

Работа с источниками данных



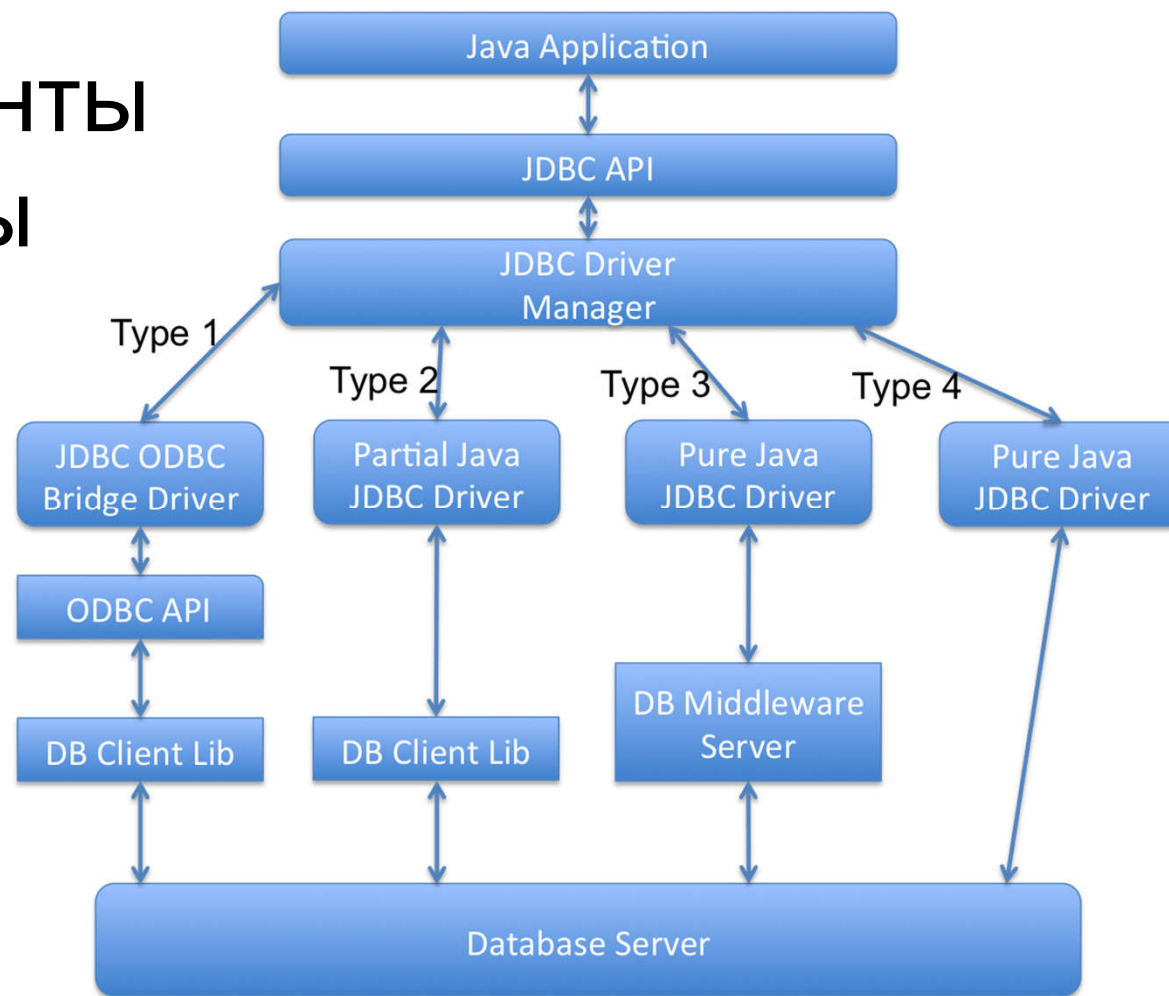
протокол JDBC, коннекторы,...



JDBC - Java Database Connectivity

платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета `java.sql`, входящего в состав Java SE.

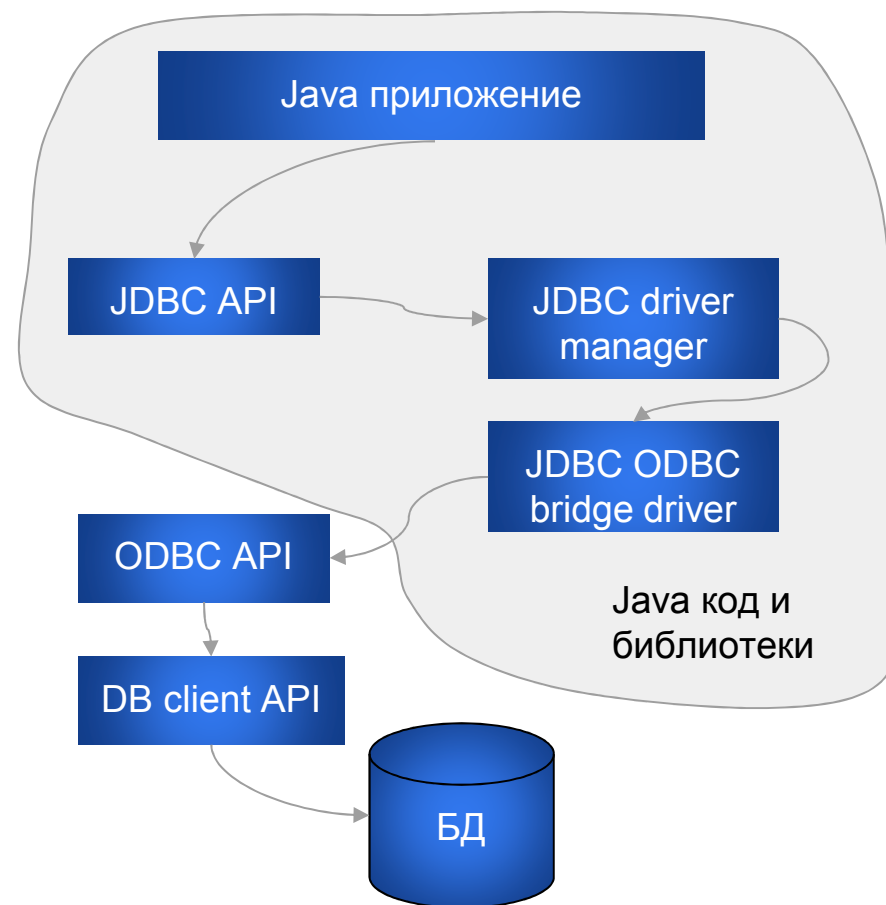
Варианты работы



Детали: <https://javarevisited.blogspot.com/2012/05/different-types-of-jdbc-drivers-in-java.html>

Базовые компоненты для доступа к БД

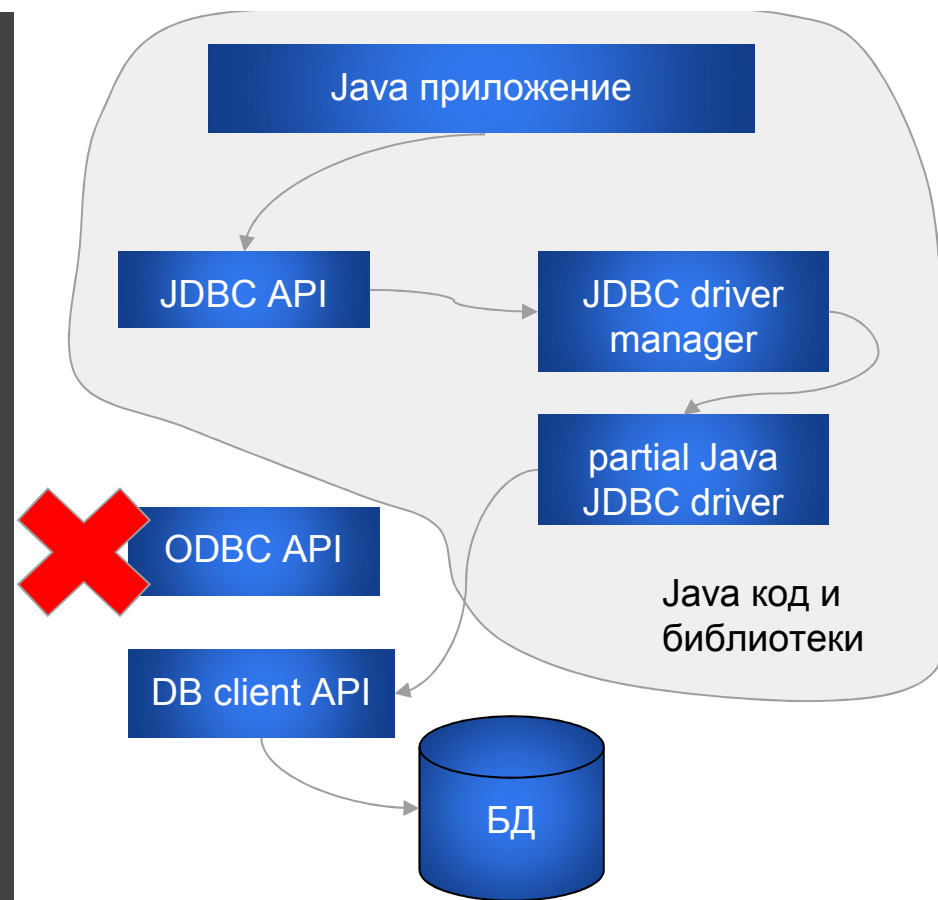
Type 1



ODBC API - драйвер доступа к БД,
предоставленный СУБД для ОС

Базовые компоненты для доступа к БД

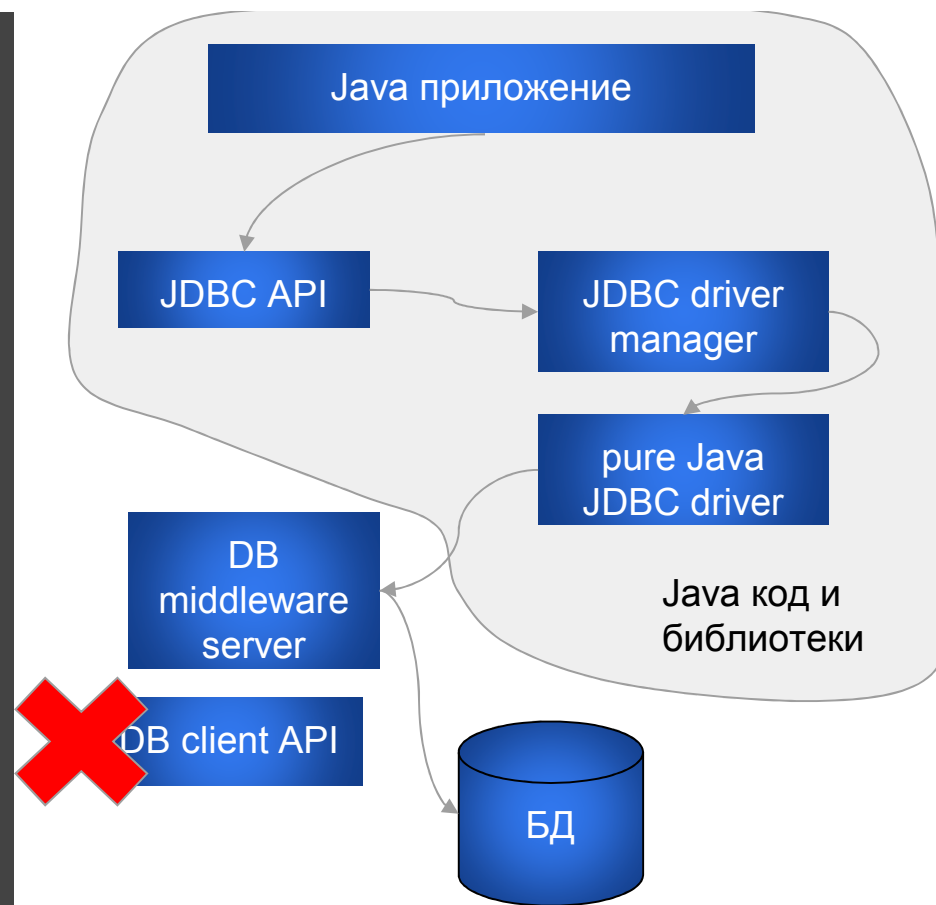
Type 2



ODBC API - драйвер доступа к БД,
предоставленный СУБД для ОС

Базовые компоненты для доступа к БД

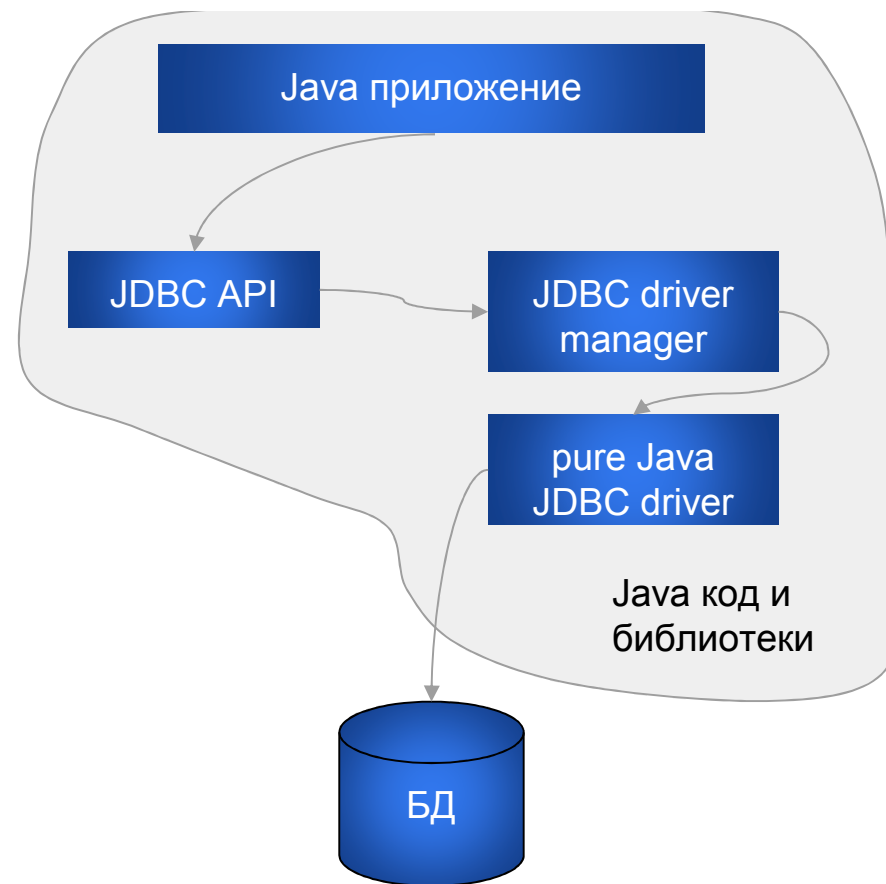
Type 3



DB middleware server - для многозвенных приложений (минимум 3хзвенных)

Базовые компоненты для доступа к БД

Type 4



Ничего больше не нужно - только драйвер и база данных. Считается лучшим драйвером, из-за портбельности и скорости работы

Ссылки на драйвера

MySQL: <https://dev.mysql.com/downloads/connector/j/>

PostgreSQL: <http://jdbc.postgresql.org/download.html>

Драйвер нужно скачать и установить,
а затем добавить путь к jar-файлу в переменную окружения CLASSPATH.

Путь к драйверу можно прописать и из Java проекта.

Элементы JDBC, доступные из Java

Все основные элементы в JDBC API реализованы как интерфейсы в пакете java.sql:

- **Driver** (java.sql.Driver)
- **Connection** (java.sql.Connection)
- **Statement** (java.sql.Statement)
 - a. **PreparedStatement** (java.sql.PreparedStatement)
 - b. **CallableStatement** (java.sql.CallableStatement)
- **ResultSet** (java.sql.ResultSet)
- **DatabaseMetaData** (java.sql.DatabaseMetaData)

Интерфейс Driver

Driver

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
// не импортируйте весь com.mysql.jdbc.*, нужен всего 1 класс

public class LoadDriver {
    public static void main(String[] args) {
        try {
            // в DriverManager регистрируется драйвер для MySQL
            Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception ex) {
            System.out.println("Драйвер для mysql не найден!")
        }
    }
}
```

В нашем случае, класс `com.mysql.jdbc.Driver` реализует интерфейс `java.sql.Driver`

DriverManager

DriverManager

Базовый класс для управления драйверами подключения к базам данных.

В классе DriverManager определены **статические** методы: [getConnection](#), [getDriver](#), посмотреть полный список здесь:

<https://docs.oracle.com/javase/7/docs/api/java/sql/DriverManager.html>

Интерфейс Connection

Connection

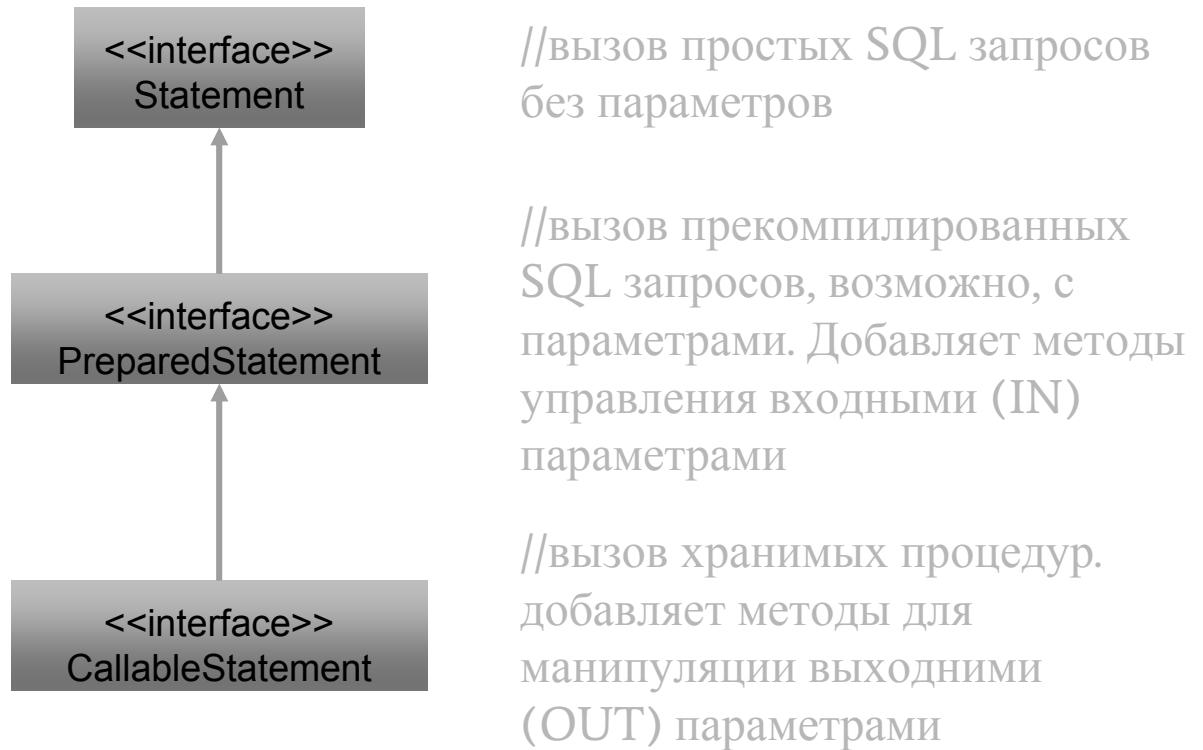
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

Connection conn = null;
//...
try {
    conn =
    DriverManager.getConnection("jdbc:mysql://localhost/mpes?" +
                                "user=user39&password=user39");
    // что-то делаем с соединением - запросы, например, посылаем
    conn.close(); //закрыли соединение ...
} catch (SQLException ex) {
    // сообщение об ошибках
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
```


Интерфейс Statement

Statement

Интерфейс Statement предоставляет базовые методы для выполнения запросов и извлечения результатов.



Statement

Интерфейс Statement предоставляет три различных метода выполнения SQL-выражений: `executeQuery`, `executeUpdate` и `execute`, в зависимости от SQL-запроса.

`executeQuery` - для запросов, результатом которых является один единственный набор значений (`SELECT`).

`executeUpdate` - для операторов `INSERT`, `UPDATE` или `DELETE`, а также для операторов DDL, например, `CREATE TABLE` и `DROP TABLE`.

`execute` используется, когда операторы SQL возвращают более одного набора данных, более одного счетчика обновлений или и то, и другое.

<https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html>

Statement (1)

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

Connection conn = null;
//...
try {
    conn =
    DriverManager.getConnection("jdbc:mysql://localhost/mpes?" +
                                "user=user39&password=user39");
    // что-то делаем с соединением - запросы, например, посылаем
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM
    table");

    conn.close(); //закрыли соединение ...
} catch (SQLException ex) {
    // сообщение об ошибках
```

Statement (2)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
Connection conn = null;
//...
try { conn =
DriverManager.getConnection("jdbc:mysql://localhost/mpes?" +
                             "user=user39&password=user39");
// что-то делаем с соединением - запросы, например, посылаем
Statement stmt = conn.createStatement();
ResultSet rs = stmt.execute("DELETE FROM table
                              WHERE c=0");
conn.close(); //закрыли соединение ...
} catch (SQLException ex) {
    // сообщение об ошибках
}
```

table		
String a	String b	Int c

Интерфейс ResultSet

ResultSet

Интерфейс ResultSet предоставляет результат обращения к базе данных, и содержит:

- методы для обращения к конкретной строке результата (используется понятие курсора)
`first(), last(), next(), previous(), absolute(), afterLast(), beforeFirst()`
- методы-getters - для получения значения в конкретном столбце строки таблицы результата
`getLong(), getBoolean, getInt()...`
- методы-setters (точнее, updaters) - для изменения значения в конкретном столбце строки таблицы результата
`updateRow(), updateLong(), updateBoolean(), updateInt()...`

ResultSet

```
try {  
    //создаем соединение conn  
    Connection conn = DriverManager.getConnection(...);  
    // что-то делаем с соединением - запросы, например, посылаем  
    Statement stmt = conn.createStatement();  
    ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM table");  
    while (rs.next()) {  
        System.out.println(rs.getString("a"));  
    }  
    rs.absolute(2);  
    System.out.println(rs.getString("b")); //see the value of field b, row #2  
    rs.updateString("b", "new value of field _b_ at row #2");  
    rs.updateRow(); //обновили строку в базе данных  
    conn.close(); //закрыли соединение  
} catch (SQLException ex) {  
    // сообщение об ошибках  
}
```

table		
String a	String b	Int c

<https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html>

Интерфейс DatabaseMetaData

DatabaseMetaData

Интерфейс DatabaseMetaData предоставляет сведения о том, что можно делать в конкретной СУБД и с конкретным драйвером JDBC:

- методы для выяснения версии СУБД и драйвера
`getDriverName()`, `getDriverVersion()`, `getDatabaseProductName()`,...
- методы для выяснения значений переменных, с которыми запущена СУБД (максимальные размеры полей, системные таблицы, ...)

<https://docs.oracle.com/javase/7/docs/api/java/sql/DatabaseMetaData.html>