07.05.2019 в 09:00 LegGnom

Работа с PostgreSQL в Python

PostgreSQL - одна из самых продвинутых и широко используемых систем управления реляционными базами данных. Он чрезвычайно популярен по многим причинам, среди которых есть открытый код, его расширяемость и способность обрабатывать различные типы приложений и различные нагрузки.

С помощью Python вы можете легко установить соединение с вашей базой данных PostgreSQL. Существует множество драйверов Python для PostgreSQL, наиболее популярным из которых является «psycopg». Текущая версия - psycopg2.

В этой статье мы обсудим, как получить доступ к базе данных PostgreSQL в Python с помощью драйвера psycopg2.

Модуль psycopg2

Мы можем интегрировать Postgres с Python, используя модуль psycopg2. psycopg2 - это адаптер базы данных Postgres для Python. Чтобы использовать этот модуль, вы должны сначала установить его. Это можно сделать с помощью команды `pip`, как показано ниже:

```
$ pip3 install psycopg2
```

Обратите внимание, что я использую Python 3.5, поэтому я использовал `pip3` вместо `pip`.

После установки модуля вы можете использовать его для подключения к вашей базе данных в вашем приложении.

Подключение к базе данных

Чтобы подключиться к вашей базе данных, вы должны сначала создать объект подключения, представляющий базу данных. Затем вы должны создать объект курсора, чтобы помочь вам в выполнении ваших операторов SQL.

В следующем примере показано, как установить соединение с базой данных с именем «postgres»:

```
import psycopg2

con = psycopg2.connect(
  database="postgres",
  user="postgres",
  password="",
  host="127.0.0.1",
```

```
  port="5432"
)

print("Database opened successfully")
```

В результате получим:

```
Database opened successfully
```

Ниже приведен список параметров, которые были переданы методу `connect()` :

- `database` : Имя базы данных, к которой нужно подключиться.
- `user` : Имя пользователя, которое будет использоваться для аутентификации.
- `password` : Пароль базы данных для пользователя.
- `host` : Адрес сервера базы данных. Например, имя домена, «localhost» или IP-адрес.
- `port` : Номер порта. Если вы не предоставите это, будет использоваться значение по умолчанию, а именно 5432.

Обратите внимание, что значения для вышеуказанных параметров должны быть правильными, чтобы соединение было успешным. Если нет, то будет сгенерировано исключение. Вывод в приведенном выше коде показывает, что соединение с базой данных установлено успешно.

Создание таблицы

Чтобы создать таблицу Postgres в Python, мы используем оператор SQL `CREATE TABLE` . Этот запрос должен быть выполнен после установления соединения с базой данных. Мы также создаем объект курсора, вызывая метод `cursor()` , который принадлежит объекту `connection` . Этот объект `cursor` используется для фактического выполнения ваших команд.

Затем мы вызываем метод `execute()` объекта `cursor` , чтобы помочь нам в создании таблицы. Наконец, нам нужно зафиксировать и закрыть соединение. «Фиксация» соединения говорит драйверу о необходимости посылать команды в базу данных.

Следующий пример демонстрирует это:

```
import psycopg2

con = psycopg2.connect(
  database="postgres",
  user="postgres",
```

```
  password="Kaliakakya",
  host="127.0.0.1",
  port="5432"
)

print("Database opened successfully")cur = con.cursor()
cur.execute('''CREATE TABLE STUDENT
     (ADMISSION INT PRIMARY KEY NOT NULL,
     NAME TEXT NOT NULL,
     AGE INT NOT NULL,
     COURSE CHAR(50),
     DEPARTMENT CHAR(50));''')

print("Table created successfully")
con.commit()
con.close()
```

В результате получим:

```
Database opened successfully
Table created successfully
```

Этот метод `commit()` помогает нам применить изменения, которые мы внесли в базу данных, и эти изменения не могут быть отменены, если `commit()` выполнится успешно. Метод `close()` закрывает соединение с базой данных.

На данный момент мы создали таблицу с 4 столбцами, каждый из которых имеет различные типы данных. Приведенный выше вывод показывает, что таблица была успешно создана.

Вставка данных

Мы можем вставить одну запись или несколько записей в таблицу базы данных Postgres. Опять же, мы должны сначала установить соединение с сервером базы данных, вызвав функцию `connect()`. Затем мы должны создать объект курсора, вызвав метод `cursor()`. Наконец, мы должны выполнить инструкцию `INSERT` через метод `execute()`, чтобы добавить данные в таблицу.

Вот пример этого в действии:

```
import psycopg2
```

```
con = psycopg2.connect(
  database="postgres",
  user="postgres",
  password="Kaliakakya",
  host="127.0.0.1",
  port="5432"
)

print("Database opened successfully")
cur = con.cursor()
cur.execute(
  "INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES (3420, 'John', 18, 'Computer Science', 'ICT')"
)

con.commit()
print("Record inserted successfully")

con.close()
```

В результате получим:

```
Database opened successfully
Record inserted successfully
```

После запуска этого кода мы вставили одну запись в нашу таблицу базы данных. Это было сделано путем указания имени таблицы, а также столбцов, в которые нам нужно вставить данные. Мы также можем вставить несколько записей одной командой. Например:

```
import psycopg2

con = psycopg2.connect(
   database="postgres",
   user="postgres",
   password="Kaliakakya",
   host="127.0.0.1",
   port="5432"
)

print("Database opened successfully")

cur = con.cursor()cur.execute(
```

```
   "INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES (3419, 'Abel', 17, 'Computer Science', 'ICT')"
)
cur.execute(
  "INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES (3421, 'Joel', 17, 'Computer Science', 'ICT')"
)
cur.execute(
  "INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES (3422, 'Antony', 19, 'Electrical Engineering', 'Engineering')"
)
cur.execute(
  "INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES (3423, 'Alice', 18, 'Information Technology', 'ICT')"
)

con.commit()
print("Records inserted successfully")

con.close()
```

В результате получим:

```
Database opened successfully
Records inserted successfully
```

Поскольку метод `commit()` не вызывается до тех пор, пока мы не «выполним» все операторы `INSERT`, несколько записей вставляются с помощью одного вызова метода `commit()`.

Извлечение данных

Вы можете выбрать данные из базы данных Postgres и просмотреть записи таблицы. Сначала вы должны установить соединение с базой данных, используя функцию `connect()`. Затем следует создать новый курсор, вызвав метод `cursor()`. Созданный объект курсора можно затем использовать для выполнения оператора запроса данных из базы данных `SELECT`.

Например:

```
import psycopg2

con = psycopg2.connect(
  database="postgres",
  user="postgres",
  password="Kaliakakya",
```

```
  host="127.0.0.1",
  port="5432"
)

print("Database opened successfully")cur = con.cursor()
cur.execute("SELECT admission, name, age, course, department from STUDENT")

rows = cur.fetchall()
for row in rows:
   print("ADMISSION =", row[0])
   print("NAME =", row[1])
   print("AGE =", row[2])
   print("COURSE =", row[3])
   print("DEPARTMENT =", row[4], "\n")

print("Operation done successfully")
con.close()
```

В результате получим:

```
Database opened successfully
ADMISSION = 3420
NAME = John
AGE = 18
COURSE = Computer Science
DEPARTMENT = ICT

ADMISSION = 3419
NAME = Abel
AGE = 17
COURSE = Computer Science
DEPARTMENT = ICT

ADMISSION = 3421
NAME = Joel
AGE = 17
COURSE = Computer Science
DEPARTMENT = ICT

ADMISSION = 3422
NAME = Antony
AGE = 19
```

```
COURSE = Electrical Engineering
DEPARTMENT = Engineering

ADMISSION = 3423
NAME = Alice
AGE = 18
COURSE = Information Technology
DEPARTMENT = ICT

Operation done successfully
```

Здесь мы извлекли данные из базы данных, указав таблицу и имена столбцов, которые нам нужно извлечь из таблицы базы данных. Эти данные возвращаются нам в виде списка кортежей, причем список «верхнего уровня» представляет собой строки данных. Тогда каждая строка является кортежем данных столбца. Если строки для запроса не возвращаются, тогда возвращается пустой список `fetchall()` .

Обновление таблиц

Мы можем обновить или изменить детали записи, которая уже была вставлена в таблицу базы данных. Сначала мы должны установить соединение с базой данных, используя метод `connect()` . Далее мы вызываем функцию `cursor()` для создания объекта курсора. Наконец, мы запускаем метод `execute()` для выполнения оператора `UPDATE` с входными значениями.

Например:

```
import psycopg2

con = psycopg2.connect(
  database="postgres",
  user="postgres",
  password="Kaliakakya",
  host="127.0.0.1",
  port="5432"
)

print("Database opened successfully")
cur = con.cursor()
cur.execute("UPDATE STUDENT set AGE = 20 where ADMISSION = 3420")
con.commit()

print("Total updated rows:", cur.rowcount)
cur.execute("SELECT admission, age, name, course, department from STUDENT")
```

```
rows = cur.fetchall()
for row in rows:
   print("ADMISSION =", row[0])
   print("NAME =", row[1])
   print("AGE =", row[2])
   print("COURSE =", row[2])
   print("DEPARTMENT =", row[3], "\n")

print("Operation done successfully")
con.close()
```

В результате получим:

```
Database opened successfully
Total updated rows: 1
ADMISSION = 3419
NAME = 17
AGE = Abel
COURSE = Abel
DEPARTMENT = Computer Science

ADMISSION = 3421
NAME = 17
AGE = Joel
COURSE = Joel
DEPARTMENT = Computer Science

ADMISSION = 3422
NAME = 19
AGE = Antony
COURSE = Antony
DEPARTMENT = Electrical Engineering

ADMISSION = 3423
NAME = 18
AGE = Alice
COURSE = Alice
DEPARTMENT = Information Technology

ADMISSION = 3420
NAME = 20
```

```
AGE = John
COURSE = John
DEPARTMENT = Computer Science

Operation done successfully
```

В приведенном выше примере мы обновили значение столбца `AGE` для всех строк, для которых установлено значение `ADMISSION` 3420. После выполнения обновления мы извлекаем эти данные, чтобы убедиться, что соответствующие строки / столбцы были обновлены.

Удаление строк

Чтобы удалить запись из таблицы базы данных Postgres, мы должны сначала установить соединение с сервером базы данных. Во-вторых, объект курсора должен быть создан путем вызова функции `cursor()` . Затем мы запускаем оператор `DELETE` для выполнения удаления.

Например:

```
import psycopg2

con = psycopg2.connect(
  database="postgres",
  user="postgres",
  password="Kaliakakya",
  host="127.0.0.1",
  port="5432"
)

print("Database opened successfully")
cur = con.cursor()cur.execute("DELETE from STUDENT where ADMISSION=3420;")
con.commit()

print("Total deleted rows:", cur.rowcount)
cur.execute("SELECT admission, name, age, course, department from STUDENT")

rows = cur.fetchall()
for row in rows:
   print("ADMISSION =", row[0])
   print("NAME =", row[1])
   print("AGE =", row[2])
   print("COURSE =", row[3])
   print("DEPARTMENT =", row[4], "\n")
```

```
print("Deletion successful")
con.close()
```

В результате получим:

```
Database opened successfully
Total deleted rows: 1
ADMISSION = 3419
NAME = Abel
AGE = 17
COURSE = Computer Science
DEPARTMENT = ICT

ADMISSION = 3421
NAME = Joel
AGE = 17
COURSE = Computer Science
DEPARTMENT = ICT

ADMISSION = 3422
NAME = Antony
AGE = 19
COURSE = Electrical Engineering
DEPARTMENT = Engineering

ADMISSION = 3423
NAME = Alice
AGE = 18
COURSE = Information Technology
DEPARTMENT = ICT

Deletion successful
```

В этом примере мы удалили все записи, в которых регистрационный номер учащегося равен 3420, что в данном случае составляет всего одну строку. После извлечения данных с помощью `SELECT` мы видим, что эта запись не является частью вышеприведенного вывода, подтверждая, что она была удалена из базы данных.

Заключение

В Python есть разные способы доступа к базе данных PostgreSQL. Существует много драйверов баз данных для Python, которые мы можем использовать для этой цели, но psycopg - самый популярный. В этой статье мы показали, как установить модуль, установить соединение с базой данных PostgreSQL и выполнить распространенные SQL-запросы с использованием кода Python.

#PostgreSQL #Python