

MySQL

• Partea III. Limbajul SQL (I)

Ce ne așteaptă?

1. Introducere în SQL
2. Variabile SQL
3. Operatori SQL
4. Instrucțiuni de definiție - CREATE
5. Instrucțiuni de definiție - ALTER
6. Instrucțiuni de definiție - DROP

1. Introducere în SQL

Ce este SQL?

- **SQL (Structured Query Language)** – singurul limbaj standardizat ce permite stocarea, manipularea și accesarea datelor bazelor de date
- **SQL a fost standardizat în anii 80 dar fiecare producător implementează SQL cu anumite abateri și prin aplicarea extinderilor specifice ceea ce nu îl face în totalitate compatibil între diferite DBMS**
- **Acest curs va aborda bazele implementării MySQL ale standardului SQL**
- **SQL permite:**
 - **Crearea și ștergere bazelor de date**
 - **Crearea, modificarea și ștergere tabelelor în baza de date**
 - **Introducerea, actualizarea, accesarea și ștergerea datelor în tabele**
 - **Crearea vederilor și a procedurilor stocate**
 - **Setarea permisiunii de acces la tabele, vedere, proceduri, etc**

Caracteristici de sintaxa SQL

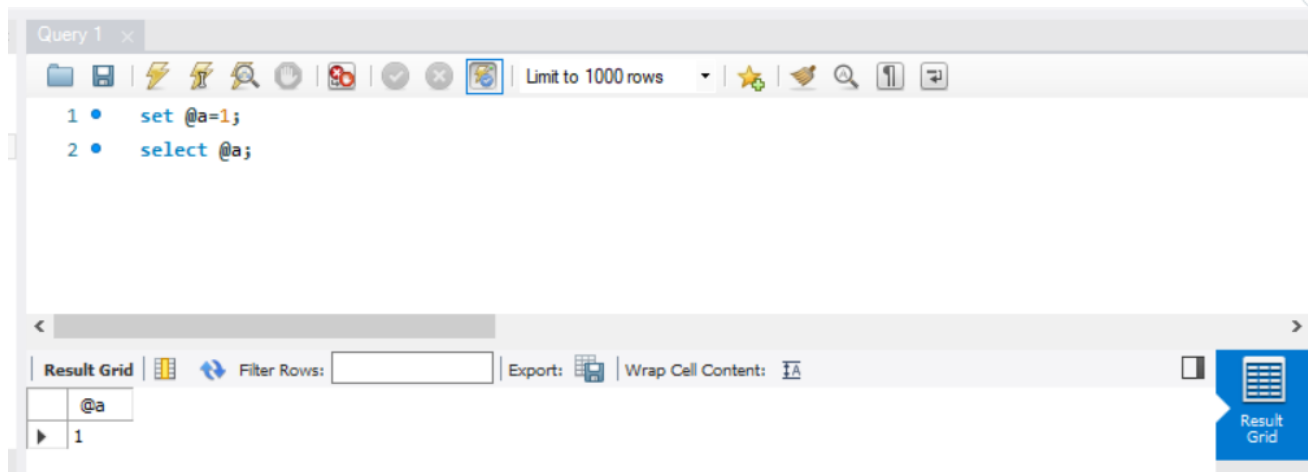
- SQL nu este sensibil la majuscule și minuscule;
- Fiecare interogare SQL (în caz că există mai multe) trebuie să se termine cu semnul ; (punct și virgulă), iar acest separator se poate modifica;
- Stringurile se scriu exclusiv între ghilimele simple (') sau în unele cazuri între ghilimele duble (");
- În MySQL se utilizează ghilimele backtick pentru denumiri de tabele și coloane dacă acestea sunt din mai multe cuvinte sau un cuvânt cheie;
- Variabilele în SQL încep întotdeauna cu semnul @.
- Comentariile uni-linie încep cu unul din simbolurile “#” sau “--” și continuă până la sfârșitul liniei
- Comentariile multi-linie încep cu unul din simbolurile “/*” și sfârșesc cu simbolul “*/”
- Comentariile condiționate încep cu unul din simbolurile “/*!”, sfârșesc cu simbolul “*/”, iar serverul MySQL le consideră parte a interogării

Tipuri de instrucțiuni SQL

- **Instrucțiuni de definiție (DDL – Data Definition Language)** – se folosesc pentru gestionarea obiectelor pe server, adică pentru crearea, modificarea și ștergerea bazei, tabelelor, vederilor, procedurilor stocate etc.
- Instrucțiunile DDL sunt CREATE, ALTER și DROP și de obicei sunt accesibile doar administratorilor bazei de date
- **Instrucțiuni de manipulare (DML – Data Manipulation Language)** – se folosesc pentru introducerea, accesarea, actualizarea și ștergerea datelor din baza.
- Instrucțiunile DML sunt SELECT, INSERT, UPDATE și DELETE și se mai numesc instrucțiuni CRUD (Create, Read, Update, Delete)
- **Instrucțiuni de control (DCL – Data Control Language)** – se folosesc pentru manipularea drepturilor utilizatorilor pe server.
- Instrucțiunile DCL sunt GRANT și REVOKE

Instrucțiuni SQL în MySQL Workbench

- Instrucțiunile SQL se introduc în câmpul Interogări (Query)



- Pentru execuția întregului script sau a porțiunii selectate a acestuia se selectează butonul
- Pentru execuția linie script unde este plasat cursorul se selectează butonul
- Pentru salvarea scriptului în fișier .sql se selectează butonul
- Pentru încărcarea unui script în fișier .sql se selectează butonul

2. Variabile SQL

Sintaxa variabilelor

- **Variabila - obiect în memorie prin intermediul căruia se apelează valoarea ce i-a fost atribuită.**
- **Reguli de sintaxă a variabilelor în SQL**
 - **În fața fiecărei variabile, trebuie să se găsească semnul @ (pentru diferențiere de denumirile coloanelor);**
 - **Nu este permis ca variabila să conțină un caracter special**
 - **Variabila din SQL se poate începe și cu un număr**
 - **Variabila trebuie să aibă o denumire intuitivă**

Definirea variabilelor MYSQL

- Pentru definirea variabilelor în SQL se folosește cuvântul-cheie **SET** și operatorul de atribuire = sau **:=**

```
SET @a='MySQL!'; sau SET @a:= 'MySQL!';
```

- Se pot defini mai multe variabile pe o singură linie cu un singur SET

```
SET @a=10,@b=20,@c=30;
```

- Variabilele pot fi de tip numeric sau de tip string, iar tipul acestora este determinat automat de MySQL
- Pentru afișarea valorii variabilei se utilizează instrucțiunea **SELECT**

```
SET @a=10;
```

```
SELECT @a;
```

- Definirea și afișarea poate fi realizată pe o singură linie cu utilizarea ;

```
SET @a=10; SELECT @a;
```


Exemple de variabile SQL

- Definirea și afișarea variabilei

```
SET @a=10;
```

```
SELECT @a;
```

@a
10

- Valoarea variabilei se poate modifica sau atribui unei alte variabile

```
SET @a=10;
```

```
SET @b=@a;
```

```
SET @a=15;
```

```
SELECT @a,@b;
```

@a	@b
15	10

- Atribuirea variabilei a unei valori care este rezultatul unei interogări

```
SET @a = (SELECT 2+2);
```

```
SELECT @a
```

@a
4

- Ordinea afișării depinde de ordinea în SELECT și nu de ordinea definirii

```
SET @a=10,@b='MySQL!';
```

```
SELECT @a,@b;
```

@a	@b
10	MySQL!

```
SET @a=10,@b='MySQL!';
```

```
SELECT @b,@a;
```

@b	@a
MySQL!	10

3. Operatori SQL

Operatori de atribuire

- Operatorii de atribuire asigură alocarea unei valori variabilei
- Operatorii de atribuire sunt “=” și “:=”
- Operatorul “=” trebuie să fie însoțit de cuvântul cheie SET pentru a se considera operator de atribuire în caz contrar va fi operator de comparație
- Operatorul “:=” nu necesită cuvântul cheie SET și permite atribuirea valorilor variabilelor și în cadrul altor interogări
- Operatorul “=” în cadrul interogării SELECT

```
SELECT @a=15;
```

@a
0

- Operatorul “:=” în cadrul interogării SELECT

```
SELECT @a:=15;
```

@a
15

Operatori de comparație

- Lista operatorilor de comparație

Operator	Semnificație
=	este egal
<	mai mic decât
>	mai mare decât
<=	mai mic sau egal
>=	mai mare sau egal
<>	nu este egal

- Exemple operatorilor de comparație

@a<>@b – oferă valoarea 1 numai dacă a nu este egal cu b

@a<@b – oferă valoarea 1 numai dacă a este mai mic decât b

@a>@b – oferă valoarea 1 numai dacă a este mai mare decât b

@a<= @b – oferă valoarea 1 numai dacă a este mai mic sau egal cu b

@a>= @b – oferă valoarea 1 numai dacă a este mai mare sau egal cu b

@a<>@b – oferă valoarea 1 numai dacă a nu este egal cu b

Operatori aritmetici

- Lista operatorilor aritmetici

Operator	Descriere
DIV	operatorul de împărțire cu rezultat întreg
/	operatorul de împărțire cu rezultat flotant
-	operatorul de scădere
% or MOD	operatorul modul (restul ce rămâne de la împărțire)
+	operatorul de adunare
*	operatorul de înmulțire
-	operatorul de schimbare a semnului argumentului

- Exemple operatorilor aritmetici

```
SET @a=10;  
SELECT @a+3;  
SELECT @a-3;  
SELECT @a*3;  
SELECT @a/3;  
SELECT @a div 3;  
SELECT @a%3;
```

Operatori logici

- Lista operatorilor logici

Operator	Semnificație
AND, &&	ȘI logic
NOT, !	negație
, OR	SAU logic
XOR	SAU cu excludere

- Exemple operatorilor logici

```
SET @a=1;  
SELECT @a != 2 AND @a != 3;  
SELECT @a != 2 && @a != 3;  
SELECT @a < 2 OR @a > 3;  
SELECT @a < 2 || @a > 3;  
SELECT !@a < 2;  
SELECT NOT@a < 2;  
SET @a=2,@b=3;  
SELECT @a = 2 XOR @b = 3;
```

4. Instrucțiuni de definiție - CREATE

Crearea unei baze cu parametrii implicați

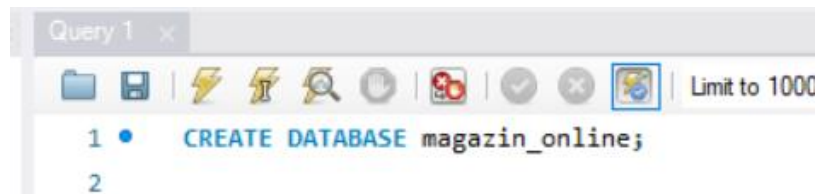
- Instrucțiunea **CREATE** – creează obiecte în baza de date (baza de date, tabele, vederi, proceduri stocate, etc)
- Crearea bazei de date cu parametrii implicați

```
CREATE DATABASE nume_baza;
```

sau

```
CREATE SCHEMA nume_baza;
```

- Exemplu



Crearea unei baze cu setări

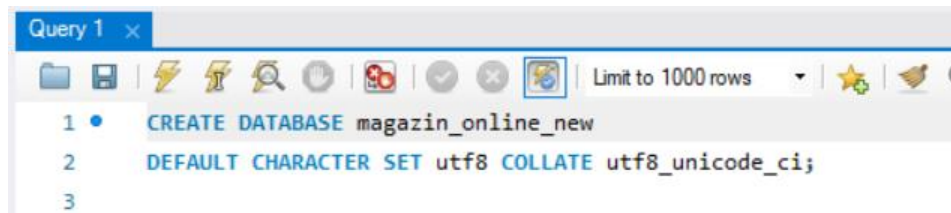
- Crearea bazei de date cu parametrii implicați și verificarea inexistenței unei baze cu acest nume

```
CREATE DATABASE IF NOT EXISTS nume_baza;
```

- Crearea bazei de date cu setarea setului de caractere și reguli de corelare

```
CREATE DATABASE nume_baza DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

- Exemplu



Crearea unui tabel cu parametrii implicați

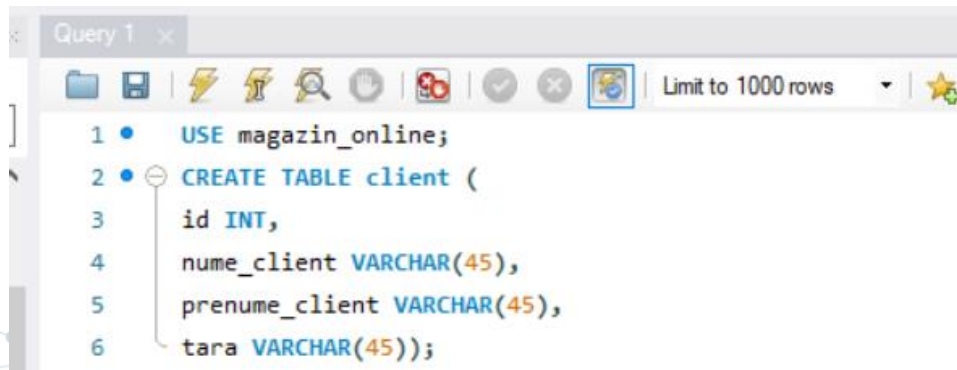
- Pentru crearea unui tabel inițial se selectează baza de date unde se va crea acesta

`USE nume_baza;`

- Crearea unui tabel cu parametrii implicați

```
CREATE TABLE nume_tabel (  
    nume_coloana_1 tip_date,  
    nume_coloana_2 tip_date ,  
    nume_coloana_3 tip_date,  
    nume_coloana_4 tip_date );
```

- Exemplu

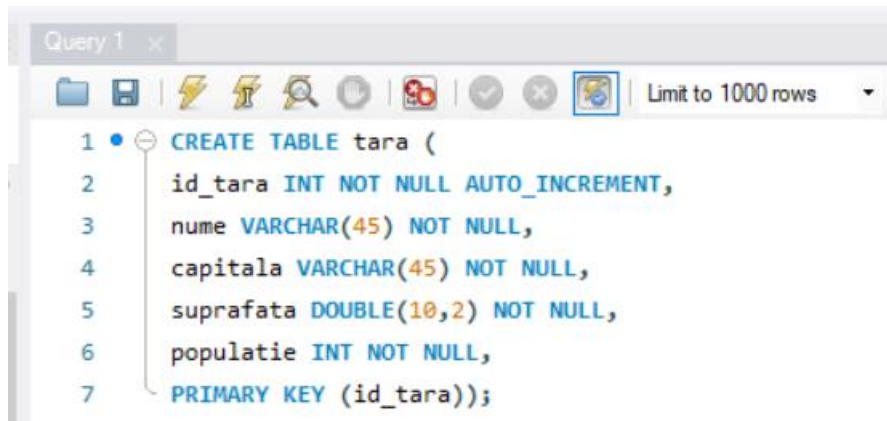


Crearea unui tabel cu setări

- Crearea unui tabel cu setarea parametrilor coloanelor

```
CREATE TABLE nume_tabel (  
  nume_coloana_1 INT NOT NULL AUTO_INCREMENT,  
  nume_coloana_2 tip_date NOT NULL,  
  nume_coloana_3 tip_date NOT NULL,  
  nume_coloana_4 tip_date,  
  PRIMARY KEY (nume_coloana_1 ));
```

- Exemplu



```
Query 1 x  
1 • CREATE TABLE tara (  
2   id_tara INT NOT NULL AUTO_INCREMENT,  
3   nume VARCHAR(45) NOT NULL,  
4   capitala VARCHAR(45) NOT NULL,  
5   suprafata DOUBLE(10,2) NOT NULL,  
6   populatie INT NOT NULL,  
7   PRIMARY KEY (id_tara));
```

Vizualizarea instrucțiunilor de crearea

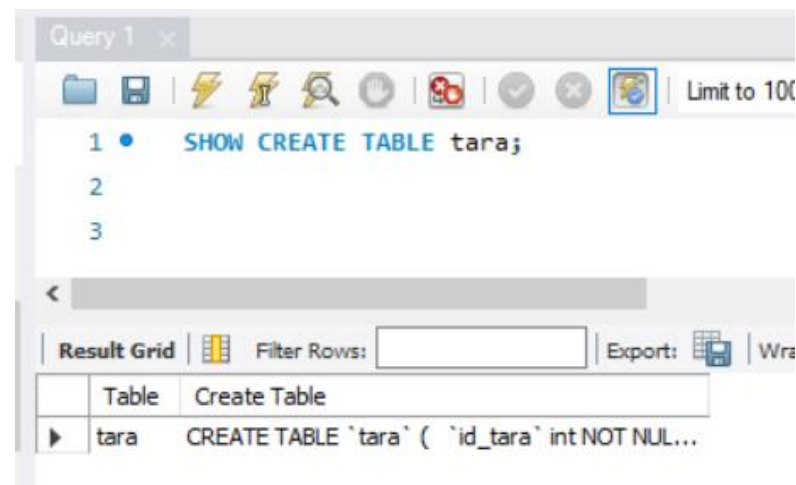
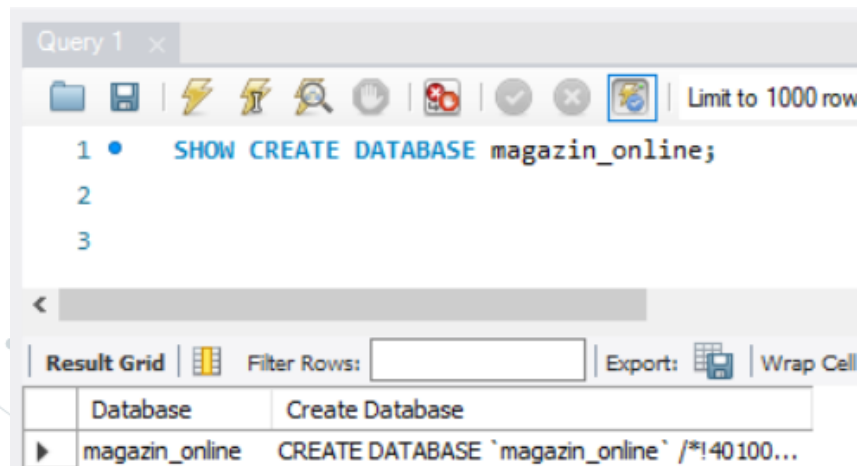
- Instrucțiunea **SHOW CREATE** – reîntoarce script de creare a obiectelor bazei de date

`SHOW CREATE DATABASE nume_baza`

sau

`SHOW CREATE TABLE nume_tabel`

- Exemple



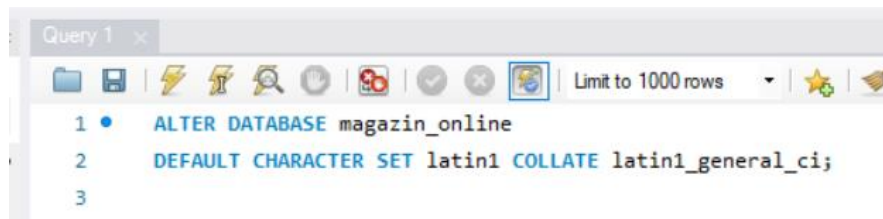
5. Instrucțiuni de definiție - ALTER

Modificări a setărilor bazei de date

- Instrucțiunea ALTER – modifica obiectele bazei de date (nu si datele)
- Modificarea bazei de date – se modifica setul de caractere si regulile de corelare (nu modifica denumirea)

```
ALTER DATABASE nume_baza DEFAULT CHARACTER SET latin1 COLLATE latin1_general_ci;
```

- Exemplu

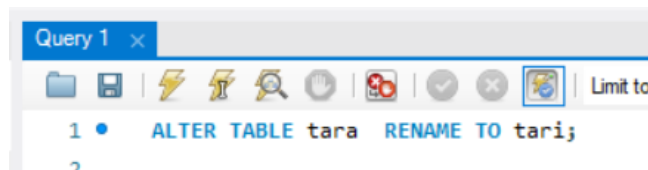


Modificări a setărilor tabelului (1)

- **Modificarea numelui tabelului**

```
ALTER TABLE nume_tabel RENAME TO nume_tabel_nou;
```

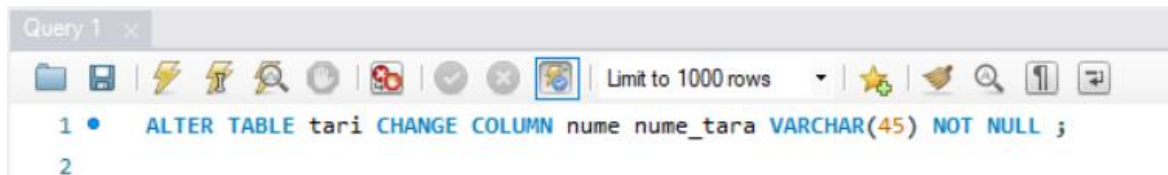
- **Exemplu**



- **Modificarea numelui unei coloane a tabelului**

```
ALTER TABLE nume_tabel  
CHANGE COLUMN nume_coloana nume_coloana_nou tip_date optiuni;
```

- **Exemplu**

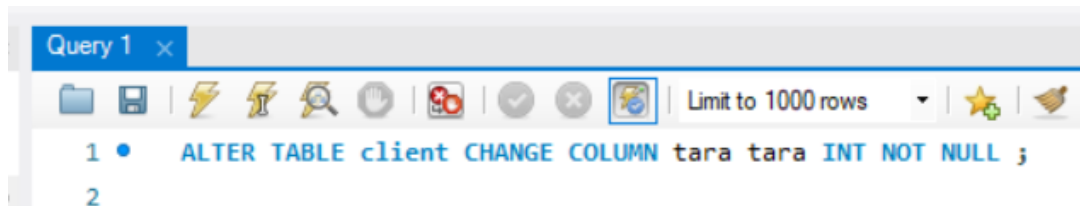


Modificări a setărilor tabelului (2)

- **Modificarea tipului de date a unei coloane a tabelului**

```
ALTER TABLE nume_tabel  
CHANGE COLUMN nume_coloana nume_coloana tip_date_nou optiuni;
```

- **Exemplu**



- **Adăugarea unei coloane tabelului**

```
ALTER TABLE nume_tabel ADD COLUMN nume_coloana tip_date optiuni;
```

- **Exemplu**

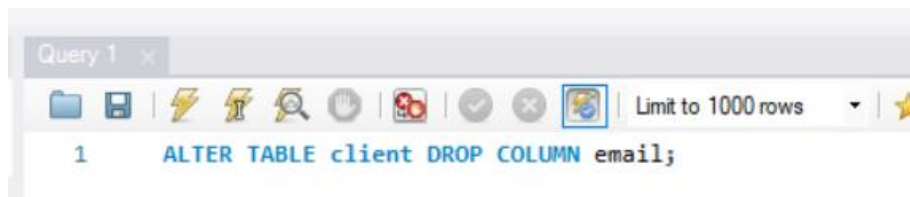


Modificări a setărilor tabelului (3)

- Ștergerea unei coloane a tabelului

```
ALTER TABLE nume_tabel DROP COLUMN nume_coloana;
```

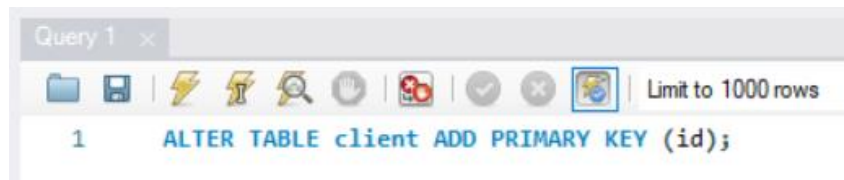
- Exemplu



- Setarea cheii primare

```
ALTER TABLE nume_tabel ADD PRIMARY KEY (nume_coloana);
```

- Exemplu

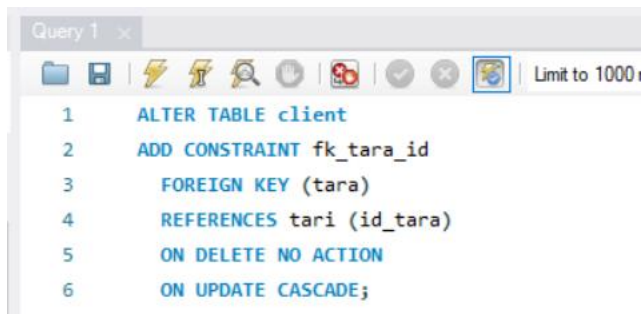


Setarea cheii străine

- Setarea cheii străine *nume_cheie* astfel încât în tabelul curent *nume_tabel* coloana *nume_coloana_1* să se prezinte datele din tabelul de legătură *nume_tabel_legat* coloana *nume_coloana_2* cu opțiuni de actualizarea și ștergere

```
ALTER TABLE nume_tabel  
ADD CONSTRAINT nume_cheie  
    FOREIGN KEY (nume_coloana_1)  
    REFERENCES nume_tabel_legat (nume_coloana_2 )  
    ON DELETE optiuni  
    ON UPDATE optiuni;
```

- Exemplu: setarea cheii străine *fk_tara_id* ce leagă coloana *tara* din tabelul *client* cu coloana *id_tara* din tabelul *tari*



```
Query 1 x  
1 ALTER TABLE client  
2 ADD CONSTRAINT fk_tara_id  
3 FOREIGN KEY (tara)  
4 REFERENCES tari (id_tara)  
5 ON DELETE NO ACTION  
6 ON UPDATE CASCADE;
```

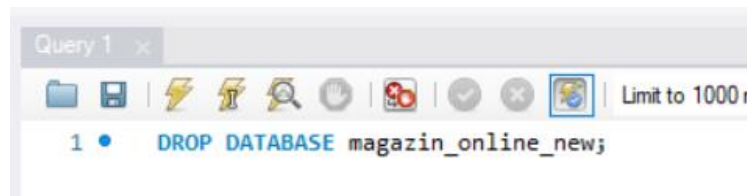
6. Instrucțiuni de definiție - DROP

Ștergerea bazei de date

- Instrucțiunea **DROP** – se utilizează pentru ștergerea obiectelor date de baze
- Ștergerea bazei de date

`DROP DATABASE nume_baza;`

- Exemplu

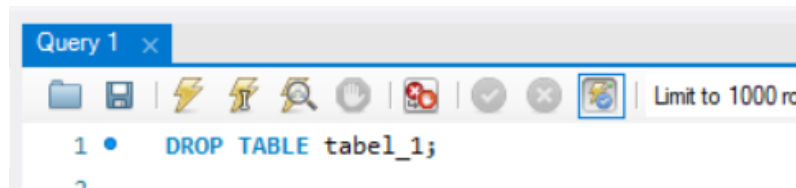


Ștergerea unui tabel al bazei de date

- Ștergerea unui tabel

`DROP TABLE nume_tabel;`

- Exemplu (necesită crearea anterioară a unui tabel cu numele tabel_1)



- Ștergerea unui tabel cu verificarea existenței sale

`DROP TABLE IF EXISTS nume_tabel;`

- Exemplu

