e.last_name,
   x.email=e.email,
   x.phone_number=e.phone_number,
   x.hire_date= e.hire_date,
   x.job_id= e.job_id,
   x.salary = e.salary,
   x.commission_pct= e.commission_pct,
   x.manager_id= e.manager_id,
   x.department_id= e.department_id
WHEN NOT MATCHED THEN
  INSERT VALUES (e.employee_id, e.first_name, e.last_name, e.email,
      e.phone_number, e.hire_date, e.job_id, e.salary, e.commission_pct, e.manager_id,
      e.department_id);
```