

UCB: Universitatea Constantin Brâncuși din Târgu-Jiu  
Automatică și Informatică Aplicată

# Baze de date

## Limbajul SQL

Adrian Runcceanu



THE INFORMATION COMPANY

# *Curs 8*

## *Limbajul SQL*

# *Limbajul SQL*

## Constrângeri

Serverul **Oracle** utilizează constrângeri pentru a preveni pătrunderea de date invalide în tabele.

Putem utiliza constrângeri pentru a realiza următoarele acțiuni:

1. Impune reguli datelor unei tabele ori de câte ori un rând este inserat, modificat sau șters din tabela.
2. Prevenirea ștergerii unei tabele în cazul în care există dependență de alte tabele
3. Furnizarea regulilor pentru instrumentele **Oracle**, cum ar fi **ORACLE DEVELOPER**.

# Tipuri de constrângeri

Constrangere	Descriere
<b>NOT NULL</b>	specifica faptul ca o coloana nu poate avea valoarea nula
<b>UNIQUE</b>	specifica o coloana sau o combinatie de coloane a carei valori trebuie sa fie unice pentru toate randurile din tabel
<b>PRIMARY KEY</b>	identifica fiecare rand al tableei
<b>FOREIGN KEY</b>	stabileste o relatie de cheie straina intre coloana si coloana tableei de referinta
<b>CHECK</b>	specifica o conditie care trebuie sa fie adevarata

# Ghidul Constrângerilor

1. Toate constrângerile sunt cuprinse într-un dicționar.
2. Este ușor să se facă referință la constrângeri dacă li se dă nume sugestive.
3. Numele unei constrângeri trebuie să urmeze un anumit standard.
4. Dacă nu se denumește constrângerea, server-ul **Oracle** generează un nume de forma **SYS\_Cn**, unde **n** este un număr întreg astfel încât numele constrângerii este unic.
5. Constrângerile definite pentru o anumita tabelă pot fi vizualizate în **USER\_CONSTRAINTS** (dicționarul tabelei).

# Definirea Constrângerilor

## Sintaxa

**CREATE TABLE [schema.] table**

**(column datatype [DEFAULT expr]**

**[column\_constraint],**

**[table\_constraint][...]);**

În sintaxa prezentată anterior avem:

<b>schema</b>	este aceeasi ca si numele titularului
<b>table</b>	este numele tablei
<b>DEFAULT expr</b>	specifica o valoare predefinita care sa fie utilizata daca o valoare este omisa in declaratia INSERT
<b>column</b>	este numele coloanei
<b>datatype</b>	este tipul datei si lungimea coloanei
<b>column_constraint</b>	este o <b>constrangere de integritate ca parte a definitiei coloanei</b>
<b>table_constraint</b>	este o <b>constrangere de integritate ca parte a definitiei tablei</b>

# EXEMPLU - Adăugarea unei constrângeri unei tabele odată cu crearea lui.

SQL Commands

Language SQL ? Rows

↻ ⌂ Q ↗ A↔

```
1 CREATE TABLE EMP1(
2   empno number(6) NOT NULL,
3   ename varchar2(20),
4   sal number(4),
5   mgr number(4),
6   job varchar2(20),
7   PRIMARY KEY(empno));
```

**CREATE TABLE** EMP1(  
empno number(6) **NOT NULL**,  
ename varchar2(20),  
sal number(4),  
mgr number(4),  
job varchar2(20),  
**PRIMARY KEY(empno));**

Constrângere  
pentru coloană

Constrângere  
pentru tabela

- De obicei constrângerile sunt create în același timp cu tabela.
- Constrângerile pot fi adăugate tabeliei după crearea ei.
- Constrângerile pot fi definite pe 2 nivele:

Nivelul constrangerii	Descriere
Coloana	Face referire la o singura coloana; poate defini orice tip de constrangere de integritate
Tabel	Face referire la una sau mai multe coloane; poate defini orice constrangere exceptand pe cea de tip NOT NULL

Sintaxa:

## 1. Constrângere la nivel de coloană

```
column [CONSTRAINT constraint_name] constraint_type
```

## 2. Constrângere la nivel de tabela

```
column,..  
[CONSTRAINT constraint_name] constraint_type  
(column,...),
```

- În sintaxa avem:

<i>constraint_name</i>	este numele constrangerii
<i>constraint_type</i>	este tipul constrangerii

# Constrângerea NOT NULL

- Constrângerea de tip **NOT NULL** asigură faptul că o coloană să nu conțină valoarea nulă.
- Ea poate fi specificată la nivel de coloană și nu la nivel de tabela.

# Exemplu

În exemplul următor constrângerea **NOT NULL** se aplică coloanelor ENAME și HIREDATE din tabela **emp\_new**.

- Pentru coloana ENAME constrângerea nu este denumită astfel încât serverul **Oracle** o să creeze un nume pentru ea.
- Pentru coloana HIREDATE constrângerea este denumită: "**NOT NULL**".

```
CREATE TABLE emp_new(
EMPNO number(6),
ENAME varchar2(10) NOT NULL,
SAL number(4),
COMM number(4),
HIREDATE number(4) NOT
NULL);
```

The screenshot shows a SQL command interface with the following details:

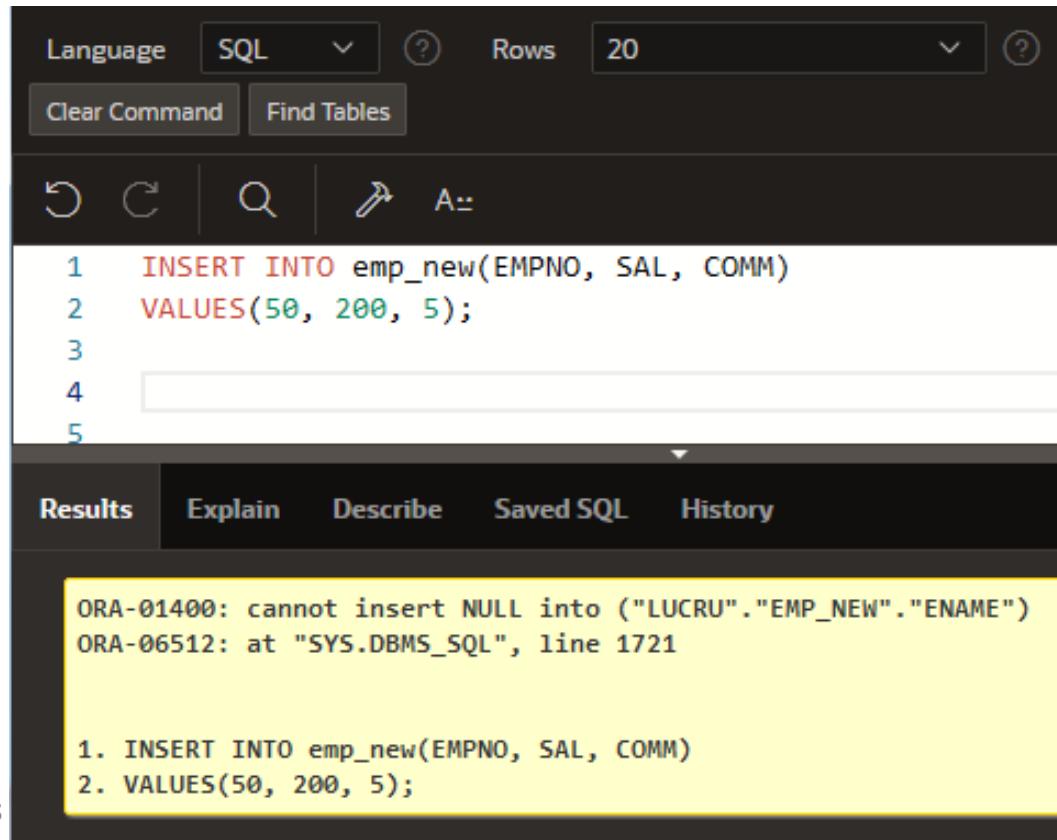
- SQL Commands**: The title of the interface.
- Language**: Set to SQL.
- Rows**: Set to 20.
- Toolbar icons**: Includes back, forward, search, and other database management icons.
- Code Area**: Displays the SQL code for creating the 'emp\_new' table, with line numbers 1 through 7.

```
1 CREATE TABLE emp_new(
2 EMPNO number(6),
3 ENAME varchar2(10) NOT NULL,
4 SAL number(4),
5 COMM number(4),
6 HIREDATE number(4) NOT NULL);
7
```

- Results Tab**: Active tab, showing the message "Table created."
- Timing**: 0.02 seconds.

Acum vom încerca să inserăm valori doar în coloanele EMPNO, SAL, COMM, dar la execuție ne va da eroare pentru că ENAME și HIREDATE au valori nule iar constrângerile ne obligă să le atribuim o valoare.

**INSERT INTO** emp\_new(EMPNO, SAL, COMM)  
**VALUES**(50, 200, 5);



The screenshot shows a SQL developer interface with the following details:

- Language:** SQL
- Rows:** 20
- Clear Command** and **Find Tables** buttons
- Toolbar icons:** Undo, Redo, Search, Insert, Paste
- SQL Editor:** Contains the following code:

```
1  INSERT INTO emp_new(EMPNO, SAL, COMM)
2  VALUES(50, 200, 5);
3
4
```
- Results Tab:** Shows the error message:

```
ORA-01400: cannot insert NULL into ("LUCRU"."EMP_NEW"."ENAME")
ORA-06512: at "SYS.DBMS_SQL", line 1721
```
- Log Tab:** Shows the executed statements:

```
1. INSERT INTO emp_new(EMPNO, SAL, COMM)
2. VALUES(50, 200, 5);
```

# Constrângerea **UNIQUE**

*Constrângerea **UNIQUE** de integritate impune ca fiecare valoare a unei coloane sau set de coloane să fie unică - două rânduri ale aceluiași tabele să nu aibă aceleași valori într-o anumită coloană sau set de coloane.*

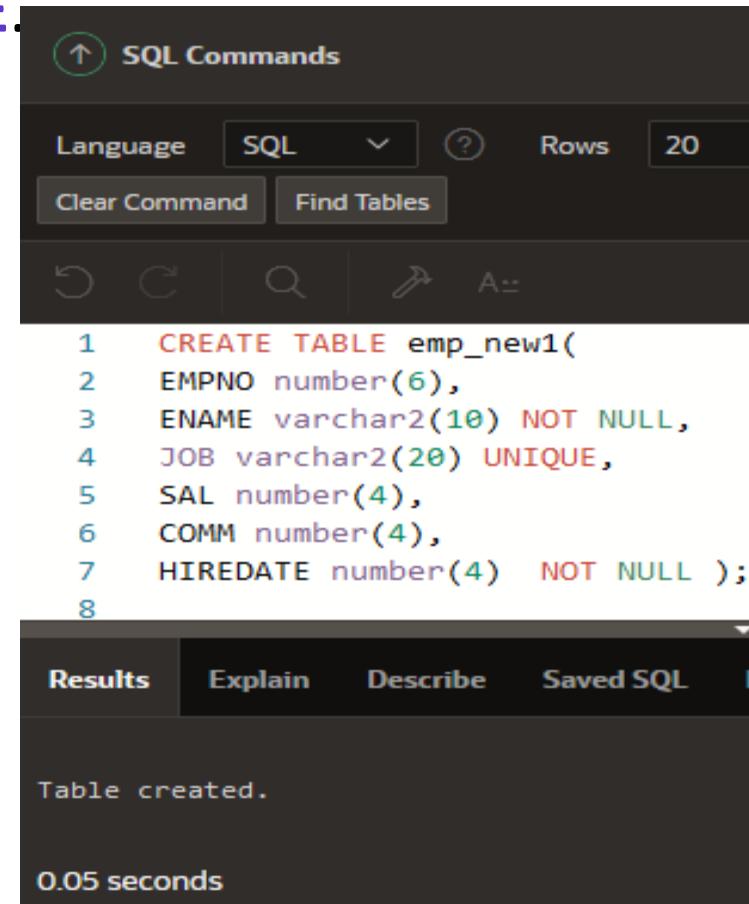
Permite includerea de valori nule numai dacă constrângerea **NOT NULL** nu este definită pentru aceeași coloană (valoarea nulă nu este considerată a fi echivalentă cu ceva).

Constrângerea **UNIQUE** poate fi definită atât la nivel de linie cât și la nivel de tabelă.

## Exemplu

- În exemplul de mai jos se aplică constrângerea **UNIQUE** coloanei JOB a tabelei **emp\_new1**.
- Numele constrângerii este **UNIQUE**.

```
CREATE TABLE emp_new1(
EMPNO number(6),
ENAME varchar2(10) NOT NULL,
JOB varchar2(20) UNIQUE,
SAL number(4),
COMM number(4),
HIREDATE number(4) NOT NULL );
```



The screenshot shows a SQL command interface with the following details:

- SQL Commands** tab is selected.
- Language: SQL
- Rows: 20
- Buttons: Clear Command, Find Tables
- Toolbar icons: Undo, Redo, Search, Edit, Sort
- SQL code (numbered 1-8):

```

1  CREATE TABLE emp_new1(
2    EMPNO number(6),
3    ENAME varchar2(10) NOT NULL,
4    JOB varchar2(20) UNIQUE,
5    SAL number(4),
6    COMM number(4),
7    HIREDATE number(4) NOT NULL );
8

```
- Results** tab is selected.
- Output: Table created.
- Time: 0.05 seconds

# Constrângerea Cheie Primară

- Constrângerea **CHEIE PRIMARĂ(PRIMARY KEY)** creează o cheie primară pentru tabela.
- Numai o singură cheie poate fi creată pentru fiecare tabelă.
- Această constrângere este o coloană sau un set de coloane care identifică în mod unic fiecare rând al tabelului.
- Nici o coloană care face parte din cheia primară nu poate conține valoarea nulă.
- Poate fi definită la nivel de coloană sau tabelă.

*O tabela poate avea o singură cheie primară dar poate avea mai multe constrângeri de tip **UNIQUE**.*

## Exemplu

În exemplul de mai jos este definită o constrângere de tip cheie primară la nivelul coloanei DEPTNO  
Numele constrângerii este **PRIMARY KEY**.

```
CREATE TABLE DEPT1 (
    DEPTNO number(6),
    DNAME varchar2(10),
    MGR number(6),
    PRIMARY KEY(DEPTNO) );
```

The screenshot shows a SQL command window with the following details:

- Header: SQL Commands, Language: SQL, Rows: 6.
- Buttons: Clear Command, Find Tables.
- Toolbar icons: Undo, Redo, Search, Insert, Paste.
- SQL Editor:

```
1 CREATE TABLE DEPT1 (
2 DEPTNO number(6),
3 DNAME varchar2(10),
4 MGR number(6),
5 PRIMARY KEY(DEPTNO) );
```
- Bottom tabs: Results (selected), Explain, Describe, Saved.
- Results pane: Table created.
- Timing: 0.04 seconds.

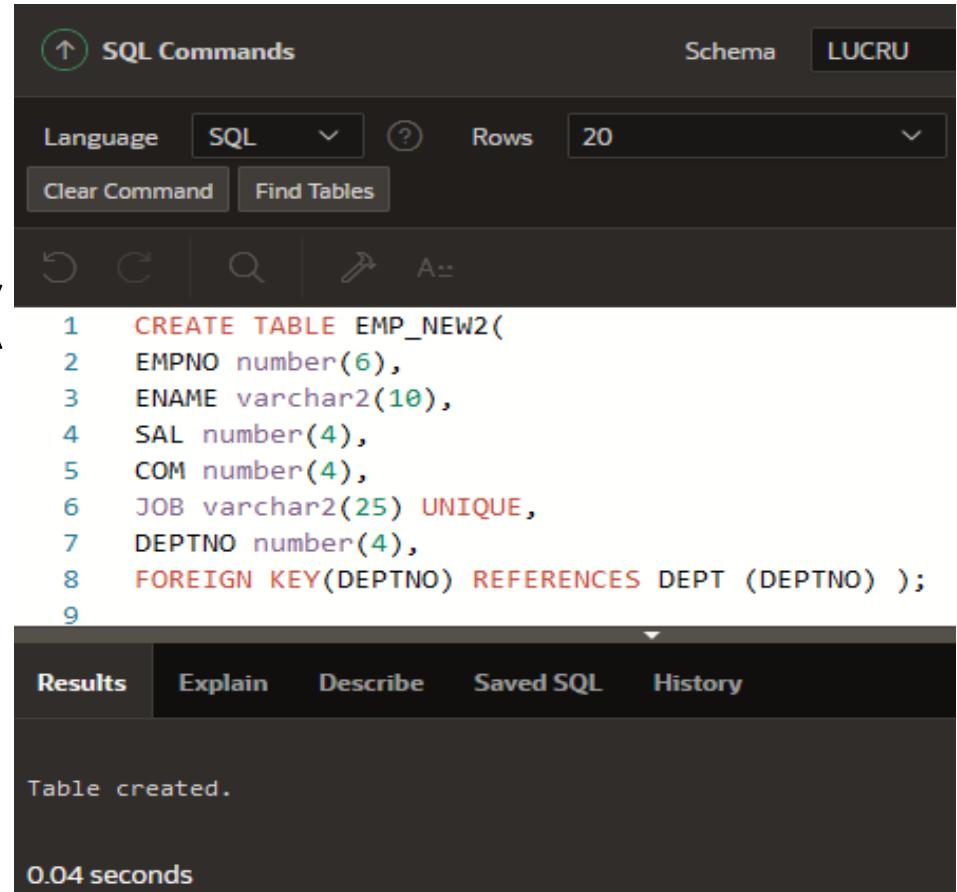
# Constrângerea FOREIGN KEY

- Constrângerea **FOREIGN KEY** definește o coloană sau o combinație de coloane ca **foreign key** și *stabilește o relație între o cheie primară și una unică în aceeași tabelă sau în tabele diferite.*
- O valoare care apare într-o tabelă trebuie să se regăsească și în cea de-a 2-a tabelă, pe coloana unde formează cheia primară.
- Constrângerile de tip **FOREIGN KEY** pot fi definite la nivel de coloană sau tabela.

# Exemplu

În următorul exemplu se definește o constrângere de tip **FOREIGN KEY** coloanei DEPTNO a tablei EMP\_NEW2 utilizând sintaxa la nivel de tabela.

```
CREATE TABLE EMP_NEW2(
    EMPNO number(6),
    ENAME varchar2(10),
    SAL number(4),
    COM number(4),
    JOB varchar2(25) UNIQUE,
    DEPTNO number(4),
FOREIGN KEY(DEPTNO) REFERENCES DEPT (DEPTNO) );
```



The screenshot shows the SQL Commands tab of an Oracle SQL developer interface. The schema is set to 'LUCRU'. The SQL command entered is:

```

1  CREATE TABLE EMP_NEW2(
2  EMPNO number(6),
3  ENAME varchar2(10),
4  SAL number(4),
5  COM number(4),
6  JOB varchar2(25) UNIQUE,
7  DEPTNO number(4),
8  FOREIGN KEY(DEPTNO) REFERENCES DEPT (DEPTNO) );
9

```

The Results tab displays the message "Table created." and a execution time of "0.04 seconds".

Se poate defini și la nivel de coloană. Sintaxa este următoarea:

```
CREATE TABLE EMP_NEW3(
    EMPNO number(6),
    ENAME varchar2(10),
    SAL number(4),
    COM number(4),
    JOB varchar2(25) UNIQUE,
    DEPTNO number(4)
    REFERENCES DEPT (DEPTNO) );
```

The screenshot shows an SQL command interface with the following details:

- SQL Commands** tab is selected.
- Language**: SQL
- Rows**: 20
- Clear Command** and **Find Tables** buttons are present.
- Toolbar icons**: Undo, Redo, Search, Edit, Sort.
- Code Area**: Shows the SQL code for creating the table, with line numbers 1 through 9. The word **UNIQUE** is highlighted in blue.

```
1 CREATE TABLE EMP_NEW3(
2     EMPNO number(6),
3     ENAME varchar2(10),
4     SAL number(4),
5     COM number(4),
6     JOB varchar2(25) UNIQUE,
7     DEPTNO number(4)
8     REFERENCES DEPT (DEPTNO) );
9
```

- Results Tab** is selected, showing the output: "Table created."
- Timing**: 0.03 seconds

- După cum am observat a dispărut din sintaxa **FOREIGN KEY**.
- O constrângere de tip **FOREIGN KEY** este definită într-o tabela copil, iar tabela care conține coloana la care se face referință este părintele.

O **FOREIGN KEY** este definită utilizând o combinație a următoarelor cuvinte cheie:

- **FOREIGN KEY** este utilizată pentru a defini o coloană în tabelul copil la nivel de tabela
- **REFERENCES** identifică tabela și coloana în tabela părinte
- **ON DELETE CASCADE** indică faptul că atunci când rândul din tabela părinte va fi șters, rândul dependent din tabela copil va fi de asemenea șters.
- **ON DELETE SET NULL** convertește valorile **FOREIGN KEY** în valori nule atunci când valoarea părinte este ștearsă.

# Constrângerea de tip **CHECK**

Constrângerea de tip **CHECK** definește o condiție pe care fiecare rând trebuie să o îndeplinească.

Următoarele expresii nu sunt permise:

- Referințe la pseudocoloanele **CURRVAL**, **NEXTVAL**, **LEVEL** și **ROWNUM**.
  - Apelul funcțiilor **SYSDATE**, **UID**, **USER**, și **USERENV**.
  - Cereri care se referă la alte valori ale altor rânduri
- Nu există un număr limitat de constrângeri de tip **CHECK** pe care să le definim pe o coloană.

Constrângerea de tip **CHECK** poate fi definită atât la nivel de coloană cât și la nivel de tabela.

# Exemplu

```
CREATE TABLE EMP_NEW4(  
EMPNO number(6),  
ENAME varchar2(10),  
SAL number(4) CHECK(sal > 0));
```

The screenshot shows an SQL command window with the following interface elements:

- Header: SQL Commands
- Language: SQL
- Buttons: Clear Command, Find Tables
- Toolbar: Undo, Redo, Search, Edit, Align
- Text Area:

```
1 CREATE TABLE EMP_NEW4(  
2 EMPNO number(6),  
3 ENAME varchar2(10),  
4 SAL number(4) CHECK(sal > 0) );  
5  
6
```
- Results Tab: Results (selected), Explain, Describe, Saved SQL
- Output:

Table created.  
0.03 seconds

# Adăugarea unei constrângeri

Se poate adăuga o constrângere pentru o tabelă existentă utilizând **ALTER TABLE** și clauza **ADD**.

## Sintaxa

```
ALTER TABLE table  
ADD [CONSTRAINT constraint] type(column);
```

În sintaxă avem:

<b>table</b>	este numele tebelului
<b>constraint</b>	este numele constrangerii
<b>type</b>	este tipul constrangerii
<b>column</b>	este numele coloanei afectate de constrangere

## Observații

- Se poate adăuga, șterge, activa sau dezactiva o constrângere, dar nu-i putem modifica structura.
- Se poate adăuga o constrângere **NOT NULL** la o coloană existentă utilizând clauza **MODIFY** a declarației **ALTER TABLE**.

## Exemplu

*Se poate adăuga o constrângere și unei tabele existent (nu numai odată cu crearea lui).*

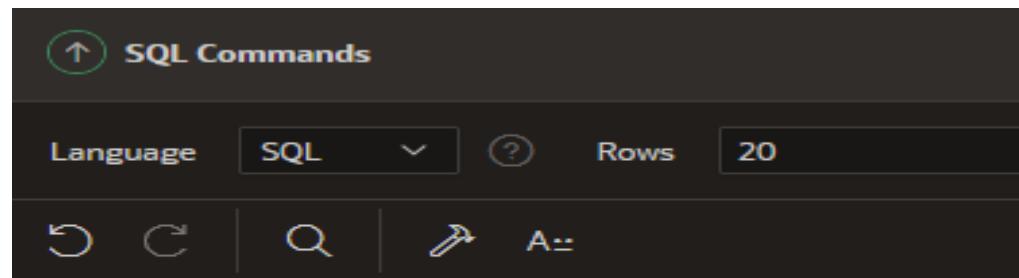
În următorul exemplu vom crea o constrângere **FOREIGN KEY** în tabela EMP.

Constrângerea asigură existența unui manager dacă există angajat în tabela EMP.

**ALTER TABLE** EMP

**ADD CONSTRAINT** FK\_Mgr

**FOREIGN KEY**(Mgr) **REFERENCES** EMP (empno)



```
SQL Commands
Language: SQL | Rows: 20
_undo | _redo | _search | _refresh | A:_
1 ALTER TABLE EMP
2 ADD CONSTRAINT FK_Mgr
3 FOREIGN KEY (Mgr) REFERENCES EMP (empno)
```

# Ștergerea unei constrângeri

Pentru a șterge o constrângere trebuie utilizată declarația **ALTER TABLE** cu clauza **DROP**.

Opțiunea **CASCADE** a clauzei **DROP** face ca și constrângerea dependentă să fie ştearsă.

## Sintaxa

```
ALTER TABLE table  
DROP PRIMARY KEY| UNIQUE (column)|  
CONSTRAINT constraint [CASCADE];
```

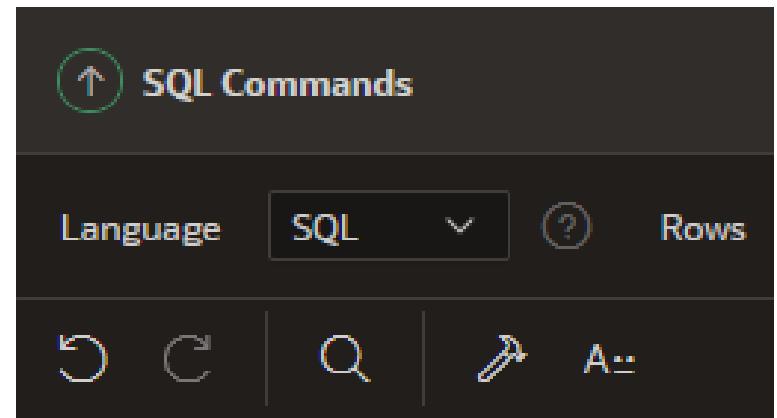
În sintaxa avem

<b>table</b>	este numele tabelului
<b>constraint</b>	este numele constrangerii
<b>column</b>	este numele coloanei afectate de constrangere

## Exemplu

1. În exemplul următor vom șterge constrângerea la nivelul coloanei MGR din tabela.

```
ALTER TABLE EMP  
DROP CONSTRAINT FK_Mgr
```



The screenshot shows a SQL command interface with the following elements:

- Header: "SQL Commands" with an upward arrow icon.
- Language dropdown: Set to "SQL".
- Buttons: Refresh, Copy, Search, Paste, and Auto.
- Text area:
  - 1 ALTER TABLE EMP
  - 2 DROP CONSTRAINT FK\_Mgr

2. În următorul exemplu vom şterge constrângerea cheie primară din tabela DEPT și cheia străină asociată coloanei DEPTNO din tabela emp\_new2.

**ALTER TABLE DEPT  
DROP PRIMARY KEY CASCADE**

The screenshot shows a SQL command interface with the following details:

- SQL Commands** tab is selected.
- Language**: SQL
- Rows**: 3
- Toolbar icons**: Undo, Redo, Search, Insert, Sort.
- SQL Editor Content**:

```
1 ALTER TABLE DEPT
2 DROP PRIMARY KEY CASCADE
3
```
- Results Tab**: Shows the output:

```
Table altered.

0.07 seconds
```
- Other Tabs**: Explain, Describe, Saved.

# Dezactivarea unei constrângerii

Dezactivarea constrângerii se efectuează cu declarația **ALTER TABLE** însorită de clauza **DISABLE**.

## Sintaxa

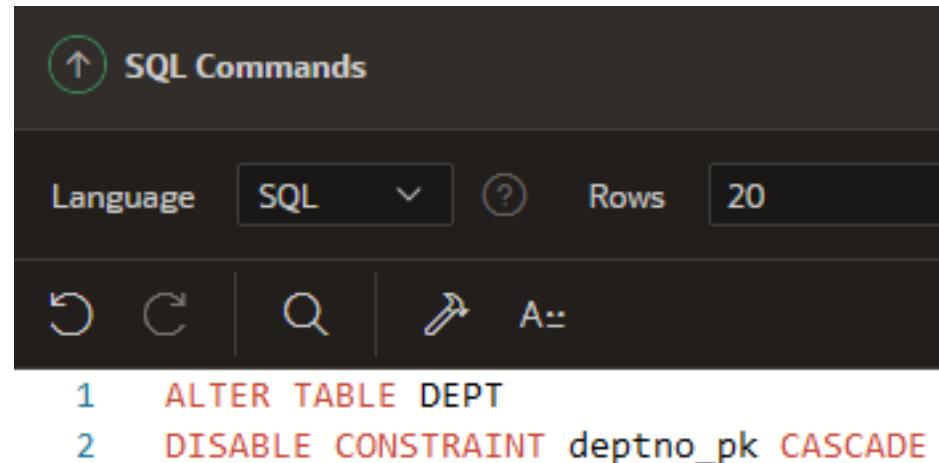
```
ALTER TABLE table  
DISABLE CONSTRAINT constraint  
[CASCADE];
```

În sintaxa avem:

<b>table</b>	este numele tabelului
<b>constraint</b>	este numele constrangerii

# Observații

- Se poate utiliza clauza **DISABLE** atât în declarația **CREATE TABLE** cât și în **ALTER TABLE**.
- Clauza **CASCADE** dezactivează constrângeri de integritate dependente



The screenshot shows a SQL command interface with the following details:

- SQL Commands** button is visible.
- Language**: SQL
- Rows**: 20
- Toolbar icons: Undo, Redo, Search, Insert, Alter.
- SQL code:
  - 1 ALTER TABLE DEPT
  - 2 DISABLE CONSTRAINT deptno\_pk CASCADE

Exemplu

**ALTER TABLE DEPT**

**DISABLE CONSTRAINT** deptno\_pk **CASCADE**

# Activarea unei constrângeri

Se poate activa o constrângere fără a o șterge sau recreea utilizând **ALTER TABLE** cu clauza **ENABLE**.

## Sintaxa

```
ALTER TABLE table
ENABLE CONSTRAINT constraint [CASCADE];
```

În sintaxa avem:

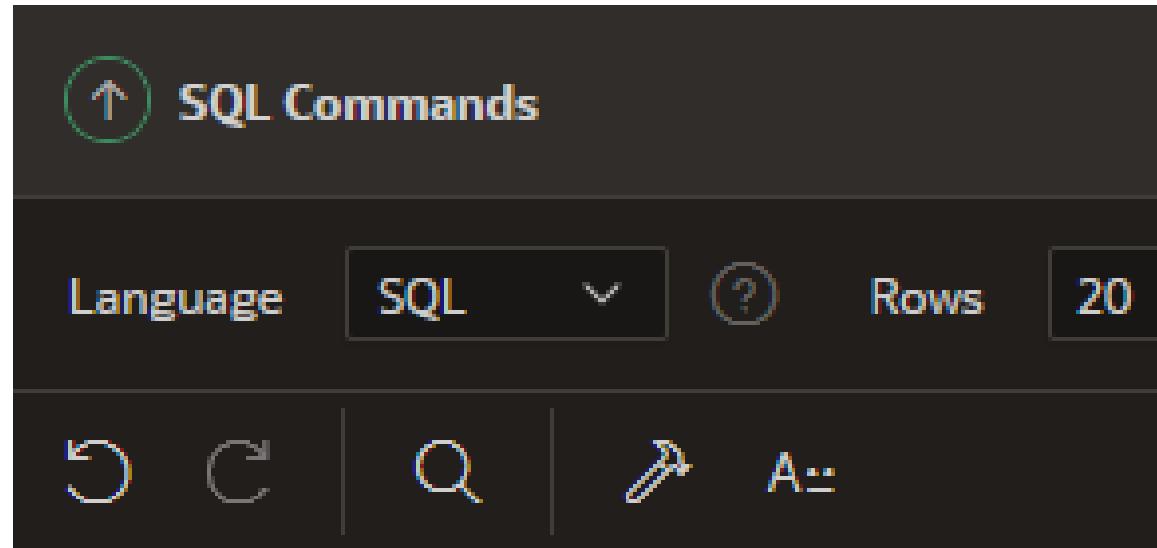
<b>table</b>	este numele tabelului
<b>constraint</b>	este numele constrangerii

# Observații

1. Dacă se activează o constrângere, această constrângere se aplică la toate datele din tabela.
1. Dacă se activează o constrângere **UNIQUE** sau **PRIMARY** se creează automat un index **UNIQUE** sau **PRIMARY**.
1. Clauza **ENABLE** se poate utiliza în ambele declarații **CREATE TABLE** cât și **ALTER TABLE**.

# Exemplu

**ENABLE CONSTRAINT deptno\_pk**



The screenshot shows a dark-themed SQL command interface. At the top, there is a green circular icon with an upward arrow and the text "SQL Commands". Below this is a toolbar with a "Language" dropdown set to "SQL", a help icon, and a "Rows" input field set to "20". At the bottom, there are several icons: a refresh circle, a circular arrow, a magnifying glass, a wrench, and an "A±" icon. A numbered line editor at the bottom contains the command "1 ENABLE CONSTRAINT deptno\_pk".

1      **ENABLE CONSTRAINT deptno\_pk**

# Constrângerile Cascadate

- Constrângerile de tip **CASCADE** sunt utilizate cu clauza **DROP COLUMN**.
- Constrângerea **CASCADE** *șterge toate constrângerile de integritate ce se referă la cheile primare și unice definite în coloanele șterse.*
- Șterge de asemenea toate constrângerile multicolone definite în coloanele șterse.

# Vizualizare Constrângeri

- După ce creem o tabelă putem verifica existența lui utilizând o comandă **DESCRIBE**.
- Singura constrângere ce se poate verifica este constrângerea **NOT NULL**.
- Pentru a se vizualiza toate constrângările din tabelă trebuie interogat tabela **USER-CONSTRAINTS**.

## Exemplu

```
SELECT constraint_name, table_name  
FROM user_constraints  
WHERE table_name = 'emp_new'
```

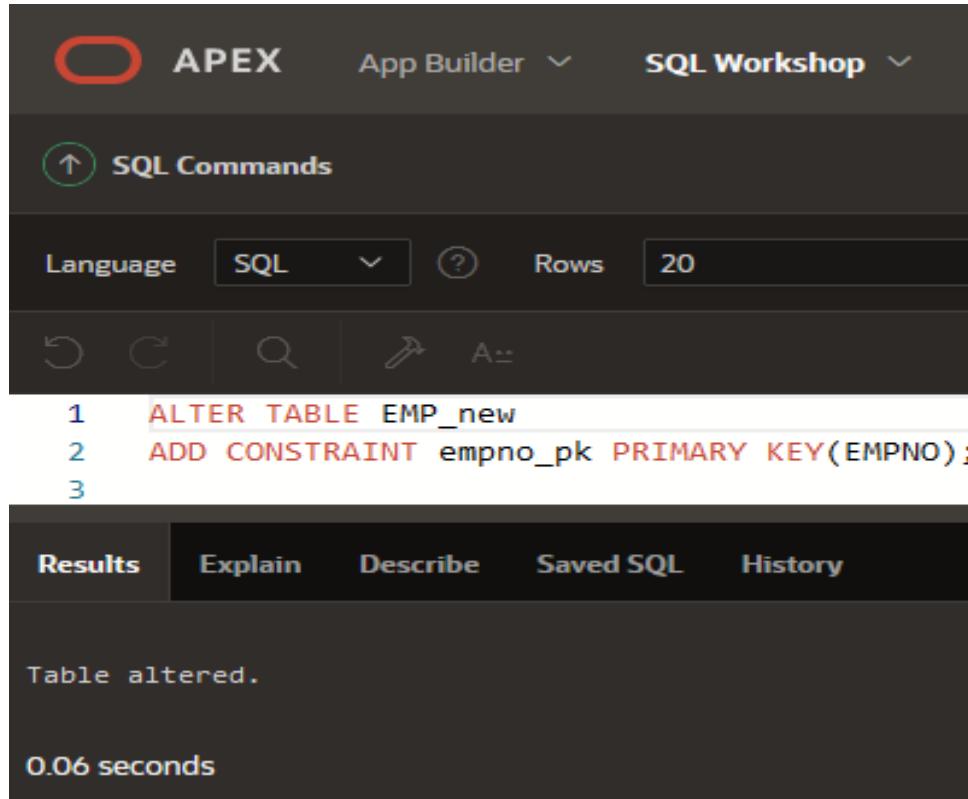
## Exerciții Propuse

1. Adăugați o constrângere **PRIMARY KEY** tablei **EMP\_NEW** pe coloana **EMPNO**. Constrângerea trebuie unică la creare. Numele constrângerii este empno\_pk.
2. Ștergeți constrângerea creată mai sus.
3. Vizualizați constrangerile din tabela **EMP\_NEW**.

# Soluție 1:

**ALTER TABLE EMP\_NEW**

**ADD CONSTRAINT empno\_pk PRIMARY KEY(EMPNO);**



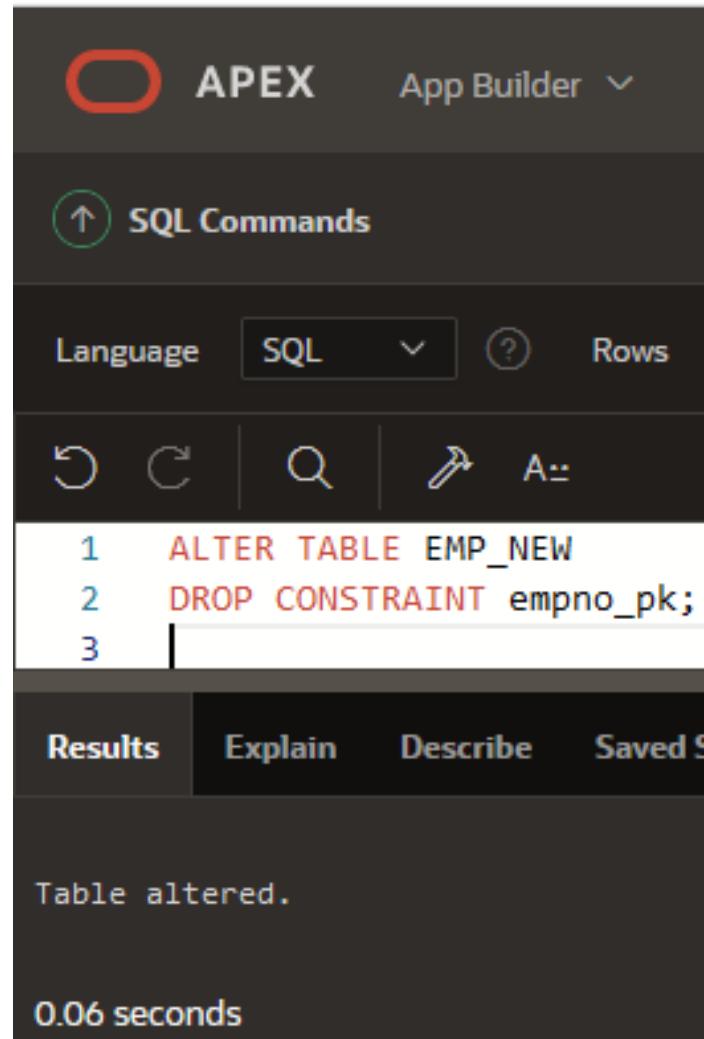
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', and 'SQL Workshop'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL' and 'Rows' set to '20'. Below the title, there are icons for back, forward, search, and refresh. The SQL editor contains three numbered lines of code:

```
1 ALTER TABLE EMP_new
2 ADD CONSTRAINT empno_pk PRIMARY KEY(EMPNO);
3
```

Below the code, a results panel displays the message 'Table altered.' and a performance metric of '0.06 seconds'.

## Soluție 2:

```
ALTER TABLE EMP_NEW  
DROP CONSTRAINT empno_pk;
```



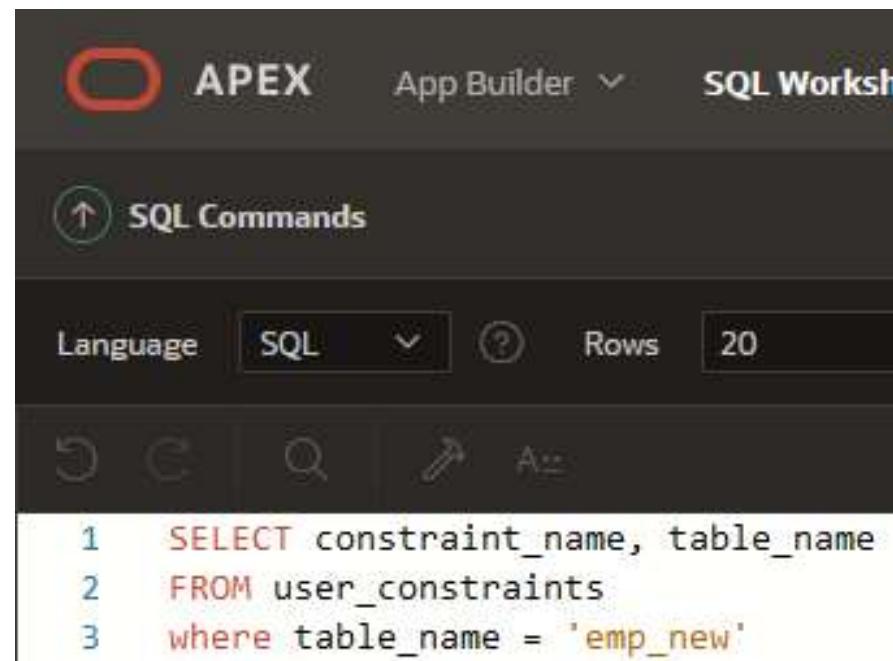
The screenshot shows the Oracle APEX App Builder interface with the "SQL Commands" tab selected. The language is set to SQL. The code entered is:

```
1 ALTER TABLE EMP_NEW  
2 DROP CONSTRAINT empno_pk;  
3
```

The results section shows the output: "Table altered." and a execution time of "0.06 seconds".

## Soluție 3:

```
SELECT constraint_name, table_name  
FROM user_constraints  
WHERE table_name = 'emp_new'
```



The screenshot shows the Oracle SQL Workshop interface. The title bar includes the APEX logo, App Builder, and SQL Workshop tabs. The main area is titled "SQL Commands". The language is set to SQL, and the results page size is 20. Below the toolbar are standard database navigation icons: back, forward, search, and refresh. The SQL code is displayed in the bottom pane:

```
1  SELECT constraint_name, table_name  
2  FROM user_constraints  
3  where table_name = 'emp_new'
```



# Întrebări?