Instrucţiuni pentru manipularea datelor

Scopul acestui referat este de a prezenta instrucţiunile SQL caracteristice limbajului de manipulare a datelor (LMD), instrucţiuni ce sunt utilizate pentru a opera modificări asupra instanţei bazei de date. Aceste instrucţiuni permit inserarea într-un tabel, precum şi actualizarea şi ştergerea de înregistrări într-un tabel. De asemenea, va fi prezentat şi modul în care pot fi controlate tranzacţiile cu ajutorul comenzilor `COMMIT`, `SAVEPOINT` şi `ROLLBACK`.

Prin parcurgerea acestui referat studentul va dobândi cunoştinţele necesare pentru:

- descrierea fiecărei comenzi LMD;
- inserarea unei înregistrări într-un tabel;
- actualizarea înregistrărilor dintr-un tabel;
- ştergerea unei înregistrări dintr-un tabel;
- controlul tranzacţiilor.

1. Limbajul de manipulare a datelor

Limbajul de manipulare a datelor (LMD) reprezintă o componentă importantă a SQL. O comandă din LMD are ca obiectiv adăugarea, modificarea sau ştergerea datelor dintr-o bază de date. O colecţie de instrucţiuni LMD care formează o unitate logică de lucru se numeşte *tranzacţie*.

Să considerăm o bază de date din domeniul bancar. Atunci când un client al băncii doreşte să transfere bani dintr-un depozit într-un cont curent, tranzacţia ar putea consta în 3 operaţii distincte: se scade suma din depozit, se măreşte suma din contul curent, înregistrează tranzacţia în jurnalul de tranzacţii. Serverul Oracle trebuie să garanteze că toate cele 3 instrucţiuni SQL sunt executate în aşa fel încât să menţină corect balanţa celor două conturi. Dacă o instrucţiune dintr-o tranzacţie este împedicată să se execute, celelalte instrucţiuni ale tranzacţiei în cauză trebuie anulate.

2. Adăugarea unei înregistrări într-un tabel

Pentru a introduce înregistrări noi într-un tabel se va utiliza instrucţiunea `INSERT`, al cărei format este:

```
INSERT INTO  tabel [(coloana [,coloana…])]
VALUES  (valoare [,valoare…]);
```

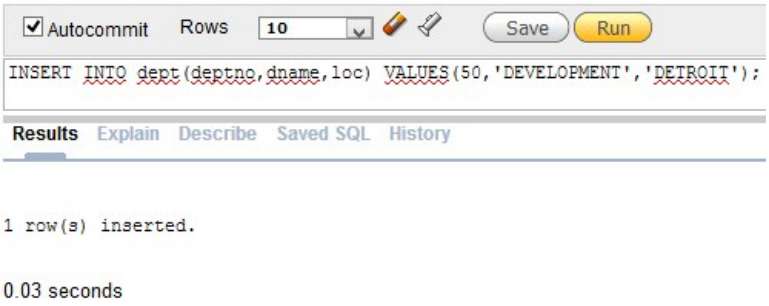
unde:

tabel	este numele tabelului
coloana	este numele coloanei în care se vor introduce valori.
valoare	este valoarea corespunzătoare coloanei.

Notă : Instrucţiunea `INSERT` împreună cu clauza `VALUES` adaugă numai un singur rând la un tabel.

Exemplu:

```
SQL> INSERT INTO dept(deptno, dname, loc) VALUES (50,
'DEVELOPMENT','DETROIT');
```



În cazul inserării unei noi înregistrări care conţine valori pentru toate coloanele, lista coloanelor din instrucţiunea `INSERT` nu mai este necesară. Dacă această listă nu este precizată, valorile trebuie enumerate conform ordinii coloanelor din tabel.

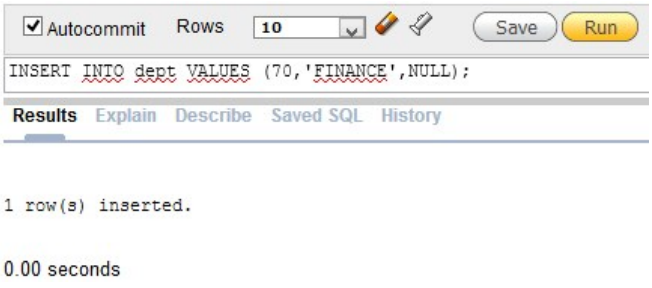
Pentru a insera înregistrări ce conţin valori null se poate utiliza una din următoarele metode:

- *metoda implicită* constă în omiterea din lista de coloane a coloanei corespunzătoare valorii null.

```
SQL> INSERT INTO dept(deptno, dname) VALUES(60, 'MIS');
```

- *metoda explicită* constă în specificarea fie a valorii NULL, fie a şirului vid (‘‘) în lista `VALUES`. Şirul vid va fi utilizat doar pentru date de tip dată calendaristică sau şir de caractere.

```
SQL> INSERT INTO dept VALUES (70,'FINANCE',NULL);
```



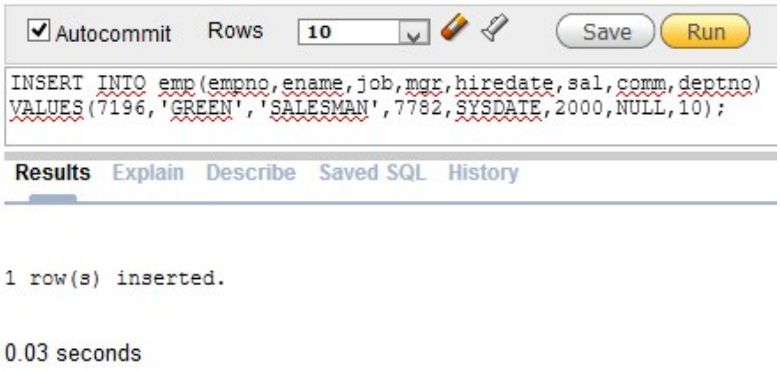
Înainte de introducerea unei valori null într-un tabel trebuie verificat dacă valoarea null este permisă în coloana corespunzătoare, inspectând câmpul `Null?` furnizat de comanda `DESCRIBE`.

2.1 Inserarea de valori speciale folosind funcţii SQL

Pentru a introduce valori speciale în tabele se pot utiliza *pseudocoloane*. Exemplul următor înregistrează informaţia pentru angajatul Green în tabelul `emp`. Pentru a introduce

data şi ora curentă în câmpul HIREDATE este folosită funcţia SYSDATE. O altă funcţie ce poate fi utilizată este funcţia USER, care furnizează numele utilizatorului curent.

```
SQL>INSERT INTO emp(empno, ename, job, mgr,hiredate, sal,
comm, deptno)VALUES(7196,'GREEN','SALESMAN',7782, SYSDATE,
2000, NULL, 10);
```

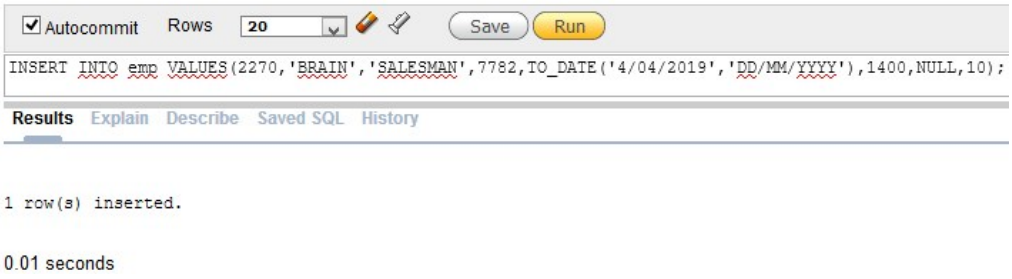


2.2 Inserarea valorilor de tip dată calendaristică

Formatul folosit de obicei pentru inserarea unei valori de tip dată calendaristică este DD-MON-YY. Dacă o dată calendaristică necesită specificarea altui secol sau a altei ore, se va apela funcţia TO_DATE.

Exemplul de mai jos înregistrează informaţia despre angajatul Brian în tabelul emp. Câmpul HIREDATE primeşte valoarea February 3, 1997. Dacă formatul RR este setat, secolul poate fi diferit de cel curent.

```
SQL>INSERT INTO emp VALUES (2269,'BRIAN','SALESMAN',7782,
TO_DATE('4/04/2019','DD/MM/YYYY'),1400, NULL, 10);
```



Formatul funcţiei TO_DATE este:

```
TO_DATE(date_char, fmt)
```

unde:

date_char	este data calendaristică sub forma unui şir de caractere
fmt	indică formatul argumentului date_char.

2.3 Inserarea de valori folosind variabile de substituţie

Este permisă scrierea unei instrucţiuni `INSERT` care să permită utilizatorului să adauge valori interactiv, folosind variabilele de substituţie SQL*Plus.

Exemplul următor înregistrează informaţiile pentru un departament în tabelul `dept`. Numărul departamentului, numele şi locaţia sunt introduse în mod interactiv de către utilizator. Pentru valori de tip dată calendaristică sau şir de caractere, ampersandul (&) şi numele variabilei sunt încadrate de apostrofuri (‘ ’).

```
SQL>INSERT INTO dept(deptno, dname, loc)VALUES
(&department_id, '&department_name','&location');
```

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

2.4 Crearea unui script pentru manipularea datelor

Instrucţiunea SQL împreună cu variabilele de substituţie pot fi salvate într-un fişier şi ori de câte ori se va executa fişierul script se va cere introducerea unor valori noi pentru variabile. Mesajele afişate la cererea introducerii valorilor pot fi modificate prin intermediul comenzii SQL*Plus `ACCEPT`.

```
ACCEPT  department_id PROMPT 'Introduceti -
        numarul departamentului:'
ACCEPT  department_name PROMPT 'Introduceti -
        numele departamentului:'
ACCEPT  location PROMPT 'Introduceti orasul:'

INSERT INTO  dept(deptno, dname, loc)
VALUES  (&department_id,'&department_name','&location');
```

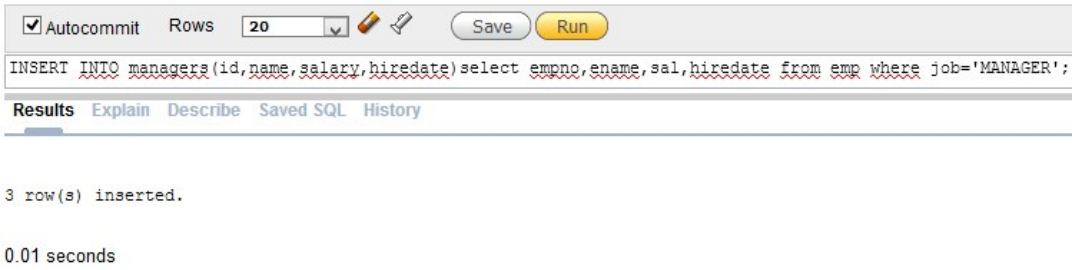
```
Introduceti numarul departamentului: 90
Introduceti numele departamentului: PAYROLL
Introduceti orasul: HOUSTON

1 row created.
```

Variabila de substitutie SQL*Plus nu trebuie precedată de & când este referită într-o comanda `ACCEPT`. Pentru a continua o comanda SQL*PLUS pe linia următoare se foloseşte caracterul ‘-‘.

2.5 Copierea înregistrărilor dintr-un alt tabel

Instrucţiunea `INSERT` poate fi utilizată pentru adăugarea unor înregistrări într-un tabel, valorile atributelor fiind derivate dintr-un tabel existent. Pentru aceasta se foloseşte în locul clauzei `VALUES` o **subinterogare**.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)SELECT
empno, ename, sal, hiredate FROM emp WHERE job = 'MANAGER';
```



Sintaxa este următoarea:

```
INSERT INTO  tabel [ coloana (, coloana) ]
             subinterogare;
```

unde: *tabel* este numele tabelului;
coloana este numele coloanei din tabelul în care se face inserarea.
subinterogare este subinterogarea care returnează înregistrări în tabel.

Notă: Numărul şi tipul câmpurilor (coloanelor) din lista specificată în instrucţiunea `INSERT` trebuie să corespundă numărului şi tipului valorilor din subinterogare.

3. Modificarea datelor dintr-un tabel

Înregistrările existente într-un tabel pot fi modificate cu ajutorul instrucţiunii `UPDATE`.

```
UPDATE tabel
SET    coloana = valoare [, coloana=valoare]
[WHERE conditie];
```

unde:

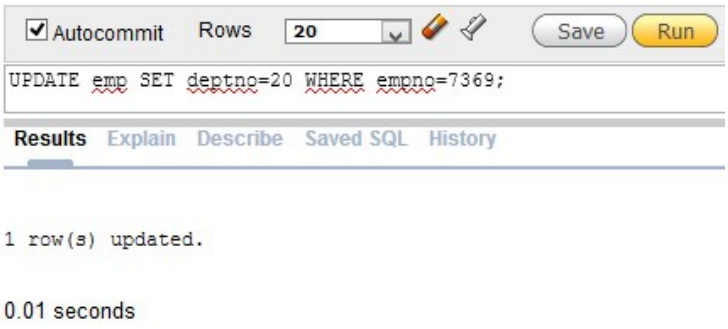
tabel este numele tabelului;
coloana este numele coloanei din tabelul în care se face modificarea;
valoare este **valoarea** sau **subinterogarea** corespunzătoare coloanei;
conditie identifică înregistrările care trebuie modificate, fiind alcătuită din expresii, nume de coloane, constante, subinterogări şi operatori de comparaţie.

Confirmarea unei operaţii de modificare se obţine prin interogarea tabelului, afişând tuplurile modificate.

Notă: Este indicată utilizarea cheii primare pentru a identifica o singură înregistrare. Folosirea altor coloane poate conduce la modificarea mai multor înregistrări. De exemplu, identificarea unei singure înregistrări din tabelul `emp` prin atributul `ename` poate fi periculoasă, deoarece pot exista mai mulţi angajaţi cu acelaşi nume.

Comanda `UPDATE` modifică numai anumite înregistrări dacă este specificată clauza `WHERE`. Exemplul următor transferă angajatul cu numărul 7782 (Clark) la departamentul 20.

```
SQL>UPDATE emp SET deptno = 20 WHERE  empno = 7369;
```



Notă: Dacă se omite clauza `WHERE` vor fi modificate toate înregistrările din tabel.

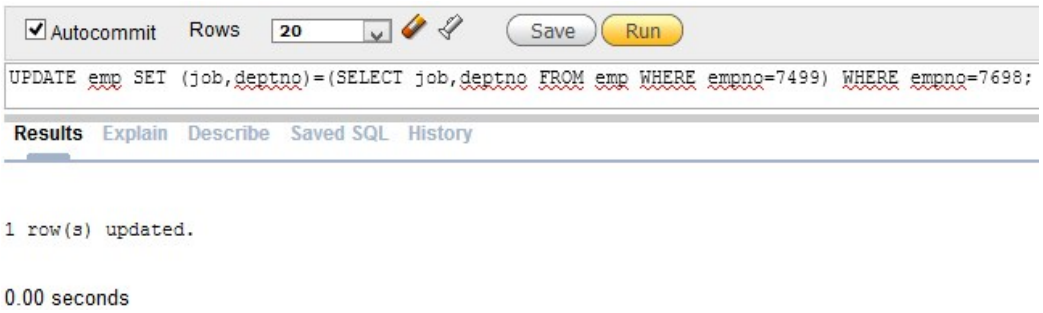
3.1 Modificarea înregistrărilor folosind subinterogări după mai multe coloane

Pentru a modifica simultan mai multe valori ale unui tuplu se va utiliza o subinterogare în clauza `SET` a instrucţiunii `UPDATE`. Formatul unei astfel de instrucţiuni `UPDATE` este:

```
UPDATE  tabel
SET (coloana, coloana, …) =
             (SELECT coloana, coloana, …
             FROM tabel
             WHERE conditie)
WHERE conditie;
```

Următoarea instrucţiune `UPDATE` modifică departamentul şi funcţia angajatului cu numărul 7698 cu valorile corespunzătoare angajatului având numărul 7499.

```
SQL> UPDATE emp SET (job, deptno) = (SELECT job, deptno FROM
emp WHERE empno = 7499) WHERE empno = 7698;
```



Pentru a efectua modificări într-un tabel pe baza valorilor din alt tabel este necesară utilizarea subinterogărilor în instrucţiunile `UPDATE`. Exemplul de mai jos actualizează tabelul `employee` pe baza valorilor din tabelul `emp`. Este schimbat numărul de departament pentru toţi angajaţii cu aceeaşi funcţie ca cea a angajatului 7788, noul lor număr de departament devenind egal cu al acestuia.

```
SQL> UPDATE employee SET deptno = (SELECT deptno FROM emp
WHERE empno = 7788)WHERE job = (SELECT job FROM emp WHERE
empno = 7788);
```

4. Ştergerea înregistrărilor dintr-un tabel

Pentru ştergerea uneia sau a mai multor înregistrări dintr-un tabel se va utiliza instrucţiunea `DELETE`, al cărei format este.

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

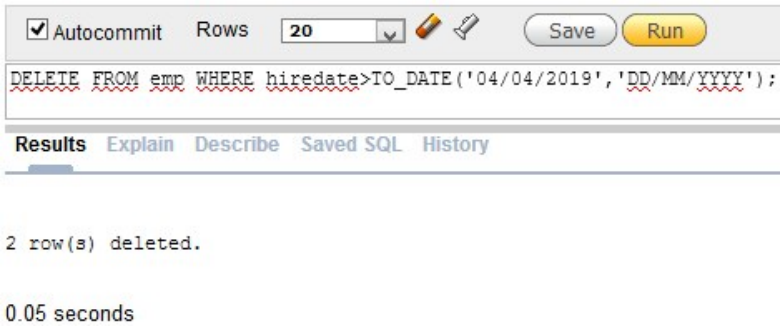
unde:

tabel	este numele tabelului;
conditie	identifică înregistrările care trebuie şterse şi este alcatuită din expresii, nume de coloane, constante, subinterogări şi operatori de comparaţie.

Condiţia din clauza `WHERE` identifică înregistrările ce vor fi şterse**. Dacă se omite clauza `WHERE` se vor şterge toate înregistrările din tabel.**

De exemplu, pentru ştergerea persoanelor angajate după 1 Ianuarie 1997 se va executa următoarea instrucţiune DELETE:

```
SQL> DELETE FROM    emp WHERE hiredate > TO_DATE ('01.01.97',
'DD.MM.YY');
```



4.1 Ştergerea înregistrărilor folosind valori dintr-un alt tabel

Ca şi în cazul instrucţiunii `UPDATE` se pot folosi subinterogări pentru a şterge înregistrări dintr-un tabel, folosind informaţiile din alt tabel. Exemplul următor şterge toţi angajaţii care lucrează în departamentul cu numele `SALES`. Subinterogarea caută în tabelul `dept` numărul departamentului `SALES`, apoi furnizează acest număr interogării principale care şterge înregistrările corespunzătoare din `emp`.

```
SQL> DELETE FROM emp WHERE deptno = (SELECT deptno FROM dept
 WHERE dname = 'SALES');
```

4.2 Încălcarea constrângerii de integritate

Încercarea de a şterge o înregistrare care conţine un câmp legat de o constrângere de integritate va fi sancţionată cu o eroare.

În exemplul de mai jos se încearcă ştergerea departamentului cu numărul 10 din tabelul `dept`, dar această încercarea ar provoca o eroare dacă numărul de departament ar fi o cheie externă pentru tabelul `emp`. Dacă înregistrarea părinte pe care dorim să o ştergem are înregistrări fii, atunci se va genera mesajul de eroare `child record found` cu numărul ORA - 02292.

```
SQL> DELETE FROM dept WHERE deptno = 10;
```

Problema impunerii unor constrângeri de integritate va fi tratată pe larg într-un referat ulterior.

Probleme:

Inserarea de date în tabelul `MY_EMPLOYEE`.

1. Descrieţi structura tabelului `MY_EMPLOYEE` pentru a identifica numele câmpurilor.

2. Adăugaţi prima înregistrare din tabelul de mai jos.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	795
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audry	aropebur	1550

3. Verificaţi adăugarea înregistrării în tabel.

4. Inseraţi următoarele trei înregistrări din tabelul de mai sus în tabelul `MY_EMPLOYEE`.

5. Modificaţi numele de familie al angajatului având ID = 3 în Drexler.

6. Modificaţi salariul la 1000 pentru toţi cei cu salariul < 900.

7. Ştergeţi înregistrarea corespunzătoare lui Betty Dancs din tabelul `MY_EMPLOYEE`.

8. Salvaţi toate modificările temporare.