

Universitatea Constantin Brâncuși din Târgu-Jiu  
Automatică și Informatică Aplicată

# Baze de date

Limbajul SQL

**THE INFORMATION COMPANY**

# *Curs 8*

## *Limbajul SQL*

# *Limbajul SQL*

## Capitolul 7

### 1. **Limbajul de manipulare al datelor (LMD)**

1.1. Adăugare o nouă înregistrare

1.2. Actualizarea datelor dintr-o tabelă

1.3. Ștergerea tuplurilor dintr-o tabelă

1.4. Instrucțiunea Merge

### 2. **Limbajul de control al datelor (LCD). Tranzacții**

# 7.1. Limbajul de manipulare al datelor (LMD)

Limbajul de manipulare al datelor este nucleul limbajului **SQL**.

Când doriți să adăugați, să actualizați, sau să ștergeți date din baza de date, executați comenzi **DML(Data Manipulation Language)**.

*O colecție de comenzi **DML** care formează o unitate logică reprezintă o tranzacție.*

# 7.1. Limbajul de manipulare al datelor (LMD)

În acest curs ne ocupăm de limbajul de manipulare al datelor (**DML**) care ne permite:

1. să adăugăm
2. să modificăm
3. sau să distrugem datele din baza de date

**Oracle 12c** conține următoarele funcții:

1. **INSERT**
2. **UPDATE**
3. **DELETE**
4. **MERGE**

# *Limbajul SQL*

## Capitolul 7

### 1. Limbajul de manipulare al datelor (LMD)

#### 1.1. Adăugarea unei noi înregistrari

#### 1.2. Actualizarea datelor dintr-o tabelă

#### 1.3. Ștergerea tuplurilor dintr-o tabelă

#### 1.4. Instrucțiunea Merge

### 2. Limbajul de control al datelor (LCD). Tranzacții

## 7.1.1. Adăugare o nouă înregistrare

Sintaxa este:

```
INSERT INTO tabela [ ( coloana [, coloana . . . ] ) ]  
VALUES ( valoare [, valoare . . . ] );
```

**tabela** – numele tabelului

**coloana** – numele coloanei din tabel

**valoare** – valoarea corespunzătoare coloanei

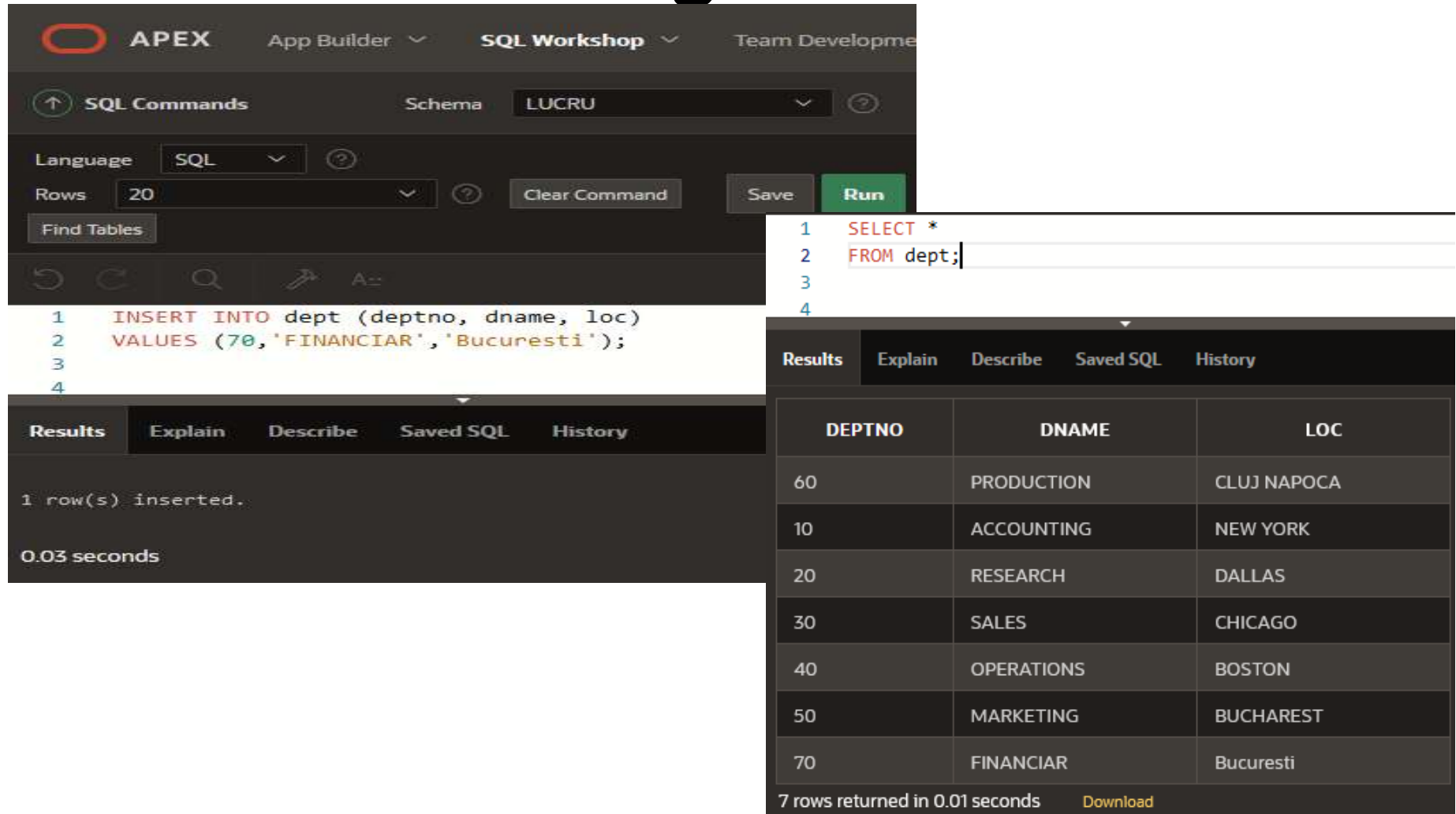
**Notă:** Se poate adăuga o singură linie o dată.

# 7.1.1. Adăugare o nouă înregistrare

Exemplu - Să se introducă un nou oras în tabela DEPT:

```
INSERT INTO dept (deptno, dname, loc)  
VALUES (70,'FINANCIAR','Bucuresti');
```

# 7.1.1. Adăugare o nouă înregistrare



The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', and 'Team Development'. The 'SQL Commands' section shows the 'Schema' set to 'LUCRU' and the 'Language' set to 'SQL'. The 'Rows' limit is set to 20. The main editor contains the following SQL command:

```
1 INSERT INTO dept (deptno, dname, loc)
2 VALUES (70, 'FINANCIAR', 'Bucuresti');
3
4
```

The 'Run' button is highlighted in green. Below the editor, the 'Results' tab is active, showing the following table:

DEPTNO	DNAME	LOC
60	PRODUCTION	CLUJ NAPOCA
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MARKETING	BUCHAREST
70	FINANCIAR	Bucuresti

At the bottom of the results pane, it indicates '7 rows returned in 0.01 seconds' and provides a 'Download' link. The status bar at the bottom of the interface shows '1 row(s) inserted.' and '0.03 seconds'.

## 7.1.1. Adăugare o nouă înregistrare

- Deoarece se poate insera o nouă linie ce conține valori pentru fiecare coloană, lista coloanelor nu mai este necesară în clauza **INSERT**.
- Totuși dacă nu utilizăm lista de coloane, valorile trebuie să fie listate în ordinea coloanelor din tabelă, iar o valoare trebuie utilizată pentru fiecare coloană.
- Pentru o utilizare mai ușoară putem folosi comanda **DESCRIBE dept**, care ne afișează câmpurile tabelii în ordinea lor, precum și tipul fiecărui câmp.

# 7.1.1. Adăugare o nouă înregistrare

## Inserarea liniilor ce conțin valori NULL

Inserarea liniilor ce conțin valori **NULL** se poate face prin două metode:

1. **metoda implicită**: Omiterea unui câmp din lista câmpurilor existente în tabela respectivă.

```
INSERT INTO dept (deptno, loc)  
VALUES (70, 'Bucuresti');
```

```
1  INSERT INTO dept (deptno, loc)  
2  VALUES (70, 'Bucuresti');
```

Câmpul DNAME a fost omis din lista

# 7.1.1. Adăugare o nouă înregistrare

2. **metoda explicită**: Specificarea cuvântului NULL în clauza **VALUES**.

```
INSERT INTO dept (deptno, dname, loc)  
VALUES (70, NULL, 'Bucuresti');
```

Campul DNAME este in lista, dar valoarea este NULL

```
1  INSERT INTO dept (deptno, dname, loc)  
2  VALUES (70, NULL, 'Bucuresti');
```

# 7.1.1. Adăugare o nouă înregistrare

## Inserarea unor valori speciale

Funcția **SYSDATE** înregistrează *data curentă și ora*.

Putem utiliza diferite funcții pentru a insera valori speciale în tabela noastră.

# 7.1.1. Adăugare o nouă înregistrare

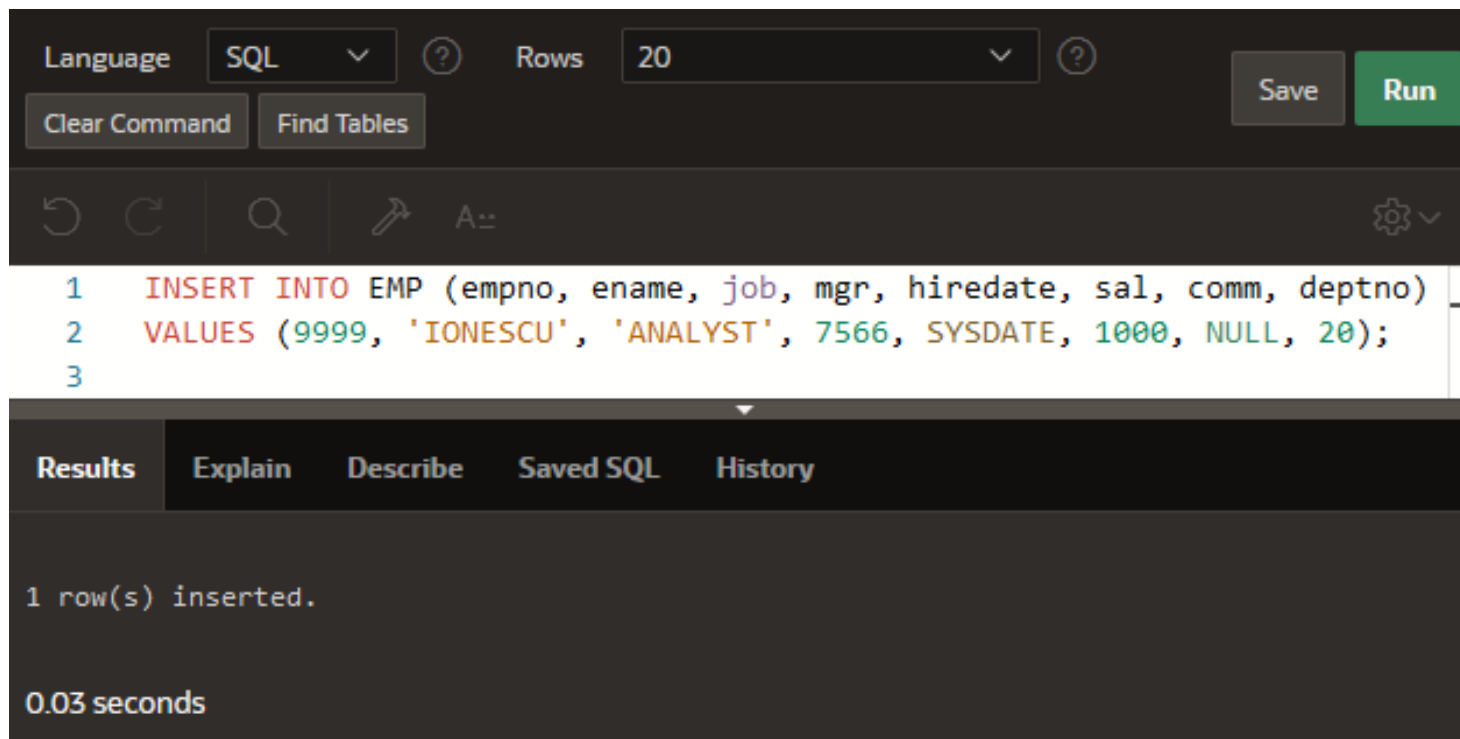
## Exemplu

Inserează în tabela Angajati datele personale, precum și data când acestea au fost introduse, prin utilizarea comenzii **SYSDATE**, care reprezintă data sistemului.

# 7.1.1. Adăugare o nouă înregistrare

**INSERT INTO** EMP (empno, ename, job, mgr, hiredate, sal, comm, deptno)

**VALUES** (9999, 'IONESCU', 'ANALYST', 7566, SYSDATE, 1000, NULL, 20);



The screenshot shows a SQL IDE interface. At the top, there are controls for 'Language' (set to SQL), 'Rows' (set to 20), and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. Below these is a toolbar with icons for undo, redo, search, and a settings gear. The main area contains the following SQL code:

```
1  INSERT INTO EMP (empno, ename, job, mgr, hiredate, sal, comm, deptno)
2  VALUES (9999, 'IONESCU', 'ANALYST', 7566, SYSDATE, 1000, NULL, 20);
3
```

Below the code editor is a 'Results' tab with sub-tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' sub-tab is active and displays the output: '1 row(s) inserted.' and '0.03 seconds'.

# 7.1.1. Adăugare o nouă înregistrare

## Copierea informațiilor dintr-o altă tabelă

- Se scrie comanda **INSERT** cu ajutorul unui **subquery**.
- **Nu se utilizează clauza VALUES.**
- Potriviți numărul de câmpuri din clauza **INSERT** cu cel din subquery.
- Se poate folosi clauza **INSERT** pentru a adăuga linii într-o tabelă unde valorile sunt dintr-o altă tabelă.
- În loc de clauza **VALUES**, folosim un subquery.

# 7.1.1. Adăugare o nouă înregistrare

## Sintaxa

```
INSERT INTO tabela [ coloana (, coloana) ]  
subquery(subcerere);
```

***tabela*** – numele tablei

***coloana*** – numele câmpului din tabelă

***subquery (subinterogare)*** – subquery-ul care returnează câmpurile din cealaltă tabelă

## 7.1.1. Adăugare o nouă înregistrare

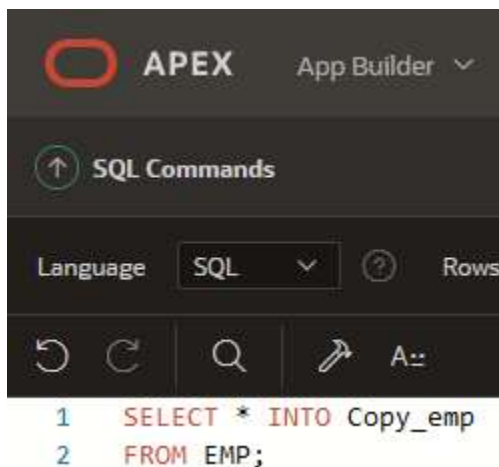
- Numărul de coloane și tipurile de date din lista câmpurilor din clauza **INSERT** trebuie să se potrivească cu valorile și tipurile de date din subquery.
- Pentru a crea o copie a liniilor unei tabele, vom folosi **SELECT \*** în subquery.

# 7.1.1. Adăugare o nouă înregistrare

## Exemplu

Se introduc datele din tabela EMP într-o altă tabelă numită Copy\_emp.

```
INSERT INTO copy_emp  
SELECT * FROM emp;
```



```
1 SELECT * INTO Copy_emp  
2 FROM EMP;
```

Subinterogarea selectează toate câmpurile tablei de unde se preiau date

# *Limbajul SQL*

## Capitolul 7

### 1. Limbajul de manipulare al datelor (LMD)

#### 1.1. Adăugare o nouă înregistrare

#### 1.2. Actualizarea datelor dintr-o tabelă

#### 1.3. Ștergerea tuplurilor dintr-o tabelă

#### 1.4. Instrucțiunea Merge

### 2. Limbajul de control al datelor (LCD). Tranzacții

## 7.1.2. Actualizarea datelor dintr-o tabelă

Schimbarea liniilor existente folosind clauza **UPDATE**.

### Sintaxa

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

## 7.1.2. Actualizarea datelor dintr-o tabelă

În sintaxă:

- ***tabela*** - numele tabelului
- ***coloana*** - numele coloanei în care vor fi introduse datele
- ***valoarea*** - valoarea corespunzătoare din subquery (subinterogare)
- ***condiție*** - identificarea câmpurilor care vor fi actualizate

## 7.1.2. Actualizarea datelor dintr-o tabelă

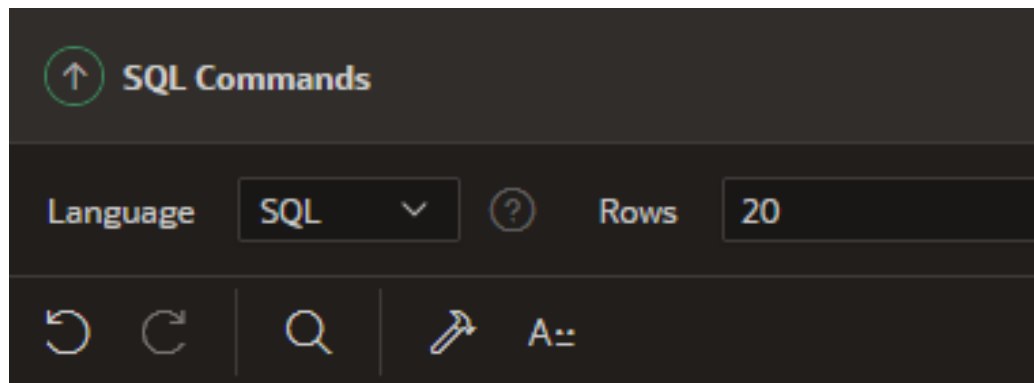
### Notă:

- În general se folosește cheia primară pentru a identifica o linie.
- Utilizarea altei coloane poate duce la actualizarea mai multor linii.
- De exemplu într-o relație numita *persoane* putem avea de două sau mai multe persoane cu același nume.

## 7.1.2. Actualizarea datelor dintr-o tabelă

### Exemplu 1

```
UPDATE EMP  
SET job = 'SALESMAN', hiredate = SYSDATE  
WHERE ename = 'IONESCU';
```



The screenshot shows a SQL Commands window with a dark background. At the top, it says "SQL Commands" with an upward arrow icon. Below that, there are controls for "Language" (set to "SQL") and "Rows" (set to "20"). At the bottom, there are icons for undo, redo, search, and a keyboard shortcut "A::".

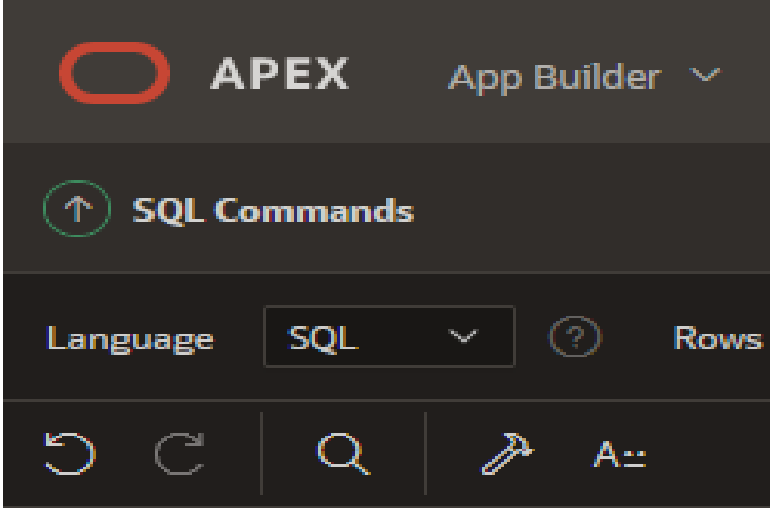
```
1 UPDATE EMP  
2 SET job = 'SALESMAN', hiredate = SYSDATE  
3 WHERE ename = 'IONESCU';
```

S-au modificat câmpurile  
job și hiredate pentru  
angajatul cu numele  
specificat

## 7.1.2. Actualizarea datelor dintr-o tabelă

### Exemplu 2

```
UPDATE EMP  
SET deptno = 99  
WHERE empno = 7499;
```



The screenshot shows the APEX SQL Commands interface. At the top, there is a red oval icon and the text 'APEX' and 'App Builder'. Below that, there is a green circle with an upward arrow and the text 'SQL Commands'. The 'Language' dropdown is set to 'SQL'. Below the interface, the SQL command is displayed in a list format:

```
1 UPDATE EMP  
2 SET deptno = 99  
3 WHERE empno = 7499;
```

S-a modificat deptno  
pentru angajatul cu  
empno specificat

# *Limbajul SQL*

## Capitolul 7

### 1. Limbajul de manipulare al datelor (LMD)

1.1. Adăugare o nouă înregistrare

1.2. Actualizarea datelor dintr-o tabelă

**1.3. Ștergerea tuplurilor dintr-o tabelă**

1.4. Instrucțiunea Merge

### 2. Limbajul de control al datelor (LCD). Tranzacții

## 7.1.3. Ștergerea tuplurilor dintr-o tabelă

Se pot șterge tupluri dintr-o tabelă utilizând clauza **DELETE**.

### Sintaxa

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

## 7.1.3. Ștergerea tuplurilor dintr-o tabelă

În sintaxa:

- **tabela** - numele tabelului
- **condiție** - identifică liniile care trebuie șterse și este compusă din:
  1. nume de câmpuri
  2. expresii
  3. constante
  4. subquery-uri
  5. și operatori de comparație

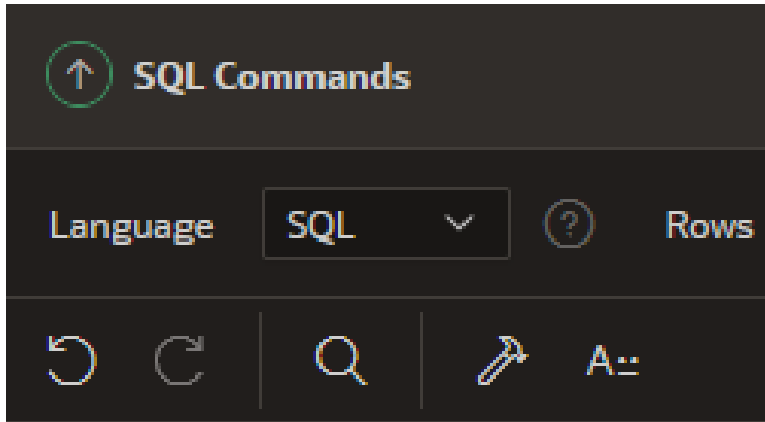
## 7.1.3. Ștergerea tuplurilor dintr-o tabelă

- Șterge anumite tupluri dintr-o tabelă specificând clauza **WHERE** în declarația funcției **DELETE**.
- Se poate confirma operația de ștergere prin afișarea tuplurilor șterse cu ajutorul declarației lui **SELECT**.

## 7.1.3. Ștergerea tuplurilor dintr-o tabelă

În exemplul următor șterge angajații care lucrează în departamentul 10 din tabela Angajati.

```
DELETE FROM emp  
WHERE deptno = 10;
```



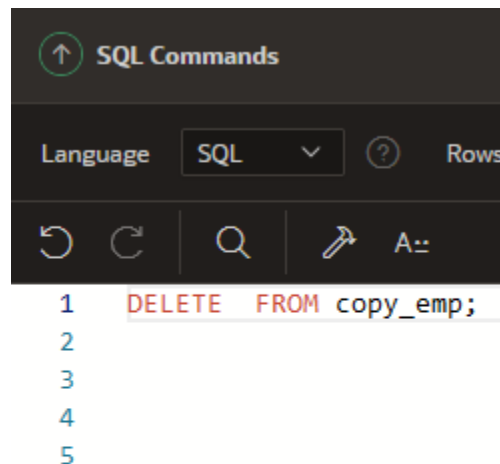
The screenshot shows a dark-themed SQL command window. At the top, it says "SQL Commands" with an upward arrow icon. Below that, there's a "Language" dropdown menu set to "SQL" and a "Rows" label with a question mark icon. A toolbar contains icons for undo, redo, search, and insert. The main area displays the SQL command:

```
1 DELETE FROM emp  
2 WHERE deptno = 10;
```

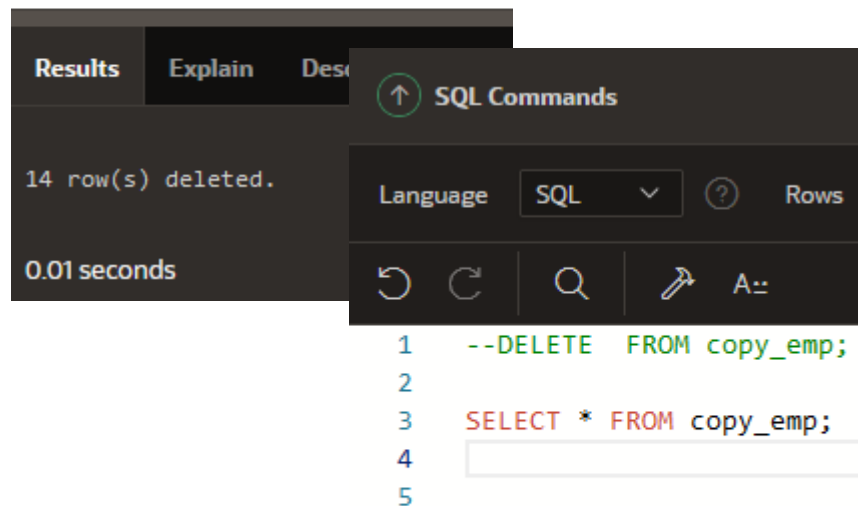
## 7.1.3. Ștergerea tuplurilor dintr-o tabelă

- Dacă se omite clauza **WHERE** toate câmpurile din tabelă vor fi șterse.
- Al doilea exemplu șterge toate tuplurile (înregistrările) din tabela Copy\_emp deoarece nu a fost specificată clauza **WHERE**.

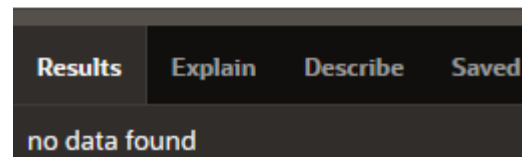
**DELETE FROM** copy\_emp;



```
SQL Commands
Language SQL
1 DELETE FROM copy_emp;
2
3
4
5
```



```
Results Explain Describe
14 row(s) deleted.
0.01 seconds
SQL Commands
Language SQL
1 --DELETE FROM copy_emp;
2
3 SELECT * FROM copy_emp;
4
5
```

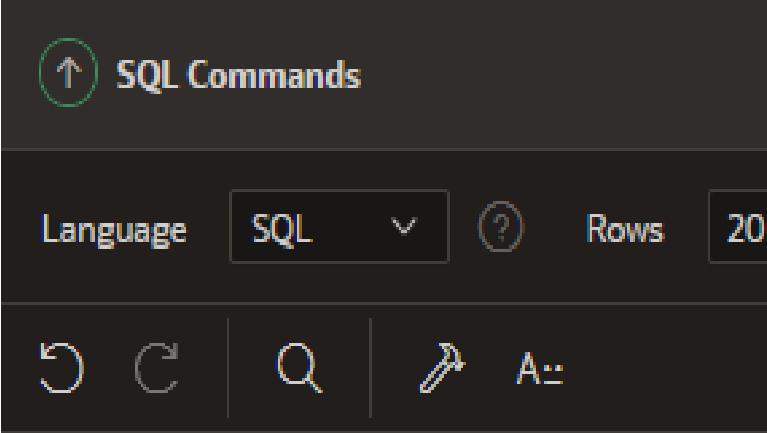


```
Results Explain Describe Saved
no data found
```

## 7.1.3. Ștergerea tuplurilor dintr-o tabelă

Se pot șterge și mai multe linii.

```
DELETE FROM emp  
WHERE deptno IN (10, 20);
```



The screenshot shows a dark-themed SQL command window titled "SQL Commands". It features a "Language" dropdown menu set to "SQL", a "Rows" dropdown menu set to "20", and several icons for navigation and execution. Below the interface, the following SQL command is displayed and executed:

```
1 DELETE FROM emp  
2 WHERE deptno IN (10, 20);
```

S-au șters angajații care lucrează în departamentele identificate cu deptno specificate

# *Limbajul SQL*

## Capitolul 7

### 1. Limbajul de manipulare al datelor (LMD)

1.1. Adăugare o nouă înregistrare

1.2. Actualizarea datelor dintr-o tabelă

1.3. Ștergerea tuplurilor dintr-o tabelă

**1.4. Instrucțiunea Merge**

### 2. Limbajul de control al datelor (LCD). Tranzacții

## 7.1.4. Instrucțiunea Merge

Instrucțiunea **MERGE** permite *inserarea sau actualizarea condiționată a datelor* într-un/dintr-un tabel al bazei de date.

## 7.1.4. Instrucțiunea Merge

Sintaxa ei simplificată este următoarea:

```
MERGE INTO tabel_destinatie [alias]
USING {tabel_sursa | vizualizare | subinterogare} [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);
```

## 7.1.4. Instrucțiunea Merge

Instrucțiunea efectuează:

**UPDATE** dacă înregistrarea există deja în tabel

**INSERT** dacă înregistrarea este nouă.

**Obs:** *DEFAULT* reprezintă valoarea implicită a unei coloane, dacă a fost definită la crearea tabelului. Altfel este *null*.

## 7.1.4. Instrucțiunea Merge

### Exemplu

- Să se șteargă din tabelul *copy\_emp* toți angajații care câștigă comision.
- Să se actualizeze data angajării (*SYSDATE*).
- Să se introducă sau să actualizeze datele din tabelul *copy\_emp* pe baza tabelului *EMP*.
- La fiecare pas, analizați conținutul tabelului *copy\_emp*.

## 7.1.4. Instrucțiunea Merge

```
MERGE INTO copy_emp x
USING emp e
ON ( x.empno = a.empno )
WHEN MATCHED THEN
  UPDATE SET
    x.ename = e.ename,
    x.job = e.job,
    x.hiredate = e.hiredate,
    x.sal = e.sal,
    x.comm = e.comm,
    x.mgr = e.mgr,
    x.deptno = e.deptno
```

## 7.1.4. Instrucțiunea Merge

**WHEN NOT MATCHED THEN  
INSERT VALUES (e.empno, e.ename, e.job,  
e.hiredate, e.sal, e.comm, e.mgr, e.deptno );**

```
1  MERGE INTO copy_emp x
2      USING emp e
3      ON ( x.empno = e.empno )
4      WHEN MATCHED THEN
5          UPDATE SET
6              x.ename = e.ename,
7              x.job = e.job,
8              x.hiredate = e.hiredate,
9              x.sal = e.sal,
10             x.comm = e.comm,
11             x.mgr = e.mgr,
12             x.deptno = e.deptno
13     WHEN NOT MATCHED THEN
14         INSERT VALUES (e.empno, e.ename, e.job, e.hiredate, e.sal, e.comm, e.mgr, e.deptno );
```

# *Limbajul SQL*

## Capitolul 7

### 1. Limbajul de manipulare al datelor (LMD)

1.1. Adăugare o nouă înregistrare

1.2. Actualizarea datelor dintr-o tabelă

1.3. Ștergerea tuplurilor dintr-o tabelă

1.4. Instrucțiunea Merge

### 2. Limbajul de control al datelor (LCD). Tranzacții

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### Procesarea unei Tranzactii

*O tranzacție este o operație asupra unei baze de date care implică una sau mai multe modificări în una sau mai multe tabele.*

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

Există două clase de tranzacții:

1. Tranzacții **DML (Data Manipulation Language)** care conțin un număr oarecare de blocuri **DML** și pe care **ORACLE** le tratează ca o singură entitate sau o singură unitate logica de lucru
2. Tranzacții **DDL (Data Definition Language)** care conțin un singur bloc **DDL**

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

- Nu pot exista situații "jumătate de drum" în timpul execuției unei tranzacții, așa încât unele modificări specificate în tranzacție să fie aplicate bazei de date și altele nu.
- Pentru fiecare tranzacție ori toate modificările sunt aplicate bazei de date, ori nici una din modificări nu este îndeplinită (sunt toate abandonate - **discarded** ).

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

O tranzacție începe când prima comandă executabilă **DML** sau **DDL** este întâlnită și se termină în una din următoarele situații:

1. Întâlnește **COMMIT/ROLLBACK**
2. Comanda **DDL** se termină
3. Anumite erori (**DEADLOCK**)
4. **EXIT** - ieșire din SQL\*Plus
5. Eroare sistem
6. Un bloc **DDL** este executat automat și de aceea implicit încheie o tranzacție
7. După încheierea unei tranzacții, următorul bloc executabil **SQL** va lansa automat următoarea tranzacție

# 7.2. Limbajul de control al datelor (LCD). Tranzacții

## Tranzacții

Tip	Descriere
<b>Limbajul de manipulare al datelor (DML)</b>	Este constituit din mai multe cereri DML pe care serverul ORACLE le trateaza ca pe o singura entitate sau o unitate logica
<b>Limbajul de definire al datelor (DDL)</b>	Este format dintr-o singura cerere DDL
<b>Limbajul de control al datelor (DCL)</b>	Este format dintr-o singura cerere DCL

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### Permanentizarea Modificărilor

- Pentru ca modificările să rămână permanente, ele trebuie executate asupra bazei de date.
- Comanda **COMMIT** realizează permanentizarea modificărilor;
- Comanda **ROLLBACK** permite să abandonăm sau să anulăm modificările.
- **Modificarea/modificările, executate asupra bazei de date între 2 comenzi **COMMIT** reprezintă o tranzacție.**
- Până când tranzacția nu este executată, nici una din modificări nu este vizibilă utilizatorilor.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### Înlăturarea modificărilor nedorite

- Modificările neexecutate pot fi abandonate prin comanda **ROLLBACK**.
- **ROLLBACK** va atribui datelor valorile care acestea le aveau imediat după executarea ultimului **COMMIT** prin anularea tuturor modificărilor făcute după ultimul **COMMIT**.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### Semnificația tranzacțiilor

- **ORACLE** asigură consistența datelor bazată pe tranzacții.
- Tranzacțiile dau utilizatorului mai multă flexibilitate și control la lucrul asupra datelor, și asigură consistența datelor în cazul unei erori a procesului utilizator sau a unei erori de sistem.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

- Tranzacțiile ar trebui să conțină doar acele comenzi **DML** care realizează o singură modificare asupra datelor.
- De exemplu un transfer de fonduri (să spunem 1000\$) între 2 conturi ar trebui să implice un debit al unui cont de 1000\$ și un credit al altui cont de 1000\$.
- Ambele acțiuni ar trebui să se încheie cu succes sau să dea eroare împreună.
- Creditul nu ar trebui executat fără debit.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### Controlul tranzacțiilor cu instrucțiuni **SQL**

Următoarele instrucțiuni **SQL** sunt utilizate când apar execuții (**commit**) sau refaceri (**rollback**):

- **COMMIT[WORK]**
- **SAVEPOINT nume\_savepoint**
- **ROLLBACK[WORK] to [SAVEPOINT] nume\_savepoint**

De notat că, **COMMIT** și **ROLLBACK** sunt instrucțiuni (blocuri) **SQL**.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

Cele 3 blocuri **SQL** utilizate pentru controlul tranzacțiilor sunt explicate mai jos:

Sintaxa: **COMMIT[WORK];**

- Permanentizeaza schimbările în tranzacția curentă
- Șterge toate punctele de salvare (**Savepoint**) din tranzacție
- Termină tranzacția
- Eliberează toate blocările (**Lock**) tranzacției
- Cuvântul cheie **WORK** este opțional

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

- Utilizatorul trebuie să explicitizeze sfârșitul tranzacției în programul aplicație utilizând **COMMIT** (sau **ROLLBACK**).
- Dacă nu se execută explicit tranzacția și programul se termină anormal, ultima tranzacție executată va fi anulată.
- Execuții implicite (**commit**) apar în următoarele situații:
  1. înainte de o comandă **DDL**
  2. după o comandă **DDL**
  3. la închiderea normală a unei baze de date

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

Blocurile **DDL** cauzează mereu execuții (**commit**) în timpul execuției lor.

Dacă introduceți un bloc **DDL** după câteva blocuri **DML**, blocul **DDL** cauzează apariția unui **commit** înaintea propriei execuții, încheind tranzacția curentă.

Astfel dacă blocul **DDL** este executat până la capăt, este și înregistrat.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### SAVEPOINT

Sintaxa:

**SAVEPOINT** nume\_savepoint

*Exemplu:*

*SAVEPOINT terminare\_actualizari*

Poate fi utilizat pentru a împărți o tranzacție în bucăți mai mici.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

- Punctele de salvare (**savepoints**) permit utilizatorului să rețină toată munca sa la orice moment din timp, cu opțiunea de a înregistra mai târziu totul sau a anula totul sau o parte din ea.
- Astfel, pentru o tranzacție lungă, se pot salva părți din ea, pe măsura execuției, la sfârșit înregistrându-se sau refăcându-se conținutul inițial.
- La apariția unei erori nu trebuie executat din nou fiecare bloc.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

- La crearea unui nou punct de salvare cu același nume ca al unuia dinainte, primul punct este șters.
- Numărul maxim de puncte de salvare pentru un proces utilizator este implicit 5.
- Aceasta limită poate fi schimbată.

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

### ROLLBACK

**ROLLBACK[WORK] to [SAVEPOINT]  
nume\_punct\_salvare**

Instrucțiunea ROLLBACK este utilizată pentru a reface un lucru (o prelucrare asupra informațiilor dintr-o baza de date).

## 7.2. Limbajul de control al datelor (LCD). Tranzacții

- Cuvântul cheie "**work**" este opțional.
- Întoarcerea la un punct de salvare este de asemenea opțională.
- Dacă se utilizează **ROLLBACK** fără clauza **TO SAVEPOINT**, atunci:
  1. se termina tranzacția
  2. se anulează modificările din tranzacția curentă
  3. șterge toate punctele de salvare din tranzacție
  4. eliberează blocările tranzacției

# Probleme propuse

1. Sa se mareasca cu 3% salariul angajatului cu prenumele "ADAMS".

(2 solutii: una cu **Select** si una cu **Update**)

# Probleme propuse

2. Creați o copie a tabelii EMP cu numele Copy\_EMP care să conțină toate câmpurile tabelii originale.

Afișați din tabela Copy\_EMP numele, salariul și departamentul pentru toți angajații care au salariul cuprins între 1000 și 2000.

# Probleme propuse

3. Actualizati salariile tuturor angajaților din tabela Copy\_EMP, prin indexare cu 10%.

Afișați angajații cu noile salarii în ordinea descrescătoare a acestora.

# Întrebări?