Java™

JDBC

База Данных

База данных - набор сведений, хранящихся некоторым упорядоченным способом.

Система управления базами данных (СУБД) - это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д.

SQL - язык структурированных запросов, основной задачей которого является предоставление простого способа считывания и записи информации в базу данных.

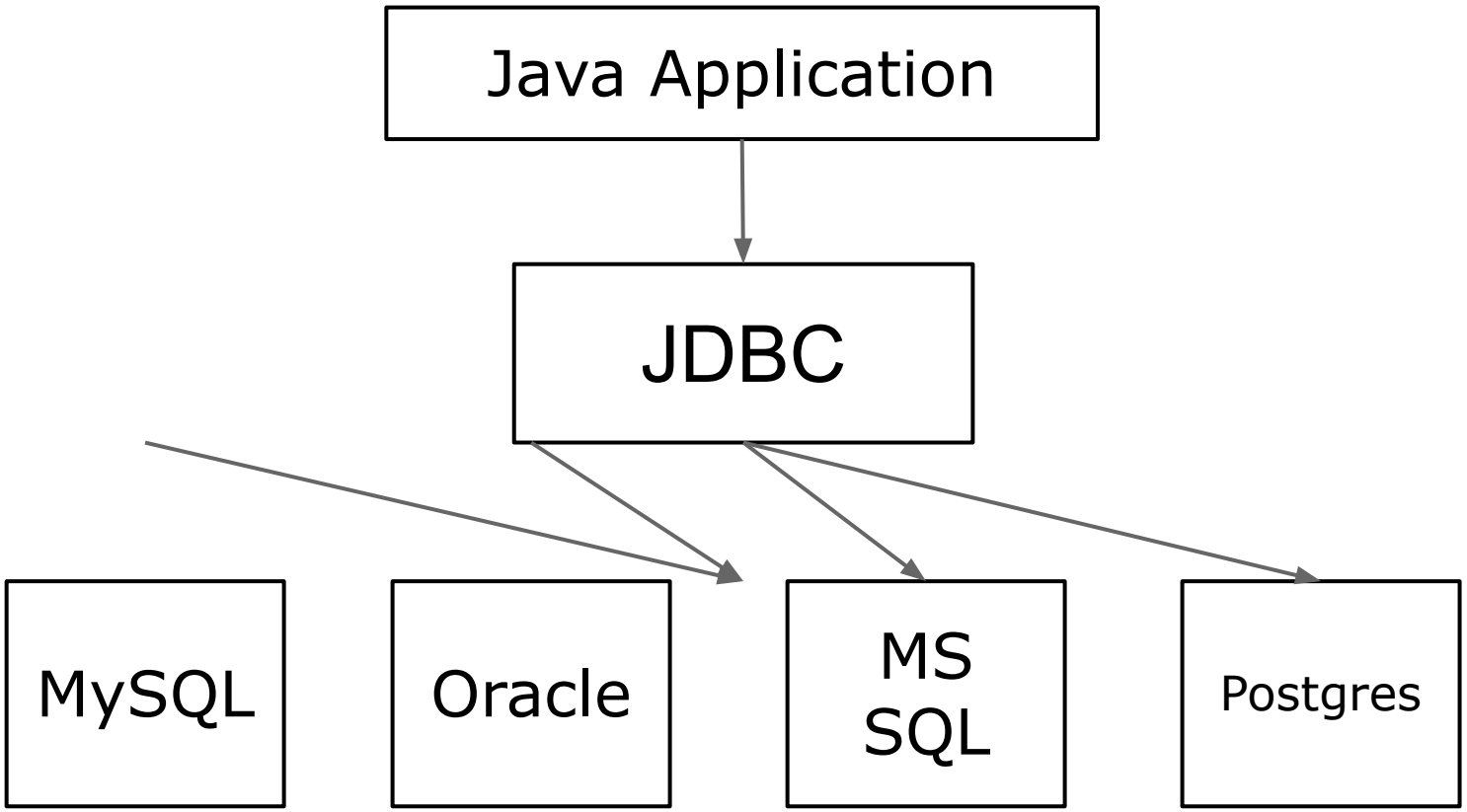
JDBC

JDBC (Java DataBase Connectivity) — соединение с базами данных на Java.

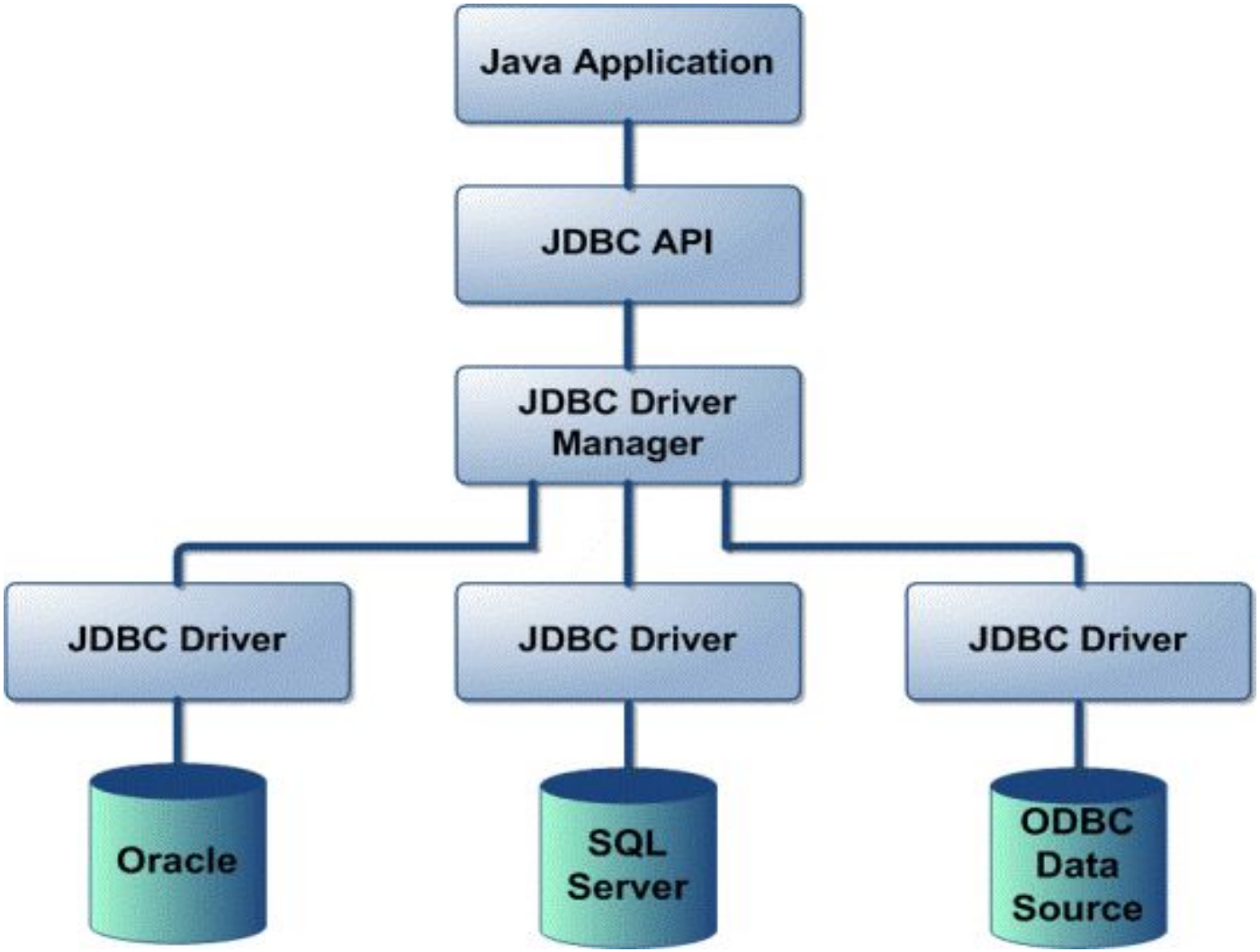
Основная цель:

- Установить соединение с базой данных
- Посылать запросы и изменять состояние базы данных
- Обрабатывать результаты запросов

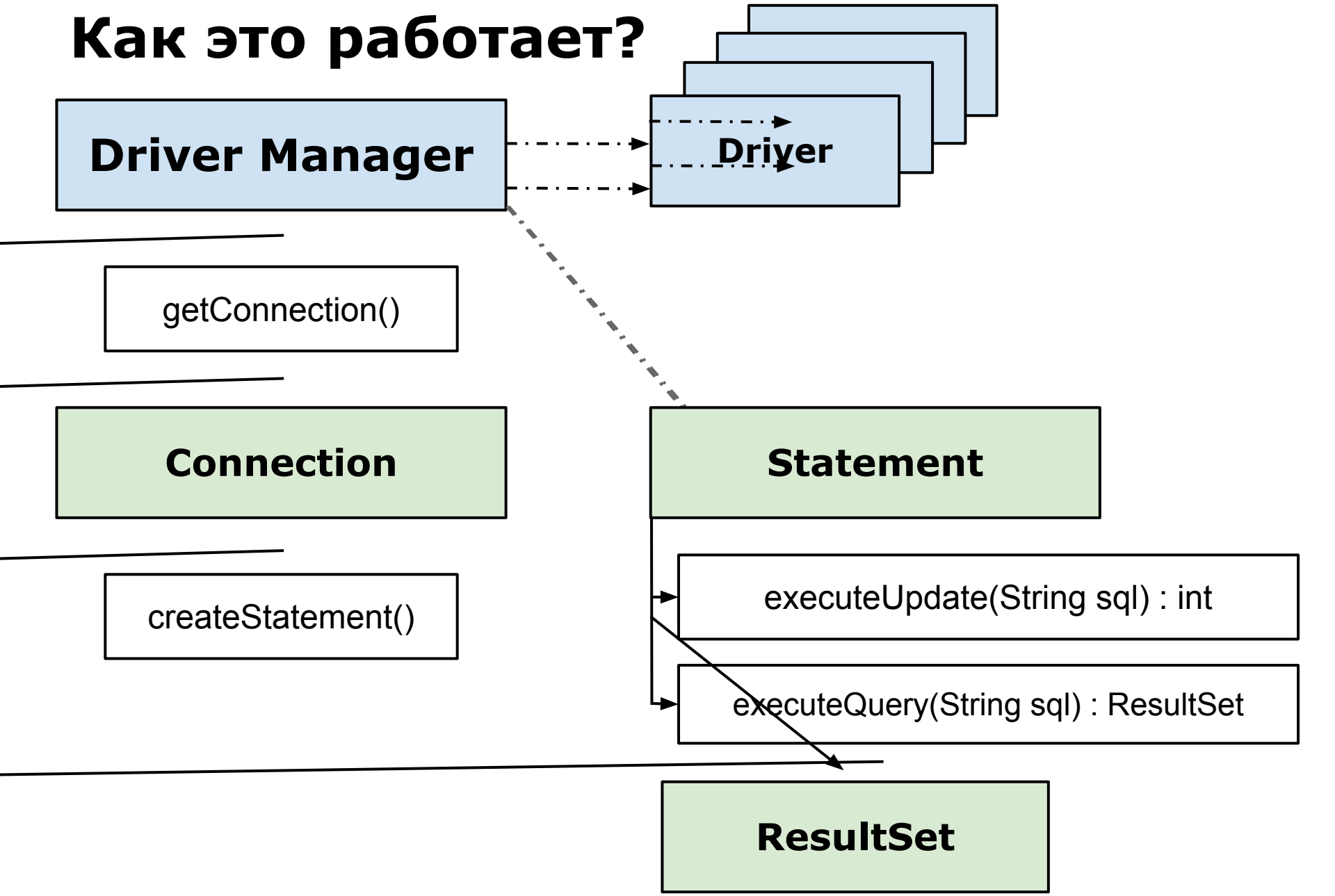
Зачем это всё нужно?



Архитектура JDBC



Как это работает?



Подключение

```
// Шаг 1: Установка параметров подключения
String url = "jdbc:postgresql://127.0.0.1:5432/test"; // *
String name = "user";
String password = "123456";
try {
     // Шаг 2: Выбор драйвера и установка connection
      Class.forName("org.postgresql.Driver"); // Для версий до 4.0
      Connection connection = DriverManager.getConnection(url, name,
                              password);
      Statement statement = connection.createStatement();
      // Шаг 3: Выполняем запрос и получаем ResultSet
      ResultSet rs = statement.executeQuery("SELECT * FROM users WHERE id >
                              2 AND id < 10");
      // Шаг 4: Разбор ResultSet'а
      while (rs.next()) {
System.out.println("Номер в выборке: " + rs.getRow()
     + "Номер в базе: " + rs.getInt("id") + "Имя: " +
rs.getString("firstname"));
}
} catch (Exception ex) {
...
} finally { // Шаг 5: Закрываем connection
     if (connection != null) {
      connection.close(); // * Каскадное закрытие statement, resultSet
      }
}
```

Connection Pool

```
PGPoolingDataSource ds = new PGPoolingDataSource();
ds.setServerName(host);
ds.setDatabaseName(database);
ds.setUser(user);
ds.setPassword(password);
ds.setMaxConnections(100);
ds.setInitialConnections(20);
...
ds.getConnection();
// составление SQL query, запрос, обработка запроса
...
// освобождение Connection-а, а не закрытие
ds.closeConnection();
```

Statement

Statement

```
st.executeQuery(
 SELECT * FROM users
       WHERE id = 2 AND age > 25");
```

PreparedStatement

```
PreparedStatement pst =
     connection.prepareStatement(
"SELECT * FROM users WHERE id = ? AND age > ?");
pst.setLong(1, 2);
pst.setInt(2, 25);
pst.executeQuery();
```

CallableStatement

```
CallableStatement clst = con.prepareCall(
```

ResultSet

```
Sql = "SELECT * FROM users";
```

	id	firstname	lastname	age
0	34	“Иван”	“Иванов”	26
1	35	“Владимир”	“Петров”	47

```
next(); // перевод курсор на следующую строку
previous(); // перевод курсор на предыдущую строку

// получение значения по номеру столбца
getInt(int index)
getString(int index)

// получение значения по названию столбца
getInt(String columnName)
getString(String columnName)
```

Заполнение объекта User

```
class User() { // базовый класс сущности
    long id;
    String firstName;
    String lastName;
    int age;
}
```

```
Statement st = connection.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM users"); //
запрос в БД
List<User> users = new ArrayList<>(); // создание списка
юзеров
while (rs.next()) {
    User user = new User();
    user.id = rs.getLong("id");
    user.firstName = rs.getString("firstname");
    user.lastName = rs.getString("lastname");
    if (!rs.isNull("age")) {
        user.age = rs.getLong("age"); / *
    }
users.add(user); // добавление очередного юзера в список
}
```

Транзакции

Транзакция — группа последовательных операций, которая представляет собой логическую единицу работы с данными.

Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта.

Принципы ACID

Atomicity — транзакции атомарны, то есть либо все изменения транзакции фиксируются (commit), либо все откатываются (rollback);
Consistency — транзакции не нарушают согласованность данных, то есть они переводят базу данных из одного корректного состояния в другое.
Isolation — работающие одновременно транзакции не влияют друг на друга.
Durability — если транзакция была успешно завершена, никакое внешнее событие не должно привести к потере совершенных ей изменений.

Пример работы с транзакциями

```
    // Отключаем режим автоматического коммита
    connection.setAutoCommit(false);

    try {
st.executeUpdate(
"UPDATE users SET age = 25 WHERE id = 8");
st.execute(
"INSERT INTO users (firstname,lastname,age) " +
+ "VALUES ('Петр','Сидоров', 29) ");

connection.commit(); // фиксируем транзакцию
} catch (SQLException e) {
    // Если что-то пошло не так, откатываем всю транзакцию
connection.rollback();
}
```

Deadlocks

```
// Транзакция 1
begin;
select * from A where id = 1;
select * from A where id = 2;
end;
```

```
// Транзакция 2
begin;
select * from A where id = 2;
select * from A where id = 1;
end;
```

```
// Транзакция 1
begin;
select * from A where id = 1;
select * from A where id = 2;
end;
```

```
// Транзакция 2
begin;
select * from A where id = 1;
select * from A where id = 2;
end;
```

Вопросы?