УДК 004.05

РАЗРАБОТКА БАЗЫ ДАННЫХ СКРИПТА ДЛЯ СБОРА ПРОДУКТОВЫХ МЕТРИК ИЗ СЕРВИСА УПРАВЛЕНИЯ ПРОЕКТАМИ

Терехин Д.Е.,

магистрант,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Виноградская М.Ю.,

к.пед.н., доцент,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Аннотация.

В статье рассматривается проблема разработки база данных скрипта для сбора продуктовых метрик. При разработке скрипта авторы использовали СУБД PostgreSQL которая обладает следующими преимуществами: поддержкой различных типов индексов, разных форматов данных (включая JSON), высокой производительностью, надежностью и целостностью данных. Для взаимодействия с PostgreSQL средствами Python применяется удобный драйвер psycopg2, предоставляющий полный доступ ко всей функциональности системы. Также авторы представляют два метода создания базы данных: первый предполагает использование специальной среды разработки — DataGrip от JetBrains, второй предусматривает выполнение команд через терминальную оболочку. В конце делаются выводы о проделанной работе.

Ключевые слова: метрики, скрипт, база данных, СУБД PostgreSQL, среда разработки, Python.

DEVELOPING A DATABASE SCRIPT FOR COLLECTING PRODUCT METRICS FROM A PROJECT MANAGEMENT SERVICE

Terekhin D.E.
Undergraduate,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Vinogradskaya M.Y.,
Candidate of Pedagogical Sciences, Associate Professor,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Annotation.
The article discusses the problem of developing a database script for collecting product metrics. When developing the script, the authors used the PostgreSQL DBMS, which has the following advantages: support for various types of indexes, different data formats (including JSON), high performance, reliability, and data integrity. To interact with PostgreSQL using Python, the convenient psycopg2 driver is used, providing full access to all system functionality. The authors also present two methods for creating a database: the first involves using a special development environment, DataGrip from JetBrains, the second involves executing commands through a terminal shell. In the end, conclusions are made about the work done.
Keywords: metrics, script, database, PostgreSQL DBMS, development environment, Python.

При разработке скрипта для сбора продуктовых метрик была использована СУБД PostgreSQL для хранения данных [1]. Она была выбрана из-за ряда преимуществ, таких как удобство работы за счет разных индексов, поддержки JSON, высокой производительности, надежности и целостности данных.

PostgreSQL является реляционной системой управления базами данных с открытым исходным кодом и широко используется в разных приложениях. Эта СУБД поддерживает различные типы таблиц, полнотекстовый поиск и транзакции на уровне отдельных записей. PostgreSQL также обеспечивает обработку строковых и текстовых данных фиксированной и переменной длины.

PostgreSQL - это мощная система управления реляционными базами данных, используемая в различных областях благодаря своей продвинутой функциональности и гибкости. Она пользуется популярностью среди разработчиков во многом благодаря своей расширяемости и способности эффективно обрабатывать разнообразные типы приложений и нагрузку. Взаимодействие с базой данных PostgreSQL из Python также является простым благодаря наличию множества драйверов, одним из популярных является psycopg, сейчас актуальной версией является psycopg2. Подключение и использование PostgreSQL с помощью Python позволяет разработчикам максимально эффективно использовать все преимущества данной СУБД в своих проектах [3; 4].

Для взаимодействия скрипта с базой данных будет использоваться модуль psycopg2. Этот модуль представляет собой адаптер базы данных PostgreSQL для Python, который позволяет взаимодействовать с базой данных из Python-приложений. Чтобы начать использовать модуль psycopg2, необходимо сначала установить его. Это можно сделать с помощью команды pip, которая позволит установить этот модуль в виртуальное окружение проекта. Установка psycopg2 с помощью pip делает процесс настройки Python с PostgreSQL более простым и удобным. Для подключения к базе будет использоваться следующая

```
def db_connect():
    db = psycopg2.connect(f'postgresql://{USER}:{PASSWORD}@{HOST}:{PORT}/{DATABASE}')
    db.autocommit = True
    return db
```

конструкция (рисунок 1).

Рис.1 - Подключение к базе данных (составлено авторами)

Чтобы создать соединение необходимы следущие параметры:

– USER: Имя пользователя, которое будет использоваться для работы с базой и аутентификации

– PASSWORD: Пароль пользователя

– HOST: Адрес сервера или домена базы данных

– PORT: Номер порта. По умолчанию используется 5432

– DATABASE: Имя базы данных к которой создаётся подключение

В результате выполнения кода будет создано соединение с базой данных. В случае неправильных значений будет вызвано исключение.

Для создания таблицы в базе данных Postgres с использованием Python, применяется оператор SQL CREATE TABLE. Этот запрос следует выполнить после установления соединения с базой данных. Затем создадим объект курсора, вызывая метод cursor(). Курсор используется для выполнения конкретных команд. Метод execute() объекта курсора используется для создания таблицы. После этого необходимо зафиксировать изменения и закрыть соединение.

На рисунке 2 приведён пример кода в которм, при условии отсутствия таблицы в базе будет создана новая таблица с названием команды.

```
def create_table(db, sheet, columns=False, flow_efficiency=False):
    cursor = db.cursor()
    cursor.execute(f"SELECT table_name FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = '{sheet}'")
    if not cursor.rowcount:
        cursor.execute(f"CREATE TABLE {sheet} ()")
        if columns:
            for column in columns:
                _append_column(sheet, f'"{column}"', cursor, flow_efficiency)
```

Рис.2 - Создание таблицы при выполнении условия (составлено авторами)

Для добавления данных в таблицу базы данных Postgres необходимо устанавить соединение с сервером базы данных. После этого создаётся объект курсора, при вызове метода cursor(). Наконец, для добавления записей в таблицу выполняем инструкцию INSERT через метод execute(). (рисунок 3)

```
def _append_row(result, columns, sheet, cursor):
    for item in result:
        row = []
        for value in item:
            if isinstance(value, str):
                if value:
                    value = value.replace("'", '"')
                    row.append(f"'{value}'")
                else:
                    row.append(f'NULL')
            if isinstance(value, (int, float)):
                row.append(f"'{value}'")
        cursor.execute(f"INSERT INTO {sheet} ({', '.join(columns)}) VALUES ({', '.join(row)})")
```

Рис.3 - Добавление записи после формирования списка данных (составлено авторами)

Для получения данных из базы данных используется инструкция SELECT. Запрос выполняется так же после создания подключения (рисунок 4).

```
def get_date(db, sheet):
    cursor = db.cursor()
    sheet = sheet.replace(' ', '_').lower()
    cursor.execute(f"SELECT Дата FROM {sheet} ORDER BY Дата DESC LIMIT 1")
    return cursor.fetchone()


def get_columns(db, sheet):
    cursor = db.cursor()
    sheet = sheet.replace(' ', '_').lower()
    cursor.execute(f"SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '{sheet}' ORDER BY ORDINAL_POSITION")
    return [item[0] for item in cursor.fetchall()]
```

Рис.4 - Получение данных с помощью SELECT (составлено авторами)

Для удаления записи из таблицы базы данных необходимо сначала установить соединение с базой данных. Затем следует создать объект курсора с помощью вызова метода cursor(). После этого выполняется оператор DELETE для удаления записи (рисунок 5).

```
def refresh_line(db, sheet, result, flow_efficiency):
    cursor = db.cursor()
    sheet = sheet.replace(' ', '_').lower()
    cursor.execute(f"DELETE FROM {sheet} WHERE Дата = '{result[1][0]}'")
    append_in_table(db, sheet, result, flow_efficiency)
```

Рис.5 - Удаление данных с помощью DELETE (составлено авторами)

Существует ещё один способ взаимодействия с записями в базе данных. Для изменения существующей записи можно воспользоваться UPDATE для изменения всех или определённых столбцов в базе данных.

В Python существует множество способов взаимодействия с базами, но в данном проекте будет использоваться именно psycopg2 из-за его простого и понятного синтаксиса [5]. Мы рассмотрели два варианта создания базы данных через IDE DataGrip от JetBrains и через консоль.

Для создания базы через DataGrip понадобится скачать и установить приложение с официального сайта https://www.jetbrains.com/ru-ru/datagrip/. Далее создадим отдельного пользователя, который будет взаимодействовать с базой и таблицами. Перейдём в меню File -> New -> User (рисунок 6).

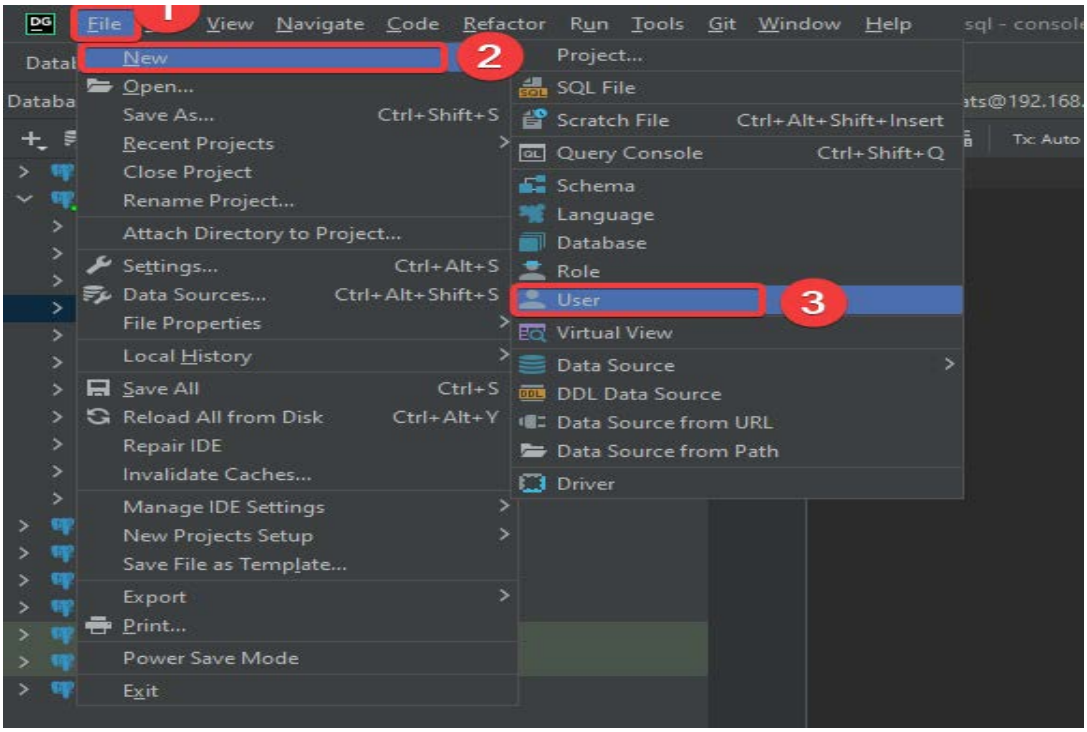

Рис.6 - Меню IDE (составлено авторами)

В открывшемся окне укажем имя пользователя и поставим галочку на против Create DB для возможности создания базы (рисунок 7).

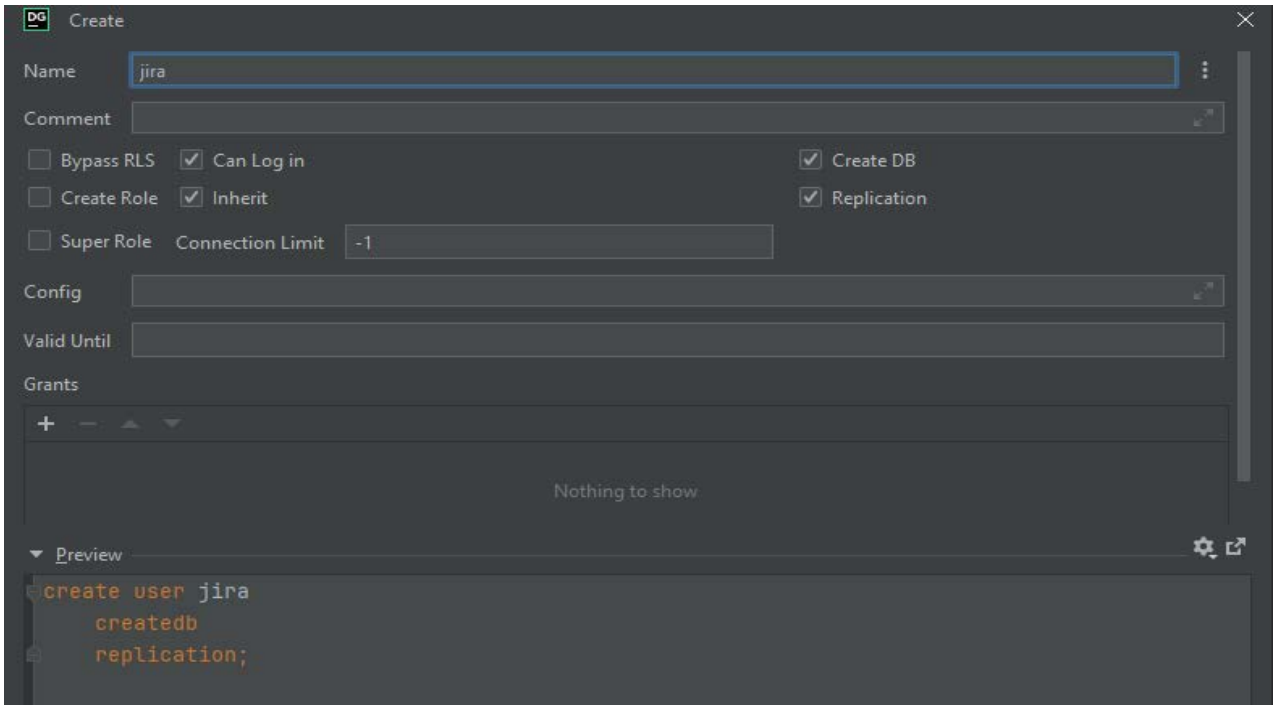

Рис.7 - Окно создания пользователя (составлено авторами)

Следующим шагом будет создание новой базы данных, нажимаем New -> Database (рисунок 8).

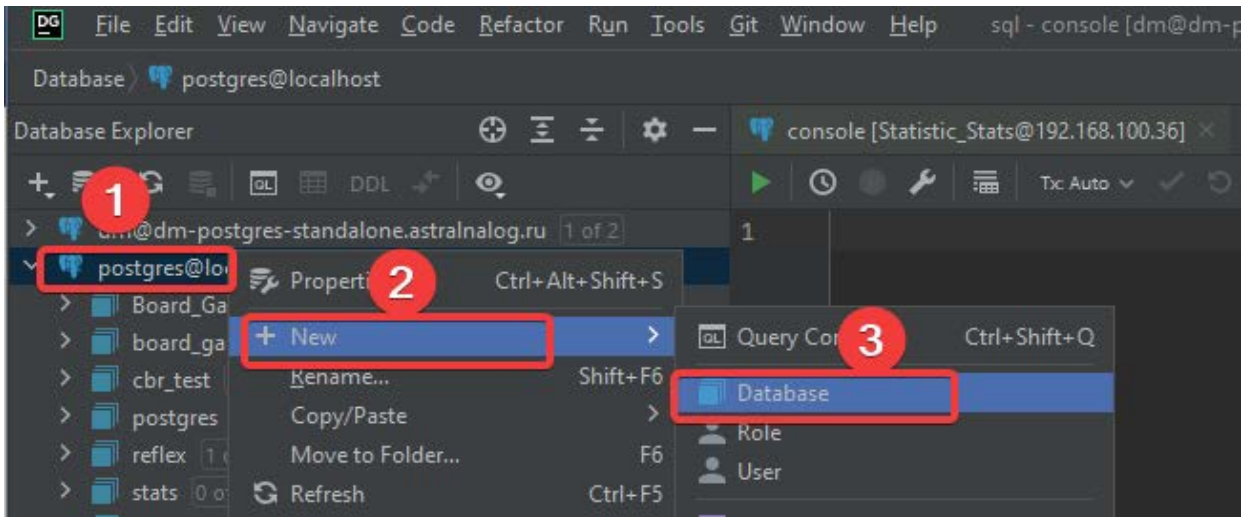

Рис.8 - Шаги создания базы (составлено авторами)

В открывшемся окне указываем название базы данных и владельца (Рисунок 9).

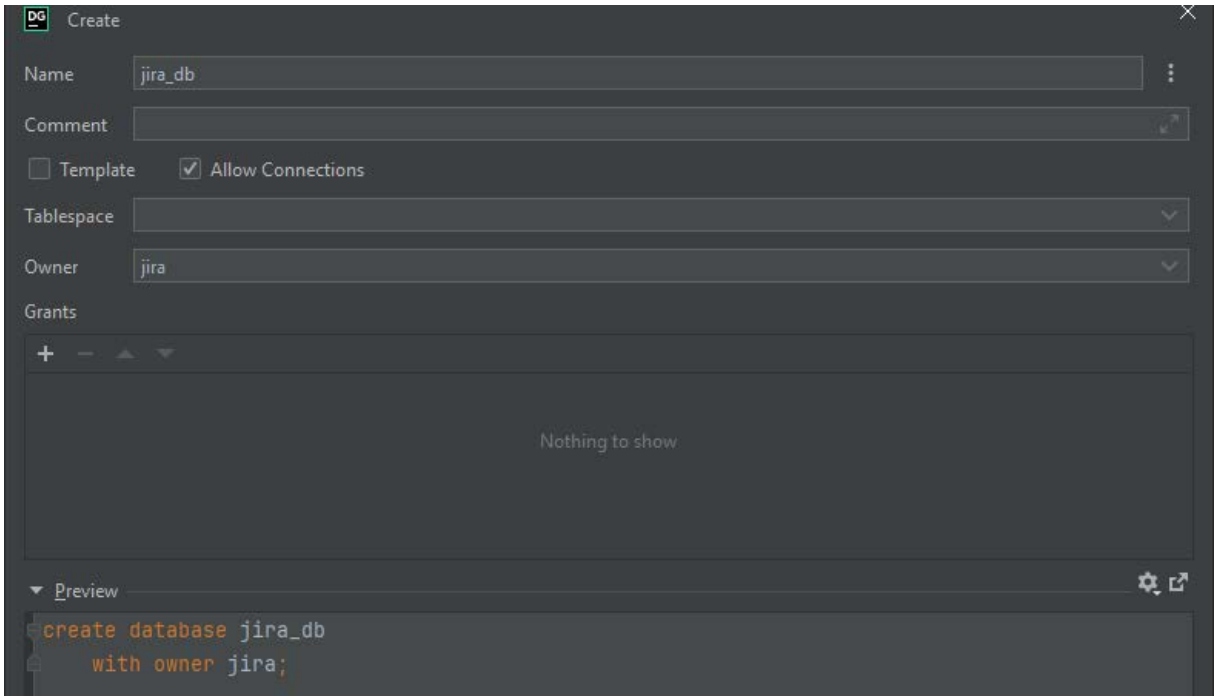

Рис.9 - Окно создания базы данных (составлено авторами)

В результате в списке появится новая база “jira_db”.

Создание базы данных через консоль подразумевает схожие действия такие как создание пользователя, присвоение названия базе, установка роли. Прежде чем использовать команды Postgres необходимо подключиться к серверу для этого прописываем: “psql -h localhost -p 5432 -U postgres -d postgres”, вводим пароль и попадаем в оболочку psql в которой будем проводить дальнейшие действия (рисунок 10).

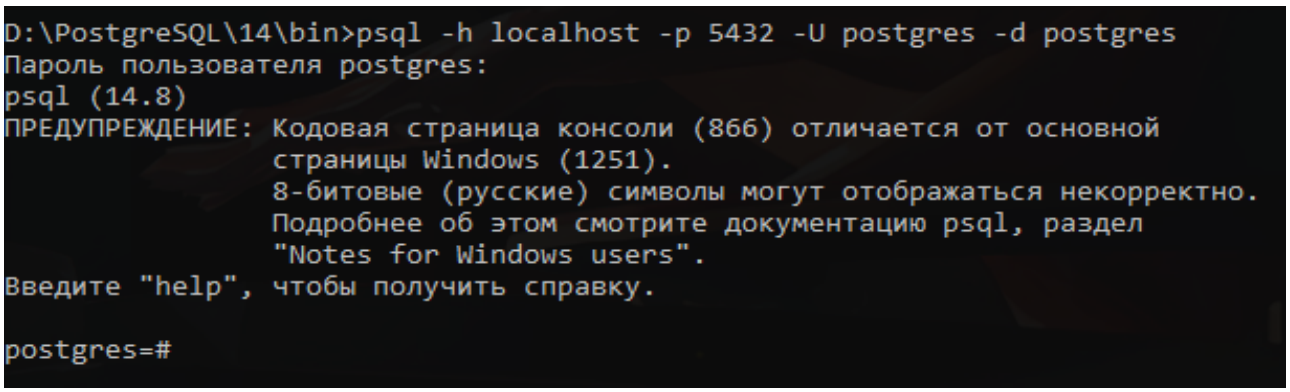

Рис.10 - Консоль psql (составлено авторами)

Для создания нового пользователя необходимо прописать команду: “CREATE USER jira WITH CREATEDB LOGIN INHERIT;” после ввода получим ответ CREATE ROLE, что сигнализирует об успешном создании пользователя. Приступим к созданию базы. С помощью команды “CREATE DATABASE jira_db;” создадим базу данных и сделаем пользователя “jira” владельцем: “ALTER DATABASE jira_db owner to jira_dm;”

Результат должен выглядеть следующим образом (рисунок 11).

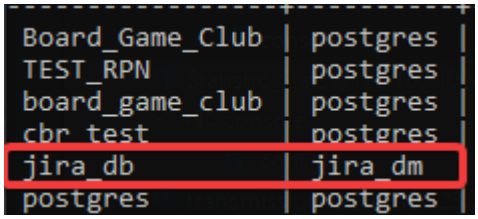

Рис.11 - Просмотр списка баз данных в psql (составлено авторами)

После выполнения всех вышеперечисленных действий можно приступать к заполнению файла конфигурации и запуску обработки.

Таким образом, при разработке скрипта для сбора продуктовых метрик была использована СУБД PostgreSQL для хранения данных. Она была выбрана из-за ряда преимуществ, таких как удобство работы за счет разных индексов, поддержки JSON, высокой производительности, надежности и целостности данных. PostgreSQL является реляционной системой управления базами данных с открытым исходным кодом и широко используется в разных приложениях. Эта СУБД поддерживает различные типы таблиц, полнотекстовый поиск и транзакции на уровне отдельных записей. PostgreSQL также обеспечивает обработку строковых и текстовых данных фиксированной и переменной длины. Взаимодействие с базой данных PostgreSQL из Python также является простым благодаря наличию множества драйверов, одним из популярных является psycopg, сейчас актуальной версией является psycopg2. Подключение и

использование PostgreSQL с помощью Python позволяет разработчикам максимально эффективно использовать все преимущества данной СУБД в своих проектах. В результате проделанной работы была спроектирована база данных и описаны основные методы, позволяющие выполнять необходимые выгрузки.

Библиографический список:

1. Григорьев, М. В. Проектирование информационных систем: учеб. пособие для вузов / М. В. Григорьев, И. И. Григорьева. -М.: Юрайт, - 2021. — 318 с. — (Высшее образование). — ISBN 978-5-534-01305-4. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/470711/p.1 (дата обращения: 24.05.2024).

2. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — М.: Юрайт, 2021. — 432 с.

3. Прохоренок, Н. А. Python 3 и PyQt 5. Разработка приложений / Николай Прохоренок, Владимир Дронов. - Санкт-Петербург : БХВ-Петербург - 2016. - 832 с.

4. Прохоренок, Н. А. Python 3 и PyQt. Разработка приложений / Н.А. Прохоренок. - СПб.: БХВ-Петербург - 2012. — 704 с.

5. Райордан, Р. Основы реляционных баз да нных. Базовый курс. Теория и практика / Р. Райордан. - М.: Русская редакция - 2001. — 390 с.

Оригинальность 75%