



Tema 23.

Python și MySQL



Ce ne așteaptă?

1. Stabilirea conexiunii cu serverul MySQL
2. Crearea, modificarea și ștergerea tabelelor
3. Înscrierea datelor în tabele
4. Citirea datelor din baza de date
5. Actualizarea și ștergerea datelor

1. Cum se creează o conexiune la serverul MySQL?
2. Cum se creează o bază de data pe server?
3. Cum se conectează la o bază de date existentă?
4. Cum se creează un tabel în baza de date MySQL?
5. Cum se modifică setările unui tabel?
6. Cum se șterge un tabel din baza de date?
7. Cum se înscriu date într-un tabel al bazei de date MySQL?
8. Cum se citesc datele dintr-un tabel al bazei de date?
9. Cum se actualizează datele dintr-un tabel?
10. Cum se șterg datele dintr-un tabel al bazei de date?

1. Stabilirea conexiunii cu serverul MySQL

Instalarea componentelor necesare

- Mediul MySQL este bazat pe sistemul client server ce necesită existența serverului MySQL, conectorilor MySQL/Python și aplicația Python în calitate de client
- Conectorii MySQL/Python pot fi instalați în timpul instalării serverului MySQL
- Pentru elaborarea aplicației Python este necesară instalarea interpretorului Python dar și a unui mediu de lucru (de exemplu PyCharm)
- Pentru lucru cu baza de date MySQL în Python trebuie instalat modul `mysql-connector-python`

```
pip install mysql-connector-python
```

- Pentru lucru cu baza de date MySQL în Python trebuie instalat modul După instalarea modului acesta se importă cu numele `mysql.connector`

```
import mysql.connector
```

Funcția connect()

- Pentru crearea conexiunii se utilizează funcția `connect()` a modului `mysql.connector`
- Funcția `connect()` primește ca parametri:
 - `host` – adresa host de pe care se apelează serverul MySQL
 - `user` – numele utilizatorului bazei de date
 - `password` – parola de conectare la baza de date
 - `database` – baza de date dacă există
- Pentru introducerea parolei se utilizează metoda `getpass()` a modulul `getpass` ce asigură securitatea acesteia
- În PyCharm pentru utilizarea modului `getpass` este necesară emularea terminalului pentru acesta se va bifa opțiunea `Emulate terminal in ouput console` din meniul `Run->Edit Configurations`
- Funcția `connect()` creează un obiect al conexiunii care pentru închiderea automată la finalul sesiunii se include într-un bloc `with`
- Pentru captarea erorilor de conexiune se utilizează un bloc `try-except`

Exemplu de stabilirea a conexiunii

- **Codul Python de stabilirea a unei conexiuni**

```
from getpass import getpass
from mysql.connector import connect, Error
try:
    with connect(
        host="localhost",
        user=input("Enter username: "),
        password=getpass("Enter password: "),
    ) as connection:
        print(connection)
except Error as e:
    print(e)
```

- **Rezultatul execuției codului Python de stabilirea a unei conexiuni**

```
Enter username: root
Enter password:
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001698770CFA0>
```

Crearea unei baze de date

- **Interogarea SQL**

```
CREATE DATABASE nume_baza;
```

- Pentru executarea interogărilor SQL se creează un obiect cursor permite obiectului conexiunii să interacționeze cu serverul
- Cursorul se creează cu `cursor()` a obiectului `connection`

```
cursor = connection.cursor()
```

- Ca și în cazul conexiunii, cursorul trebuie închis după terminarea sesiunii de aceea se va include într-un bloc `with` pentru închiderea automată
- Interogarea se transmite ca parametru string a metodei `execute()` a cursorului

```
interogare_crearea_bd = "CREATE DATABASE nume_baza "  
cursor.execute(interogare_crearea_bd)
```

- În SQL interogarea se finisează cu simbolul `;` care nu trebuie specificat în Python deoarece este adăugat automat

Exemplu de crearea a bazei de date

- **Codul Python de creare a bazei de date**

```
from getpass import getpass
from mysql.connector import connect, Error
try:
    with connect(host="localhost", user=input("Enter username: "),
                password=getpass("Enter password: "),
                ) as connection:
        interogare_crearea_bd = "CREATE DATABASE baza_de_test"
        with connection.cursor() as cursor:
            cursor.execute(interogare_crearea_bd)
            print('Baza de date a fost creata')
except Error as e:
    print(e)
```

- **Rezultatul execuției codului Python de crearea a bazei de date**

```
Enter username: root
Enter password:
Baza de date a fost creata
```


Vizualizarea tuturor bazelor de date

- Interogarea SQL

`SHOW DATABASES;`

- Pentru vizualizarea tuturor bazelor de date utilizează o buclă for ce va citi conținutul cursorului după executarea interogării respective
- Exemplu: codul Python scris în blocul conexiunii

```
interogare_vizual_db = "SHOW DATABASES"
with connection.cursor() as cursor:
    cursor.execute(interogare_vizual_db)
    for db in cursor:
        print(db)
```

- Rezultatul execuției codului Python

```
Enter username: root
Enter password:
('baza_date_1',)
('baza_de_test',)
('information_schema',)
('magazin_online',)
('mysql',)
('performance_schema',)
('sakila',)
('sys',)
```

Conectarea la o bază de date existentă

- Pentru conectarea la o bază de date existentă în funcția `connect()` se va trece parametrul `database` cu numele bazei de date
- Exemplu: Conectarea la baza de date – `magazin_online`

```
from getpass import getpass
from mysql.connector import connect, Error
try:
    with connect(host="localhost", user=input("Enter username: "),
                password=getpass("Enter password: "),
                database="magazin_online"
    ) as connection:
        print('Conexiunea la baza de date a reusit')
except Error as e:
    print(e)
```

- Rezultatul execuției codului Python

```
Enter username: root
Enter password:
Conexiunea la baza de date a reusit
```

2. Crearea, modificare și ștergerea tabelor

Crearea unui tabel

- Pentru crearea unui tabel cu ajutorul Python se execută metoda `execute()` a cursorului, iar în calitate de parametru se trece interogarea SQL corespunzătoare sub formă de string
- În mod implicit, conectorii MySQL nu confirmă tranzacțiile de aceea pentru confirma interogările este necesară utilizarea metodei `commit()` a conexiunii.
- Sintaxa creării unui tabel în cadrul blocului conexiunii:

```
interogare_crearea_tabel= interogare_SQL_string  
with connection.cursor() as cursor:  
    cursor.execute(interogare_crearea_tabel)  
    connection.commit()
```

Exemplu de crearea a tabelor (1)

- **Declararea a 3 stringuri cu interogările a 3 tabele**

```
interog_creare_clienti = '''CREATE TABLE clienti (  
    id_client INT NOT NULL AUTO_INCREMENT,  
    nume VARCHAR(45) NOT NULL,  
    prenume VARCHAR(45) NOT NULL,  
    email VARCHAR(45),  
    data_nasterii DATE NOT NULL,  
    PRIMARY KEY (id_client))'''  
  
interog_creare_produce = ''' CREATE TABLE produse (  
    id_produș INT NOT NULL AUTO_INCREMENT,  
    nume VARCHAR(45) NOT NULL,  
    pret DECIMAL(10,2) NOT NULL,  
    PRIMARY KEY (id_produș))'''  
  
interog_creare_vanzari = ''' CREATE TABLE vanzari (  
    id_vanzare INT NOT NULL AUTO_INCREMENT,  
    id_client INT NOT NULL,  
    id_produș INT NOT NULL,  
    data_vanzarii DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    numar_produce INT NOT NULL,  
    PRIMARY KEY (id_vanzare),  
    CONSTRAINT fk_id_client FOREIGN KEY (id_client) REFERENCES clienti (id_client)  
    ON DELETE NO ACTION ON UPDATE CASCADE,  
    CONSTRAINT fk_id_produș FOREIGN KEY (id_produș) REFERENCES produse (id_produș)  
    ON DELETE NO ACTION ON UPDATE CASCADE)'''
```

Exemplu de crearea a tabelelor (2)

- **Codul Python de creare a 3 tabele conform stringurilor cu interogări**

```
try:
    with connect(
        host="localhost",
        user=input("Enter username: "),
        password=getpass("Enter password: "),
        database="baza_de_test",
    ) as connection:
        with connection.cursor() as cursor:
            cursor.execute(interog_creare_clienti)
            cursor.execute(interog_creare_produce)
            cursor.execute(interog_creare_vanzari)
            connection.commit()
        print('Tabelele au fost create cu succes')
except Error as e:
    print(e)
```

- **Rezultatul execuției codului Python**

```
Enter username: root
Enter password:
Tabelele au fost create cu succes
```

Vizualizarea setărilor tabelului

- **Interogarea SQL**

```
DESCRIBE nume_tabel;
```

- **Pentru extragerea rezultatelor obiectului cursor se utilizează metoda fetchall()**

- **Exemplu: vizualizarea setărilor tabelului produse**

```
interog_vizual_tabel = "DESCRIBE produse"  
with connection.cursor() as cursor:  
    cursor.execute(interog_vizual_tabel)  
    rezultat = cursor.fetchall()  
    for linie in rezultat:  
        print(linie)
```

- **Rezultatul execuției codului Python**

```
Enter username: root  
Enter password:  
(('id_produș', b'int', 'NO', 'PRI', None, 'auto_increment')  
(('nume', b'varchar(45)', 'NO', '', None, '')  
(('pret', b'decimal(10,2)', 'NO', '', None, ''))
```

Modificarea setărilor unui tabel

- **Interogarea SQL de adăugare a unei coloane**

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

- **Exemplu: adăugarea a unei coloane în tabelul produse și vizualizarea setărilor**

```
interog_modif_tabel = "ALTER TABLE produse ADD COLUMN descriere VARCHAR(500);"  
interog_vizual_tabel = "DESCRIBE produse"  
with connection.cursor() as cursor:  
    cursor.execute(interog_modif_tabel)  
    cursor.execute(interog_vizual_tabel)  
    rezultat = cursor.fetchall()  
    for linie in rezultat:  
        print(linie)
```

- **Rezultatul execuției codului Python**

```
('id_produc', b'int', 'NO', 'PRI', None, 'auto_increment')  
('nume', b'varchar(45)', 'NO', '', None, '')  
('pret', b'decimal(10,2)', 'NO', '', None, '')  
('descriere', b'varchar(500)', 'YES', '', None, '')
```

Ștergerea unui tabel

- **Interogarea SQL**

```
DROP TABLE nume_tabel;
```

- **Exemplu: codul Python de ștergere a tabelului client (necesită mai întâi să fie creat)**

```
interog_stergere_tabel = „DROP TABLE client”  
with connection.cursor() as cursor:  
    cursor.execute(interog_stergere_tabel)  
    print('Tabelul a fost sters cu succes')
```

- **Rezultatul execuției codului Python**

```
Tabelul a fost sters cu succes
```

```
Process finished with exit code 0
```


3. Înscrierea datelor în tabele

Metoda execute()

- Interogarea SQL pentru înscrierea datelor

```
INSERT INTO nume_tabel (nume_col_1, nume_col_2, nume_col_3, nume_col_4)
VALUES (valoarea_col_1, valoarea_col_2, valoarea_col_3, valoarea_col_4);
```

- Pentru înscrierea datelor se utilizează metoda execute() pe obiectul cursorului și se trece ca parametru interogarea SQL sub formă de string
- Pentru validarea operație este necesară confirmarea tranzacție prin metoda commit()

```
cursor.execute(interogare_insert)
connection.commit()
```

- Metoda execute() se utilizează în cazul înscrierii unui volum mic de date sau în cazul introducerii manuale a datelor

Exemplu de înscriere a datelor prin metoda execute()

- Înscrierea datelor in tabelul clienti prin introducerea acestora de la consola
- Codul Python inclus în blocul conexiunii

```
nume = input('Introdu numele clientului: ')
prenume = input('Introdu prenumele clientului: ')
email = input('Introdu adresa de email a clientului: ')
data_nasterii = input('Introdu data nasterii clientului: ')
adresa = input('Introdu adresa clientului: ')
interog_inscriere_date = f"""INSERT INTO clienti (nume, prenume, email, adresa, data_nasterii)
                            VALUES('{nume}', '{prenume}', '{email}', '{adresa}', '{data_nasterii}')"""
with connection.cursor() as cursor:
    cursor.execute(interog_inscriere_date)
    connection.commit()
print('Datele au fost inscise cu succes')
```

- Rezultatul execuției codului Python

```
Introdu numele clientului: Petrovici
Introdu prenumele clientului: Anatol
Introdu adresa de email a clientului: petrovici@gmail.com
Introdu data nasterii clientului: 1997-07-23
Introdu adresa clientului: Suceava
Datele au fost inscise cu succes
```

Metoda executemany()

- Metoda `executemany()` se utilizează în cazul înscrierii mai multor date în tabel în special date citite dintr-un fișier
- Metoda `executemany()` necesită 2 parametri:
 - Stringul cu interogarea INSERT ce conține simbolul `%s` în locul valorilor datelor
 - O listă din cu tupluri, unde fiecare tuplu contine din valorile unei înscrieri în tabel
- Sintaxa metodei `executemany()`:

```
cursor.executemany(interogare_insert, lista_tupluri)  
connection.commit()
```

Exemplu de înscriere a datelor prin metoda executemany()

- Înscrierea mai multor in tabelul produse prin introducerea listei cu valori
- Codul Python inclus în blocul conexiunii

```
interog_inscriere_date = f"""INSERT INTO produse (nume, pret, descriere) VALUES (%s, %s, %s) """
lista_tupluri = [
    ('tricou', 24.50, 'Pentru vara'),
    ('sapca', 9.99, 'Pentru calatorie'),
    ('pantaloni', 32.75, 'Pentru vreme friguroasa'),
    ('camasa', 27.25, 'Pentru serviciu')
]
with connection.cursor() as cursor:
    cursor.executemany(interog_inscriere_date, lista_tupluri )
    connection.commit()
print('Datele au fost inscise cu succes')
```

- Rezultatul execuției codului Python

```
Datele au fost inscise cu succes
```

4. Citirea datelor din tabele

Esența citirii datelor

- Pentru citirea datelor din tabel se utilizează metoda `execute()` pe obiectul cursorului și se trece ca parametru interogarea INSERT sub formă de string
- Pentru extragerea rezultatul din obiectul cursorului se utilizează metodele:
 - `fetchall()` - extrage toate datele într-o lista ce include mai multe tupluri, fiecare tuplu conținând datele unei linii
 - `fetchone()` - extrage datele primei linii a tabelului sub forma de tuplu
 - `fetchmany(n)` - extrage primele n de linii a tabelului
- Stringul interogării INSERT poate include și alte declarații: **FROM, LIMIT, ORDER BY, WHERE, GROUP BY, DISTINCT, JOIN, etc**
- Întrucât datele se conțin în cursor, pentru a putea înscrie noi date este necesara utilizarea unui bufer deci cursorul se creează astfel

```
cursor = connection.cursor(bufferd=True)
```

Exemplu de citirea a datelor tuturor liniilor

- Citirea din tabelul produse a tuturor produselor a căror preț depășește 20
- Codul Python inclus în blocul conexiunii

```
interog_select_date = "SELECT * FROM produse WHERE pret>20"  
with connection.cursor(buffered=True) as cursor:  
    cursor.execute(interog_select_date)  
    rezultat = cursor.fetchall()  
    for linie in rezultat:  
        print(linie)
```

- Rezultatul execuției codului Python

```
(1, 'tricou', Decimal('24.50'), 'Pentru vara')  
(3, 'pantaloni', Decimal('32.75'), 'Pentru vereme friguroasa')  
(4, 'camasa', Decimal('27.25'), 'Pentru serviciu')  
(5, 'tricou', Decimal('24.50'), 'Pentru vara')  
(7, 'pantaloni', Decimal('32.75'), 'Pentru vreme friguroasa')  
(8, 'camasa', Decimal('27.25'), 'Pentru serviciu')
```

Exemplu de citirea a datelor primei linii

- Citirea din tabelul produse a primului produs ce în descriere are cuvântul serviciu
- Codul Python inclus în blocul conexiunii

```
interog_select_date = "SELECT * FROM produse WHERE descriere LIKE '%serviciu%'"
with connection.cursor(buffered=True) as cursor:
    cursor.execute(interog_select_date)
    rezultat = cursor.fetchone()
    print(rezultat)
```

- Rezultatul execuției codului Python

```
(4, 'camasa', Decimal('27.25'), 'Pentru serviciu')
```

```
Process finished with exit code 0
```

Exemplu de citirea a datelor unui număr de linii

- Citirea din tabelul produse a primelor 3 produse cu prețul mai mic de 30
- Codul Python inclus în blocul conexiunii

```
interog_select_date = "SELECT * FROM produse WHERE pret<30"  
with connection.cursor(buffered=True) as cursor:  
    cursor.execute(interog_select_date)  
    rezultat = cursor.fetchmany(3)  
    for linie in rezultat:  
        print(linie)
```

- Rezultatul execuției codului Python

```
(1, 'tricou', Decimal('24.50'), 'Pentru vara')  
(2, 'sapca', Decimal('9.99'), 'Pentru calatorie')  
(4, 'camasa', Decimal('27.25'), 'Pentru serviciu')
```


5. Actualizarea și ștergerea datelor

Actualizarea datelor

- **Interogarea SQL**

```
UPDATE nume_tabel SET nume_coloana=valoarea_nouă  
WHERE nume_coloana_cheie_primara=valoarea_cheie_primara;
```

- **Pentru actualizarea datelor se utilizează metoda execute() pe obiectul cursorului și se trece ca parametru interogarea SQL UPDATE sub formă de string**
- **Pentru validarea operație este necesară confirmarea tranzacție prin metoda commit()**

```
cursor.execute(interogare_update)  
connection.commit()
```

Exemplu de actualizarea a datelor

- Actualizarea preturilor produselor din tabelul produse prin specificarea prețului vechi și a celui nou de la consola
- Codul Python inclus în blocul conexiunii

```
pret_vechi = input('Introduceti valoarea pretului vechi: ')
pret_nou = input('Introduceti valoarea pretului nou: ')
interog_modific_date = f"""UPDATE produse SET pret={pret_nou}
                           WHERE pret={pret_vechi}"""
with connection.cursor() as cursor:
    cursor.execute(interog_modific_date)
    connection.commit()
    print('Actualizarile s-au realizat cu succes')
```

- Rezultatul execuției codului Python

```
Introduceti valoarea pretului vechi: 9.99
Introduceti valoarea pretului nou: 11.00
Actualizarile s-au realizat cu succes
```

Interogări multiple

- Metoda `execute()` acceptă ca parametru și stringuri ce includ mai multe interogări SQL
- Pentru a accepta stringuri cu mai multe interogări metodei `execute` în se mai trece un parametru `multi=True`
- În cazul interogărilor multiple metoda `execute()` returnează un iterator a cărui elemente reprezintă cursoare pentru executarea fiecărei interogări
- În cazul interogărilor multiple, unele interogări nu transmit date cursorului (`INSERT`, `UPDATE`, `DELETE`), iar altele pot să transmită date cursorului (`SELECT`) de aceea pentru a verifica dacă cursorul conține date se utilizează proprietatea `with_rows` asupra conținutului cursorului

```
interogare_multipla = 'interogare 1; interogare 2'
rezultat = cursor.execute(interogare_multipla, multi=True)
for i in rezultat:
    if i.with_rows:
        print(i)
connection.commit()
```

Exemplu interogări multiple

- Actualizarea preturilor produselor din tabelul produse prin specificarea prețului vechi și a celui nou de la consola și vizualizarea acestor produse
- Codul Python inclus în blocul conexiunii

```
pret_vechi = input('Introduceti valoarea pretului vechi: ')
pret_nou = input('Introduceti valoarea pretului nou: ')
interog_multipla_date = f"""UPDATE produse SET pret={pret_nou}
                             WHERE pret={pret_vechi};
                             SELECT * FROM produse
                             WHERE pret={pret_nou}"""

with connection.cursor() as cursor:
    rezultat = cursor.execute(interog_multipla_date, multi=True)
    for linie in rezultat:
        if linie.with_rows:
            print(linie.fetchall())
    connection.commit()
```

- Rezultatul execuției codului Python

```
Introduceti valoarea pretului vechi: 11
Introduceti valoarea pretului nou: 12
[(2, 'sapca', Decimal('12.00'), 'Pentru calatorie'), (6, 'sapca', Decimal('12.00'), 'Pentru calatorie')]
```

Ștergerea datelor

- **Interogarea SQL**

```
DELETE FROM nume_tabel WHERE nume_col=valoarea_col;
```

- **Pentru actualizarea datelor se utilizează metoda execute() pe obiectul cursorului și se trece ca parametru interogarea SQL DELETE sub formă de string**
- **Pentru validarea operație este necesară confirmarea tranzacție prin metoda commit()**

```
cursor.execute(interogare_delete)  
connection.commit()
```

Exemplu de ștergere a datelor

- Ștergerea datelor care au id-ul in intervalul 5-8
- Codul Python inclus în blocul conexiunii

```
interog_stergere_date = "DELETE FROM produse WHERE id_produc BETWEEN 5 AND 8"  
with connection.cursor() as cursor:  
    cursor.execute(interog_stergere_date)  
    connection.commit()  
print('Datele au fost sterse cu succes')
```

- Rezultatul execuției codului Python

Datele au fost sterse cu succes

Process finished with exit code 0