Platform V Pangolin SE

Руководство прикладного разработчика

(PSQ)

Руководство прикладного разработчика

Platform V Pangolin SE — система управления базами данных, основанная на PostgreSQL. В этом документе приведена информация, необходимая для разработки приложений, подключаемых к Platform V Pangolin SE.

Этот документ предназначен для разработчиков приложений, подключаемых к СУБД Platform V Pangolin SE.

Системные требования

Platform V Pangolin SE развертывается на базе ОС Альт 8 СП.

Для установки Platform V Pangolin SE на сервер на базе Linux необходимо следующее ПО:

- Ansible не ниже версии 2.9.2;
- sshpass;
- Модули Python:
  - jmespath версии 0.9.4;
  - netaddr версии 0.7.19;
  - PyYAML версии 5.3.

Функциональные требования к оборудованию перечислены в таблице ниже.

Назначение сервера	CPU (core)	RAM, GB	Локальные диски, GB	Сетевые интерфейсы	Интерфейсы сети хранения
Сервер баз данных	4	24	OS: 2 SSD, 150 GB + 150 GB	4	2
Сервер обработки событий аудита (syslog-ng)	4	24	OS: 2 SSD, 150 GB + 150 GB	4	2

Подключение и конфигурирование

Здесь описывается разработка приложения на языке Java с использованием библиотеки JDBC.

Подготовка удаленной машины к подключению

Для начала необходимо разрешить удаленные подключения напрямую к Platform V Pangolin SE. Для этого нужно отредактировать файлы конфигурации.

Сделайте резервные копии файлов конфигурации, на всякий случай. Для этого от имени пользователя *Postgres* выполните:

```
cp /pgdata/04/data/postgresql.conf /pgdata/04/data/postgresql.conf.backup
cp /pgdata/04/data/pg_hba.conf /pgdata/04/data/pg_hba.conf.backup
```

Теперь, если что-то пойдет не так, вы всегда сможете восстановить файлы в исходное состояние, выполнив:

```
cp /pgdata/04/data/postgresql.conf.backup /pgdata/04/data/postgresql.co
nf
cp /pgdata/04/data/pg_hba.conf.backup /pgdata/04/data/pg_hba.conf
```

На виртуальной машине от имени пользователя *Postgres* откройте файл *postgresql.conf*, найдите в нем параметр *listen_addresses* и установите значение равным `'*'`. Это позволит Platform V Pangolin SE принимать запросы с любого IP-адреса.

Чтобы открыть файл *postgresql.conf*, выполните в консоли эту команду:

```
vim /pgdata/04/data/postgresql.conf
```

> Для тех, кто не знаком с редактором Vim
>
> Чтобы начать редактирование файла, нажмите *I*. Чтобы выйти из режима редактирования, нажмите *Esc*. Чтобы сохранить файл и выйти из редактора, нажмите *Esc*, введите *:wq* и нажмите *Enter*. Чтобы выйти из редактора без сохранения файла, нажмите *Esc*, введите *:q* и нажмите *Enter*. Если вы запутались, то закройте файл без сохранения и попробуйте еще раз.

Теперь аналогичным образом от имени пользователя *Postgres* откройте файл *pg_hba.conf* и добавьте в него следующие строки:

```
vim /pgdata/04/data/pg_hba.conf
```

```
host    all             all             0.0.0.0/0                       m
d5
host    all             all             ::/0                            m
d5
```

Наконец, перезапустите Platform V Pangolin SE. От имени администратора ОС выполните:

```
sudo service postgresql stop
sudo service postgresql start
```

Теперь к СУБД теперь можно подключаться удаленно напрямую.

Установка JDBC

JDBC — это библиотека Java для работы с базами данных, включая PostgreSQL и Pangolin SE. Загрузите актуальную версию и поместите файл в папку на вашем компьютере, в которой вы будете писать код.

Разработка первого приложения с использованием программного продукта

В этом разделе описывается разработка приложений, взаимодействующих с СУБД Pangolin SE. Эти приложения будут подключаться к базе данных *postgres* от имени пользователя *postgres* и создавать, редактировать и удалять из неё данные.

Подключение к базе данных

Приведенный ниже код реализует подключение к базе данных *postgres* на локальной машине через порт 5433 от имени пользователя *postgres*.

```
import java.sql.DriverManager;
import java.sql.Connection;
```

```
import java.sql.SQLException;

public class Main {

   //  Учетные данные для подключения
      static final String DB_URL = "jdbc:postgresql://10.53.227.36:543
3/postgres"; //Замените на свой IP
      static final String USER = "postgres";
      static final String PASS = "B00tc@mpB00tc@mp"; //Замените на пар
оль postgres

  public static void main(String[] argv) {

    System.out.println("Проверка подключения драйвера JDBC");

    try {
        Class.forName("org.postgresql.Driver");
    } catch (ClassNotFoundException e) {
        System.out.println("Драйвер JDBC не обнаружен");
        e.printStackTrace();
        return;
    }

    System.out.println("Драйвер JDBC обнаружен, выполняется установка
соединения");

    Connection connection = null;

    try {
        connection = DriverManager
        .getConnection(DB_URL, USER, PASS);

    } catch (SQLException e) {
        System.out.println("Не удалось установить соединение");
        e.printStackTrace();
        return;
    }

    if (connection != null) {
        System.out.println("Соединение с базой данных установлено");
    } else {
        System.out.println("Не удалось установить соединение с базой д
анных");
    }
  }
}
```

Скомпилируйте программу и выполните её. Не забудьте подключить jar-файл с библиотекой JDBC.

```
java -cp ./postgresql-42.2.24.jar Main.java
```

Операция CREATE

Приведенный ниже код создаёт таблицу *mystore* в базе данных *testdb* и наполняет её данными.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;

public class Create {

   //  Учетные данные для подключения
      static final String DB_URL = "jdbc:postgresql://10.53.227.36:543
3/postgres"; //Замените на свой IP
      static final String USER = "postgres";
      static final String PASS = "B00tc@mpB00tc@mp"; //Замените на пар
оль postgres

    public static void main(String[] argv) {

        String query = "create table mystore(" + //Создание таблицы с
заголовками
        "ID int primary key     not null," +
        "Тип            text    not null," +
        "Цвет           text    not null," +
        "Материал       text    not null," +
        "Запас          int," +
        "Цена           real );" +
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  + //
... и наполнение ее данными
        "values (1, 'Футболка', 'Синий', 'Хлопок', 500, 2000.00 ); " +
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  +
        "values (2, 'Футболка', 'Желтый', 'Хлопок', 500, 2000.00 ); "
+
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  +
        "values (3, 'Футболка', 'Красный', 'Хлопок', 500, 2000.00 ); "
+
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  +
        "values (4, 'Рубашка', 'Синий', 'Шелк', 400, 2500.00 ); " +
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  +
        "values (5, 'Рубашка', 'Белый', 'Хлопок', 600, 1400.00 ); " +
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  +
        "values (6, 'Свитер', 'Синий', 'Шерсть', 100, 4000.00 ); " +
        "insert into mystore (ID,Тип,Цвет,Материал,Запас,Цена) "  +
        "values (7, 'Футболка', 'Синий', 'Нейлон', 500, 1000.00 ); ";
        try (Connection con = DriverManager.getConnection(DB_URL, USER
, PASS);

                PreparedStatement pst = con.prepareStatement(query)){
```

```
                    pst.executeUpdate();

            } catch (Exception e2) {
                        System.err.println("Операция выполнена с ошибк
ой");
                        System.err.println(e2);
                        System.exit(2);
            }

            System.out.println("Таблица создана и наполнена данными");
        }
    }
```

Скомпилируйте программу и выполните её.

```
java -cp ./postgresql-42.2.24.jar create.java
```

Операция SELECT

Приведенный ниже код выводит на экран записи из таблицы *mystore*:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;

public class Select {

   //  Учетные данные для подключения
      static final String DB_URL = "jdbc:postgresql://10.53.227.36:543
3/postgres"; //Замените на свой IP
      static final String USER = "postgres";
      static final String PASS = "B00tc@mpB00tc@mp"; //Замените на пар
оль postgres

    public static void main(String[] argv) {

        String query = "select * from mystore";

        try (Connection con = DriverManager.getConnection(DB_URL, USER
, PASS);

                PreparedStatement pst = con.prepareStatement(query);
                ResultSet rs = pst.executeQuery()) {

                    ResultSetMetaData meta = pst.getMetaData(); //Полу
чение и вывод названий столбцов таблицы
                        for (int i=1; i <= meta.getColumnCount(); i++)
                            {
                            System.out.print(meta.getColumnName(i) + "
```

```
| ");
                            }
                        System.out.print("\n");

                        while (rs.next()) { //Построчный вывод содержи
мого таблицы
                            for (int i=1; i <= meta.getColumnCount();
i++)
                                {
                                System.out.print(rs.getString(i) + " |
");
                                }
                            System.out.print("\n");
                        }

            } catch (Exception e2) {
                        System.err.println("Операция выполнена с ошибк
ой");
                        System.err.println(e2);
                        System.exit(2);
            }

            System.out.println("Таблица выведена на экран");
        }
    }
```

Скомпилируйте программу и выполните её. Вы увидите следующий результат:

```
%       java       -cp       ./postgresql-42.2.24.jar       select.java
id    |    Тип    |    Цвет    |    Материал    |    Запас    |    Цена    |
1    |    Футболка    |    Синий    |    Хлопок    |    500    |    2000    |
2    |    Футболка    |    Желтый    |    Хлопок    |    500    |    2000    |
3    |    Футболка    |    Красный    |    Хлопок    |    500    |    2000    |
4    |    Рубашка    |    Синий    |    Шелк    |    400    |    2500    |
5    |    Рубашка    |    Белый    |    Хлопок    |    600    |    1400    |
6    |    Свитер    |    Синий    |    Шерсть    |    100    |    4000    |
7    |    Футболка    |    Синий    |    Нейлон    |    500    |    1000    |
Таблица выведена на экран
```

Операция UPDATE

Приведенный ниже код изменяет данные в таблице *mystore* с помощью операции *update* и выводит на экран обновленные записи:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
```

```
public class Update {

   //  Учетные данные для подключения
      static final String DB_URL = "jdbc:postgresql://10.53.227.36:543
3/postgres"; //Замените на свой IP
      static final String USER = "postgres";
      static final String PASS = "B00tc@mpB00tc@mp"; //Замените на пар
оль postgres

    public static void main(String[] argv) {

        String query = "update mystore set Цена = 999 where Материал='
Хлопок'"; //Обновление данных в таблице

        try (Connection con = DriverManager.getConnection(DB_URL, USER
, PASS);

                PreparedStatement pst = con.prepareStatement(query)){

                    pst.executeUpdate();

            } catch (Exception e2) {
                        System.err.println("Операция выполнена с ошибк
ой");
                        System.err.println(e2);
                        System.exit(2);
            }

        System.out.println("Данные в таблице обновлены");



        // Далее идет код вывода результата из предыдущего шага

        query = "select * from mystore";

        try (Connection con = DriverManager.getConnection(DB_URL, USER
, PASS);

                PreparedStatement pst = con.prepareStatement(query);
                ResultSet rs = pst.executeQuery()) {

                    ResultSetMetaData meta = pst.getMetaData(); //Полу
чение и вывод названий столбцов таблицы
                        for (int i=1; i <= meta.getColumnCount(); i++)
                            {
                            System.out.print(meta.getColumnName(i) + "
| ");
                            }
                        System.out.print("\n");

                        while (rs.next()) { //Построчный вывод содержи
```

```
мого таблицы
                                for (int i=1; i <= meta.getColumnCount();
i++)
                                    {
                                    System.out.print(rs.getString(i) + " |
");
                                    }
                                System.out.print("\n");
                            }

                } catch (Exception e2) {
                            System.err.println("Операция выполнена с ошибк
ой");
                            System.err.println(e2);
                            System.exit(2);
                }

                System.out.println("Таблица выведена на экран");
            }
        }
```

Скомпилируйте и выполните программу. Вы увидите следующее:

```
java -cp ./postgresql-42.2.24.jar update.java
Данные в таблице обновлены
id | Тип | Цвет | Материал | Запас | Цена |
4 | Рубашка | Синий | Шелк | 400 | 2500 |
6 | Свитер | Синий | Шерсть | 100 | 4000 |
7 | Футболка | Синий | Нейлон | 500 | 1000 |
1 | Футболка | Синий | Хлопок | 500 | 999 |
2 | Футболка | Желтый | Хлопок | 500 | 999 |
3 | Футболка | Красный | Хлопок | 500 | 999 |
5 | Рубашка | Белый | Хлопок | 600 | 999 |
Таблица выведена на экран
```

Операция DELETE

Приведенный ниже код удаляет данные из таблицы *mystore* с помощью операции *delete* и выводит на экран обновленные записи:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;

public class Delete {

   //  Учетные данные для подключения
```

```
      static final String DB_URL = "jdbc:postgresql://10.53.227.36:543
3/postgres"; //Замените на свой IP
      static final String USER = "postgres";
      static final String PASS = "B00tc@mpB00tc@mp"; //Замените на пар
оль postgres

    public static void main(String[] argv) {

        String query = "delete from mystore where Материал != 'Хлопок'
"; //Удаление записей из таблицы

        try (Connection con = DriverManager.getConnection(DB_URL, USER
, PASS);

                PreparedStatement pst = con.prepareStatement(query)){

                    pst.executeUpdate();

            } catch (Exception e2) {
                        System.err.println("Операция выполнена с ошибк
ой");
                        System.err.println(e2);
                        System.exit(2);
            }

        System.out.println("Данные в таблице обновлены");



        // Далее идет код вывода результата из предыдущего шага

        query = "select * from mystore";

        try (Connection con = DriverManager.getConnection(DB_URL, USER
, PASS);

                PreparedStatement pst = con.prepareStatement(query);
                ResultSet rs = pst.executeQuery()) {

                    ResultSetMetaData meta = pst.getMetaData(); //Полу
чение и вывод названий столбцов таблицы
                        for (int i=1; i <= meta.getColumnCount(); i++)
                            {
                            System.out.print(meta.getColumnName(i) + "
| ");
                            }
                        System.out.print("\n");

                        while (rs.next()) { //Построчный вывод содержи
мого таблицы
                            for (int i=1; i <= meta.getColumnCount();
i++)
                                {
```

```
                                    System.out.print(rs.getString(i) + " |
");
                                    }
                                System.out.print("\n");
                            }

                } catch (Exception e2) {
                            System.err.println("Операция выполнена с ошибк
ой");
                            System.err.println(e2);
                            System.exit(2);
                }

                System.out.println("Таблица выведена на экран");
            }
        }
```

Скомпилируйте и выполните программу. Вы увидите следующее:

```
java -cp ./postgresql-42.2.24.jar delete.java
Данные в таблице обновлены
id | Тип | Цвет | Материал | Запас | Цена |
1 | Футболка | Синий | Хлопок | 500 | 999 |
2 | Футболка | Желтый | Хлопок | 500 | 999 |
3 | Футболка | Красный | Хлопок | 500 | 999 |
5 | Рубашка | Белый | Хлопок | 600 | 999 |
Таблица выведена на экран
```

Использование программного продукта

Platform V Pangolin SE может использоваться как СУБД широкого профиля и выполнять любые задачи, стоящие перед современными системами хранения и обработки данных, однако функциональные особенности системы, такие как расширенные средства обеспечения безопасности, контроля доступа и защиты пользовательских данных, превращают Platform V Pangolin SE в надежное решение для таких сфер деятельности, где конфиденциальность и сохранность данных имеют первостепенное значение, а их утрата или кража могут обернуться катастрофическими финансовыми и репутационными потерями и даже привести к уголовной ответственности.

В число приоритетных для Platform V Pangolin SE секторов экономики входят:

- банковская и финансовая сферы,
- государственные информационные системы и сети,
- здравоохранение.

Внедрение Platform V Pangolin SE в организации, принадлежащие к этим отраслям, приведет к повышению уровня безопасности обрабатываемой информации и снизит риски её хищения или утраты.