# Работа с базами данных **PostgreSQL**

# Подключение к базе данных

**Установка модуля по работе с PostgreSQL:**

pip install psycopg2

**Подключение модуля:**

import psycopg2

Подключение к серверу PostgreSQL

**Подключение к серверу PostgreSQL:**

Для подключения к серверу PostgreSQL применяется функция connect().

**Функция принимает следующие параметры:**

1. dbname: имя базы данных
2. user: имя пользователя
3. password: пароль пользователя
4. host: хост/адрес сервера
5. port: порт (если не указано, то используется порт по умолчанию - 5432)

psycopg2.connect(dbname="db_name", host="db_host", user="db_user", password="db_pass", port="db_port")

# Функции по работе с базой данных

**Класс connection предоставляет ряд методов для работы с подключением к БД:**

1. close(): закрывает подключение
2. cursor(): возвращает объект cursor для осуществления запросов к бд
3. commit(): поддверждает транзакцию
4. rollback(): откатывает транзакцию

**Курсор и операции с данными:**

**1. execute(query, vars=None):** выполняет одну SQL-инструкцию (набор параметров)

```
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('John Doe', 30))
```

**executemany(query, vars_list):** выполняет параметризованное SQL-инструкцию. (списки из множества строк)

```
user_list = [('Jane Doe', 25), ('Bob Smith', 35)]
cursor.executemany("INSERT INTO users (name, age) VALUES (?, ?)", user_list)
```

**mogrify(operation[, parameters]): возвращает код запроса SQL после привязки параметров**

```
operation = "INSERT INTO users (name, age) VALUES (%s, %s)"
parameters = ('Alice', 28)
sql = cursor.mogrify(operation, parameters)
```

*b"INSERT INTO people (name, age) VALUES ('Alice', 28)“*

**fetchone(): возвращает следующую строку из полученного из БД набора строк в виде кортежа.**

```
row1 = cursor.fetchone()
```

**fetchmany([size=cursor.arraysize]): возвращает набор строк в виде списка. количество возвращаемых строк передается через параметр.**

```
rows2 = cursor.fetchmany(size=2)
```

**fetchall(): возвращает все (оставшиеся) строки в виде списка.**

```
remaining_rows = cursor.fetchall()
```

# Модель выполнения запросов

**Метод Commit:**

```
import psycopg2

conn = psycopg2.connect(dbname="postgres",
                        user="postgres",
                        password="123456",
                        host="127.0.0.1")
cursor = conn.cursor()

cursor.execute(sql1)

conn.commit()   # реальное выполнение команд sql1

cursor.close()
conn.close()
```

**Метод Autocommit:**

```
import psycopg2

conn = psycopg2.connect(dbname="postgres",
                        user="postgres",
                        password="123456",
                        host="127.0.0.1")

conn.autocommit = True  # устанавливаем актокоммит

cursor = conn.cursor()
cursor.execute(sql1)    # непосредственное
выполнение команды sql1

cursor.close()
conn.close()
```

# Примеры

**Пример Создание таблицы:**

```
import psycopg2

# Устанавливаем соединение с базой данных
conn = psycopg2.connect(
    database='mydatabase',
    user='myuser',
    password='mypassword',
    host='localhost',
    port='5432'
)

# Создаем курсор для выполнения SQL-запросов
cursor = conn.cursor()

# Создаем таблицу
cursor.execute('''CREATE TABLE IF NOT EXISTS users
            (id SERIAL PRIMARY KEY,
            value TEXT)''')

conn.commit()
```

**Пример Добавление данных и вывод данных:**

```
import sqlite3

conn = sqlite3.connect('example.db')
cursor = conn.cursor()

# Вставляем несколько записей
user_list = [('Jane Doe', 25), ('Bob Smith', 35)]
cursor.executemany("INSERT INTO users (name, age) VALUES (?, ?)", user_list)
cursor.execute("SELECT * FROM users")
# fetchone()
row1 = cursor.fetchone()
print("Первая строка:", row1)

# fetchmany()
rows2 = cursor.fetchmany(size=2)
print("Следующие две строки:", rows2)

# fetchall()
remaining_rows = cursor.fetchall()
print("Оставшиеся строки:", remaining_rows)

conn.close()
```

# Самостоятельная работа

- Задание 1: Создание базы данных. Напишите Python-скрипт, который подключится к серверу PostgreSQL и создаст новую базу данных с именем "mydatabase".
- Задание 2: Создание таблицы. Напишите Python-скрипт, который подключится к базе данных "mydatabase" и создаст в ней таблицу "users" со следующими полями: id (SERIAL PRIMARY KEY), name (VARCHAR(255)), age (INTEGER).

- Задание 3: Добавление данных. Напишите Python-скрипт, который подключится к базе данных "mydatabase" и добавит в таблицу "users" следующие записи:

  Имя: John Doe, Возраст: 30

  Имя: Jane Doe, Возраст: 25

  Имя: Bob Smith, Возраст: 35
- Задание 4: Просмотр данных. Напишите Python-скрипт, который подключится к базе данных "mydatabase" и выведет все записи из таблицы "users".
- Задание 5: Удаление данных. Напишите Python-скрипт, который подключится к базе данных "mydatabase" и удалит все записи из таблицы "users" с возрастом больше 30 лет.