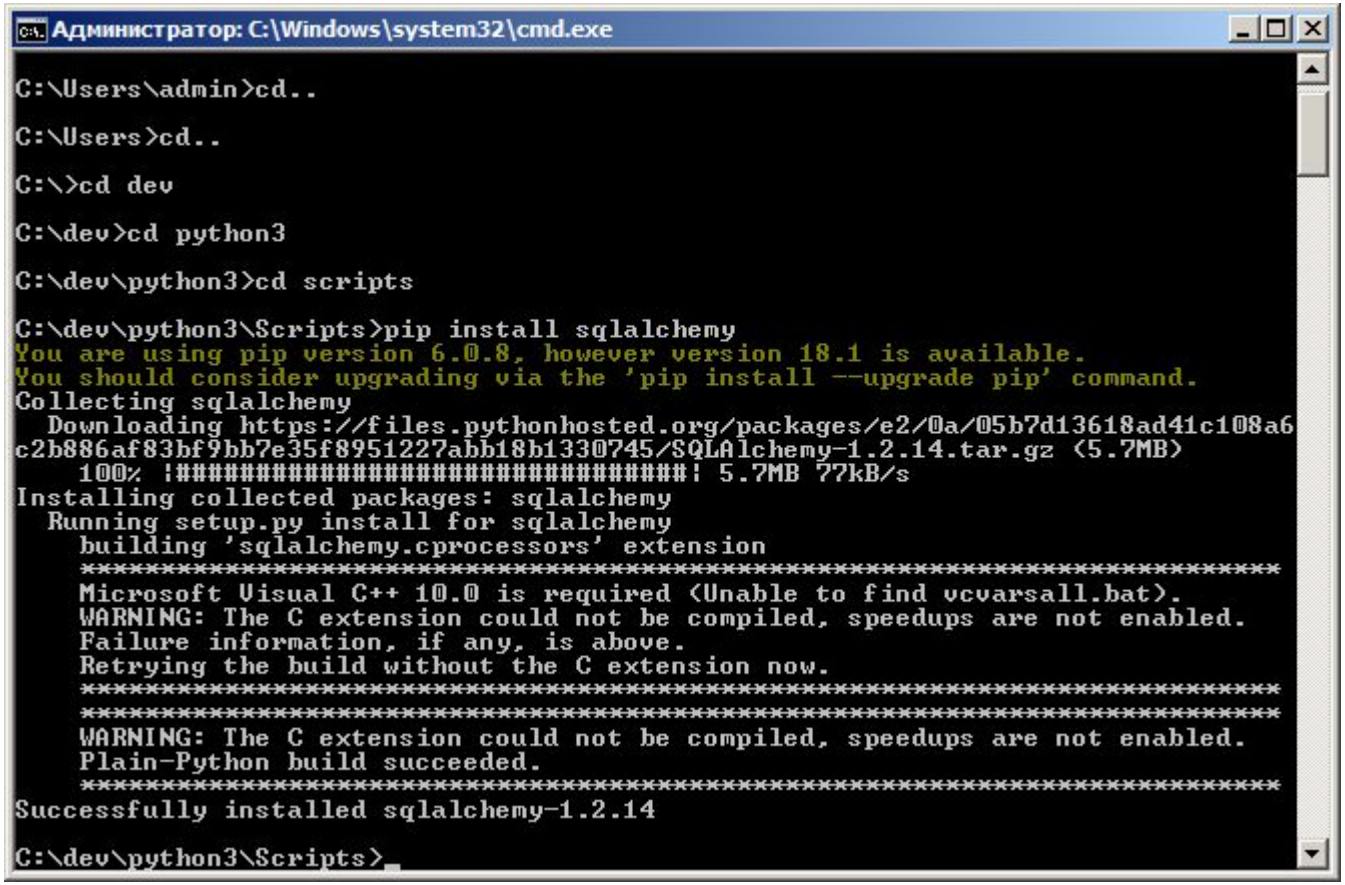
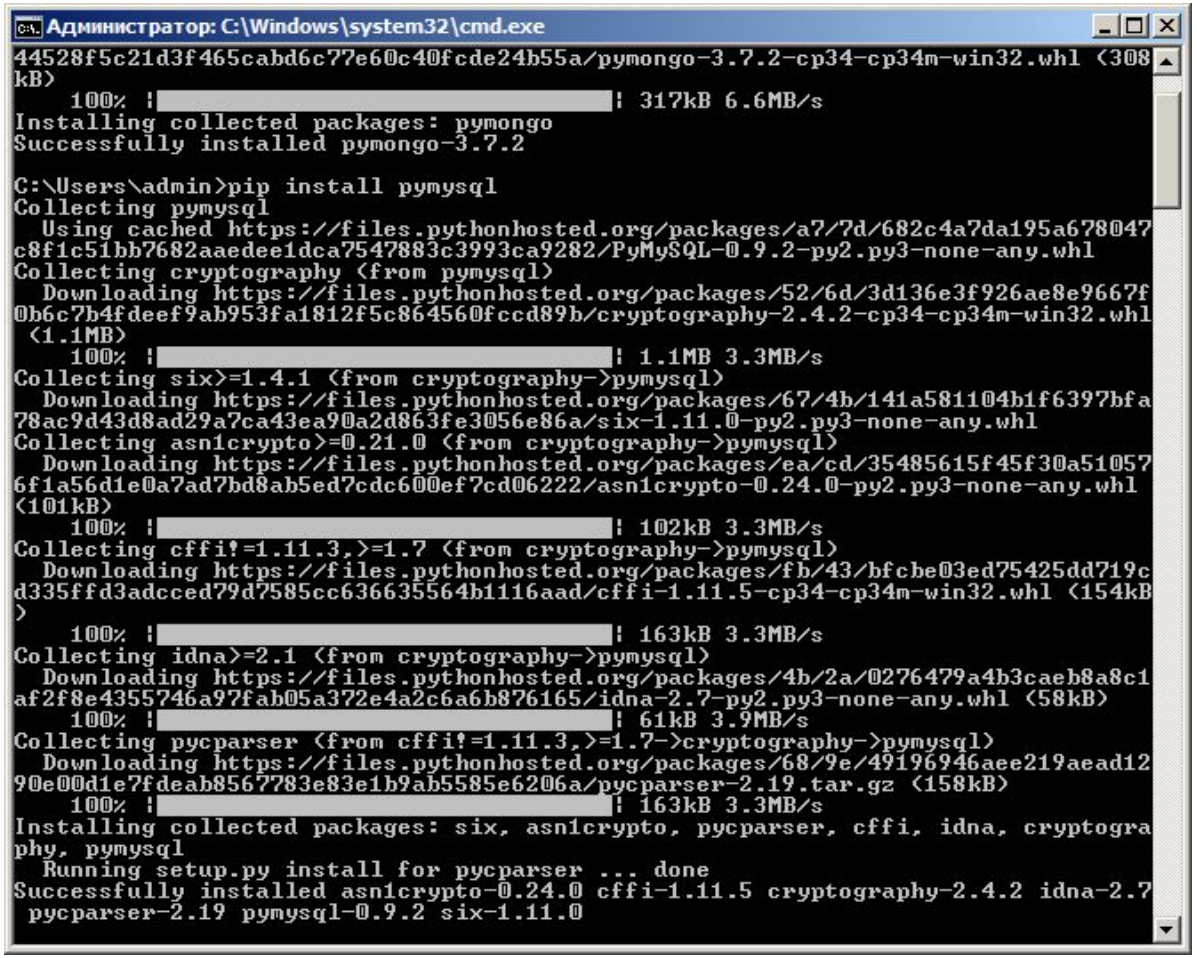
Python

SQL 2

pip install sqlalchemy



pip install pymysql



Соединение с БД

mysql002.py | log.txt

```
1  import string
2  import pymysql as MySQLdb
3  #import MySQLdb
4
5  host='localhost'
6  user='root'
7  password='12345'
8  db='test'
9  charset='utf8'
10
```

```
db=MySQLdb.connect(host=host,user=user,password=password,db=db,charset=charset)
cursor=db.cursor()
```

```
cursor.close()
db.close()
```

Проверка версий

```
db=MySQLdb.connect(host=host,user=user,password=password,db=db,charset=charset)
cursor=db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print ("Database version : %s " % data)

cursor.close()
db.close()
```

```
C:\Python34>python mysql004.py
Database version : 5.6.42-log
```

Создание таблиц

```
# Drop table if it already exist using execute() method.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement
sql = """CREATE TABLE EMPLOYEE (
   FIRST_NAME  CHAR(20) NOT NULL,
   LAST_NAME  CHAR(20),
   AGE INT,
   SEX CHAR(1),
   INCOME FLOAT )"""

cursor.execute(sql)
```

Вставка данных

```
# Prepare SQL query to INSERT a record into the database.
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
   LAST_NAME, AGE, SEX, INCOME)
   VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
```

Ввод пароля в БД (плохой способ)

- user_id = "test123 "
- password = "password"
- con.execute('insert into Login values("%s", "%s")' % (user_id, password))

```python
def unpack_line(line):
    line = line.replace("'", "")
    els = line.split(";")
    # выделяем имя, емейл, адрес и телефон
    fname = els[0]
    fmail = els[1]
    fadres = els[2]
    ftel = els[3]
    return fname, fmail, fadres, ftel
```

```python
for line in lines:
    # если в строе присутствует емейл (определяем по наличию "@")
    if line.find("@") !=0:
        # извлекаем данные из строки
        fname, fmail, fadres, ftel = unpack_line(line)
        # подставляем эти данные в SQL-запрос
        sql = """INSERT INTO contacts(name, mail, adres, tel)
        VALUES ('%(name)s', '%(mail)s', '%(adres)s', '%(tel)s')
        """%{"name":fname, "mail":fmail, "adres":fadres, "tel":ftel}
        # исполняем SQL-запрос
        cursor.execute(sql)
        # применяем изменения к базе данных
        db.commit()
```

Чтение из БД

- **fetchone()** – выбор одной строки
- **fetchall()** – выбор всех строк
- **rowcount** – число строк к которым был применен execute

```
# Prepare SQL query to INSERT a record into the database.
sql = "SELECT * FROM EMPLOYEE \
      WHERE INCOME > '%d'" % (1000)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Fetch all the rows in a list of lists.
   results = cursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
      # Now print fetched result
      print ("fname = %s,lname = %s,age = %d,sex = %s,income = %d" % \
         (fname, lname, age, sex, income ))
except:
   print ("Error: unable to fetch data")
```

```
C:\Python34>python mysql008.py
fname = Mac,lname = Mohan,age = 20,sex = M,income = 2000
fname = Mac,lname = Mohan,age = 20,sex = M,income = 2000
```

Обновление

```
# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = '%c'" % ('M')
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
```

Удаление

```
## Prepare SQL query to DELETE required records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
```

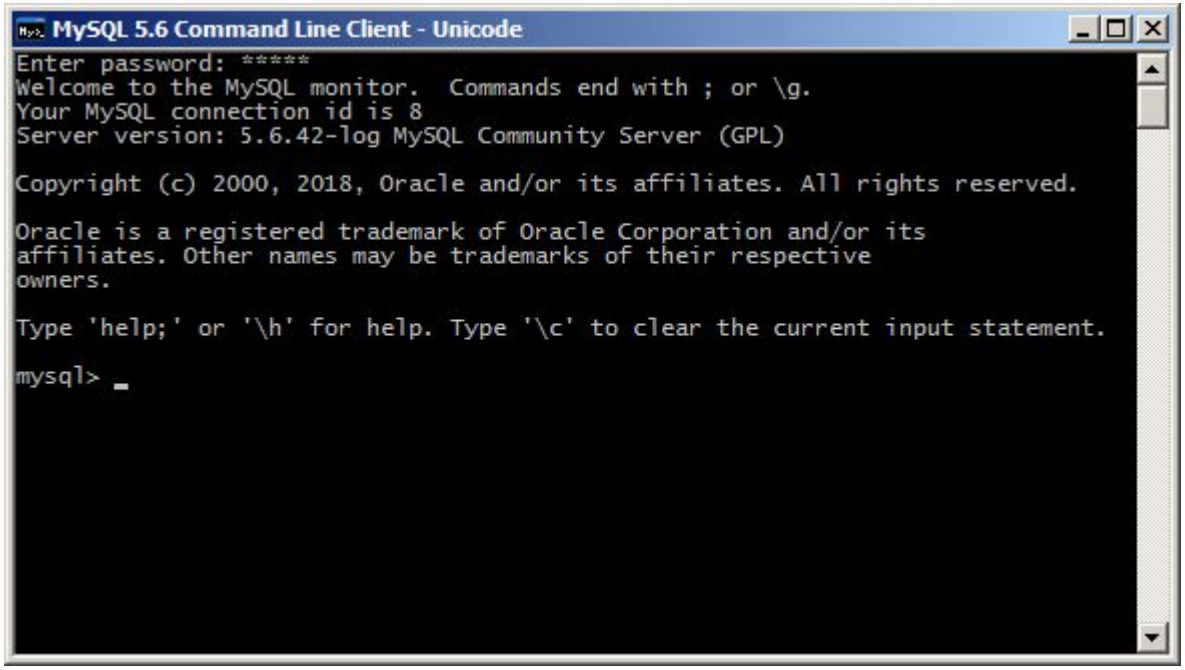
Атрибуты трансляций

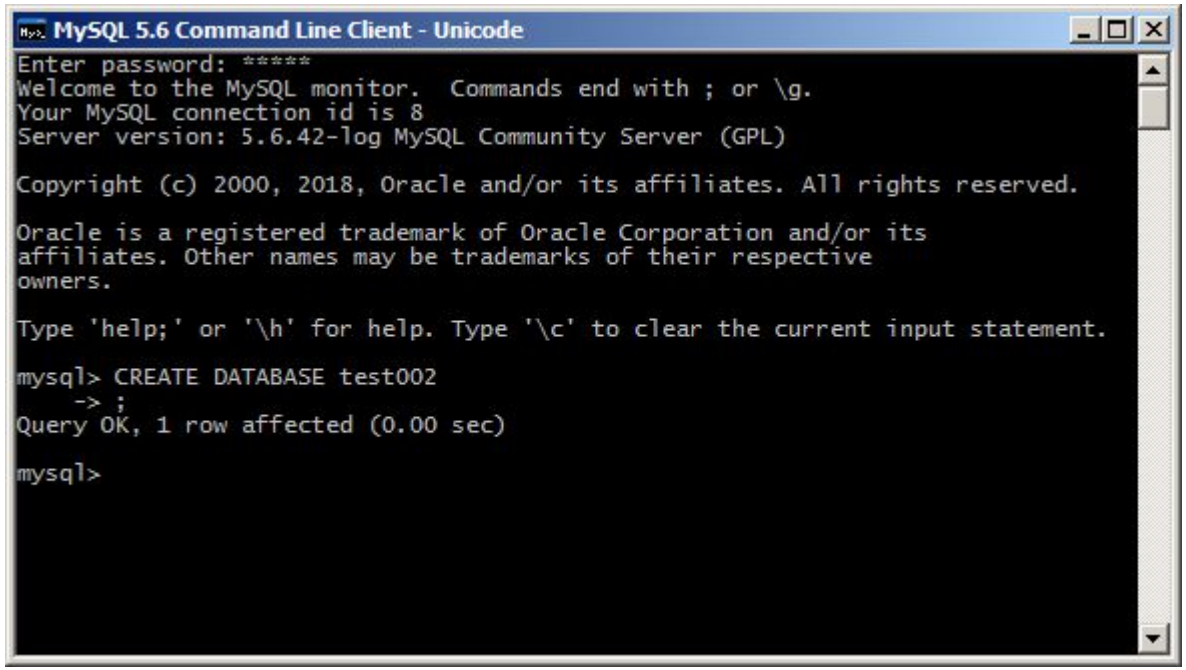
- **Atomicity** - завершение трансляции, в противном случае ничего не происходит
- **Consistency** – должна начинаться и заканчиваться в согласованном состоянии
- **Isolation** - промежуточные результаты трансляции видны за пределами трансляции
- **Durability** - данные сохраняются даже в случае сбоя системы

```
# Prepare SQL query to DELETE required records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
```

- +примеры страницы 100

pip install mysql-connector-python==2.18


MySQL 5.6 Command Line Client - Unicode
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.6.42-log MySQL Community Server (GPL)
Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>


MySQL 5.6 Command Line Client - Unicode
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.6.42-log MySQL Community Server (GPL)
Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> CREATE DATABASE test002
    -> ;
Query OK, 1 row affected (0.00 sec)
mysql>

SCHEMAS
Filter objects
sakila
test
test001
test002
test_con
world

Соединение с БД с connector

```
import mysql.connector
user='root'
password='12345'
host='127.0.0.1'
database='test002'
cnx = mysql.connector.connect(database=database,user=user,password=password,host=host)
cnx.close
```

Использование класса connection

```
#import mysql.connector
from mysql.connector import (connection)
user='root'
password='12345'
host='127.0.0.1'
database='test002'
#cnx = mysql.connector.connect(database=database,user=user,password=password,host=host)
cnx = connection.MySQLConnection(database=database,user=user,password=password,host=host)
cnx.close
```

Использование конфигурации

```
import mysql.connector
config={
'user':'root',
'password':'12345',
'host':'127.0.0.1',
'database':'test002',}

cnx = mysql.connector.connect(**config)

cnx.close()
```

Проверка на ошибки

```python
import mysql.connector
#from mysql.connector import errorcode
from mysql.connector import Error

config={
'user':'root',
'password':'12345',
'host':'127.0.0.1',
'database':'test002',
}

def connect():
    try:
        conn=mysql.connector.connect(**config)
        if conn.is_connected():
            print('Ok')
    except Error as e:
        print(e)
    finally:
        conn.close()


connect()
```

```
C:\dev\python3>python mysql_con004.py
Ok
```

Использование файла конфигурации

```
[mysql]
user=root
password=12345
host=localhost
database=test002
```

```python
from configparser import ConfigParser

import mysql.connector
from mysql.connector import Error

def read_db_config(filename='config_sql.ini', section='mysql'):
    parser = ConfigParser()
    parser.read(filename)

    # get section, default to mysql
    db = {}
    if parser.has_section(section):
        items = parser.items(section)
        for item in items:
            db[item[0]] = item[1]
    else:
        raise Exception('{0} not found in the {1} file'.format(section, filename))

    return db


db=read_db_config()
print(db)
```

```
C:\dev\python3>python mysql_con005.py
{'host': 'localhost', 'user': 'root', 'database': 'test002', 'password': '12345'
}
```

```python
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def connect():
    db_config = read_db_config()
    try:
        print('Connecting to MySQL database...')
        conn = MySQLConnection(**db_config)
        if conn.is_connected():
            print('connection established.')
        else:
            print('connection failed.')

    except Error as error:
        print(error)

    finally:
        conn.close()
        print('Connection closed.')

if __name__ == '__main__':
    connect()
```

```
C:\dev\python3>python mysql_con006.py
Connecting to MySQL database...
connection established.
Connection closed.
```

fetchone()

```python
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def query_with_fetchone():
    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM books")

        row = cursor.fetchone()

        while row is not None:
            print(row)
            row = cursor.fetchone()

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    query_with_fetchone()
```

fetchall()

```
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def query_with_fetchall():
    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM books")
        rows = cursor.fetchall()

        print('Total Row(s):', cursor.rowcount)
        for row in rows:
            print(row)

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    query_with_fetchall()
```

fetchmany()

```python
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def iter_row(cursor, size=10):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

def query_with_fetchmany():
    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM books")

        for row in iter_row(cursor, 10):
            print(row)

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    query_with_fetchmany()
```

Вставка одной строки в таблицу

```python
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def insert_book(title, isbn):
    query = "INSERT INTO books(title,isbn) "
            "VALUES(%s,%s)"
    args = (title, isbn)

    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)

        cursor = conn.cursor()
        cursor.execute(query, args)

        if cursor.lastrowid:
            print('last insert id', cursor.lastrowid)
        else:
            print('last insert id not found')

        conn.commit()
    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()

def main():
   insert_book('A Sudden Light','9781439187036')

if __name__ == '__main__':
    main()
```

Вставка нескольких строк в таблицу

```
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

INSERT INTO books(title,isbn)
VALUES('Harry Potter And The Order Of The Phoenix', '9780439358071'),
      ('Gone with the Wind', '9780446675536'),
      ('Pride and Prejudice (Modern Library Classics)', '9780679783268');

def insert_books(books):
    query = "INSERT INTO books(title,isbn) "
            "VALUES(%s,%s)"

    try:
        conn = MySQLConnection(**db_config)

        cursor = conn.cursor()
        cursor.executemany(query, books)

        conn.commit()
    except Error as e:
        print('Error:', e)

    finally:
        cursor.close()
        conn.close()

def main():
    books = [('Harry Potter And The Order Of The Phoenix', '9780439358071'),
             ('Gone with the Wind', '9780446675536'),
             ('Pride and Prejudice (Modern Library Classics)', '9780679783268')]
    insert_books(books)

if __name__ == '__main__':
    main()
```

Обновление данных

```
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def update_book(book_id, title):
    # read database configuration
    db_config = read_db_config()

    # prepare query and data
    query = """ UPDATE books
                SET title = %s
                WHERE id = %s """

    data = (title, book_id)

    try:
        conn = MySQLConnection(**db_config)

        # update book title
        cursor = conn.cursor()
        cursor.execute(query, data)

        # accept the changes
        conn.commit()

    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    update_book(37, 'The Giant on the Hill *** TEST ***')
```

Удаление данных

```
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def delete_book(book_id):
    db_config = read_db_config()

    query = "DELETE FROM books WHERE id = %s"

    try:
        # connect to the database server
        conn = MySQLConnection(**db_config)

        # execute the query
        cursor = conn.cursor()
        cursor.execute(query, (book_id,))

        # accept the change
        conn.commit()

    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    delete_book(102)
```

Создание хранимых процедур MySQL

- для получения всех книг с информацией об авторе из таблиц **books** и **authors**:

```
DELIMITER $$

USE python_mysql$$

CREATE PROCEDURE find_all()
BEGIN
 SELECT title, isbn, CONCAT(first_name,' ',last_name) AS author
 FROM books
 INNER JOIN book_author ON book_author.book_id =  books.id
 INNER JOIN AUTHORS ON book_author.author_id = authors.id;
END$$

DELIMITER ;
```

find_all() содержит оператор **SELECT** с условием **JOIN**, который извлекает название, **ISBN**и полное имя автора из таблиц **books** и **authors**. Когда мы выполняем хранимую

```
DELIMITER $$
CREATE PROCEDURE find_by_isbn(IN p_isbn VARCHAR(13),OUT p_title VARCHAR(255))
    BEGIN
 SELECT title INTO p_title FROM books
 WHERE isbn = p_isbn;
    END$$

DELIMITER ;
```

find_by_isbn() принимает два параметра: первый параметр **ISBN** (параметр IN), второй — заголовок (OUT параметр). Когда вы передаете в хранимую процедуру **ISBN**

Вызов хранимых процедур

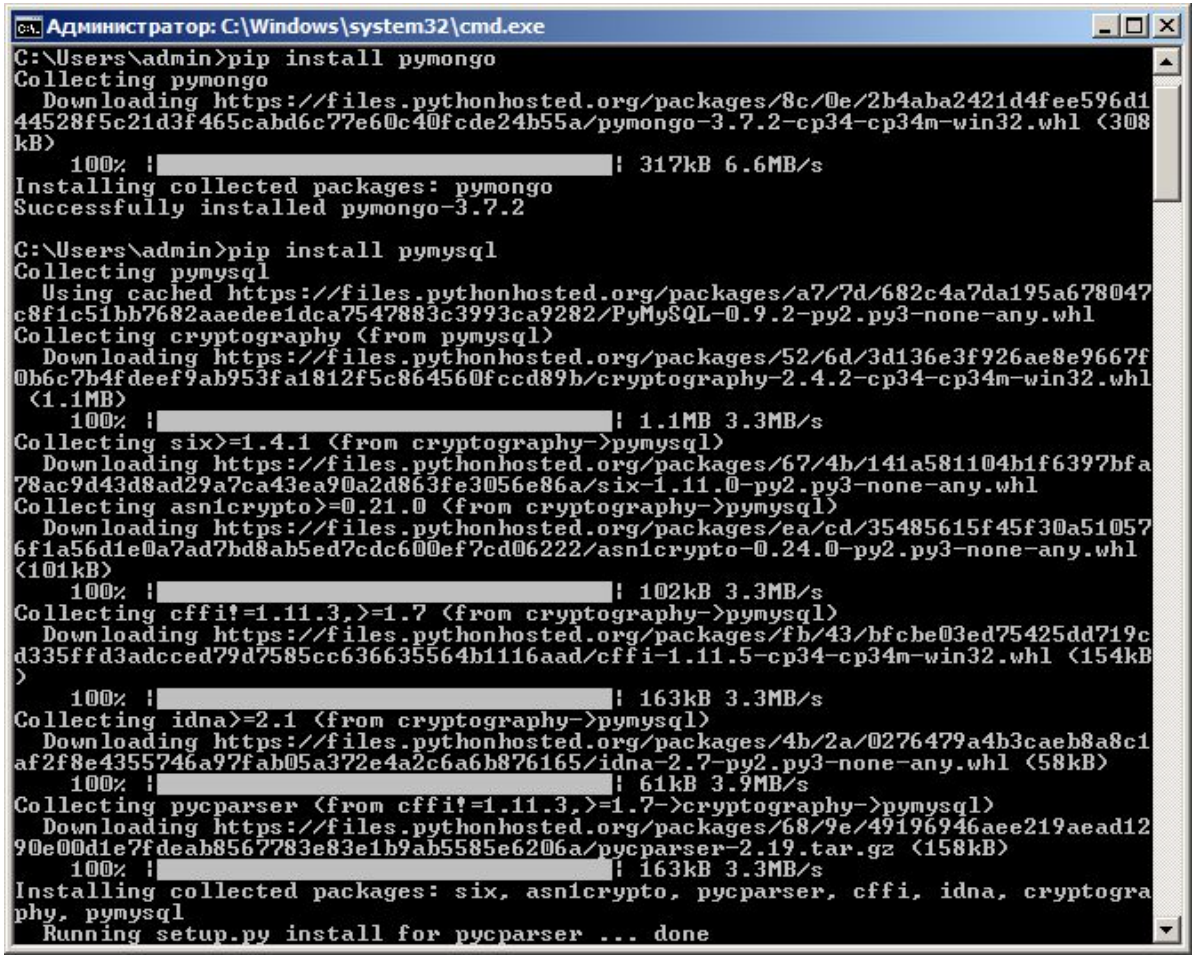
```
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def call_find_by_isbn():
    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)
        cursor = conn.cursor()

        args = ['1236400967773', 0]
        result_args = cursor.callproc('find_by_isbn', args)

        print(result_args[1])

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    call_find_by_isbn()
```

```python
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def call_find_all_sp():
    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)
        cursor = conn.cursor()

        cursor.callproc('find_all')

        # print out the result
        for result in cursor.stored_results():
            print(result.fetchall())

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    call_find_all_sp()
```

Обновление BLOB-данных

```
def read_file(filename):
    with open(filename, 'rb') as f:
        photo = f.read()
    return photo

def update_blob(author_id, filename):
    # read file
    data = read_file(filename)

    # prepare update query and data
    query = "UPDATE authors "
            "SET photo = %s "
            "WHERE id  = %s"

    args = (data, author_id)

    db_config = read_db_config()

    try:
        conn = MySQLConnection(**db_config)
        cursor = conn.cursor()
        cursor.execute(query, args)
        conn.commit()
    except Error as e:
        print(e)
    finally:
        cursor.close()
        conn.close()

def main():
    update_blob(144, "picturesgarth_stein.jpg")

if __name__ == '__main__':
    main()
```

Чтение BLOB данных

```python
from mysql.connector import MySQLConnection, Error
from mysql_con005 import read_db_config

def write_file(data, filename):
    with open(filename, 'wb') as f:
        f.write(data)

def read_blob(author_id, filename):
    # select photo column of a specific author
    query = "SELECT photo FROM authors WHERE id = %s"

    # read database configuration
    db_config = read_db_config()

    try:
        # query blob data form the authors table
        conn = MySQLConnection(**db_config)
        cursor = conn.cursor()
        cursor.execute(query, (author_id,))
        photo = cursor.fetchone()[0]

        # write blob data into a file
        write_file(photo, filename)

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

def main():
    read_blob(144,"outputgarth_stein.jpg")

if __name__ == '__main__':
    main()
```

pip install pymongo