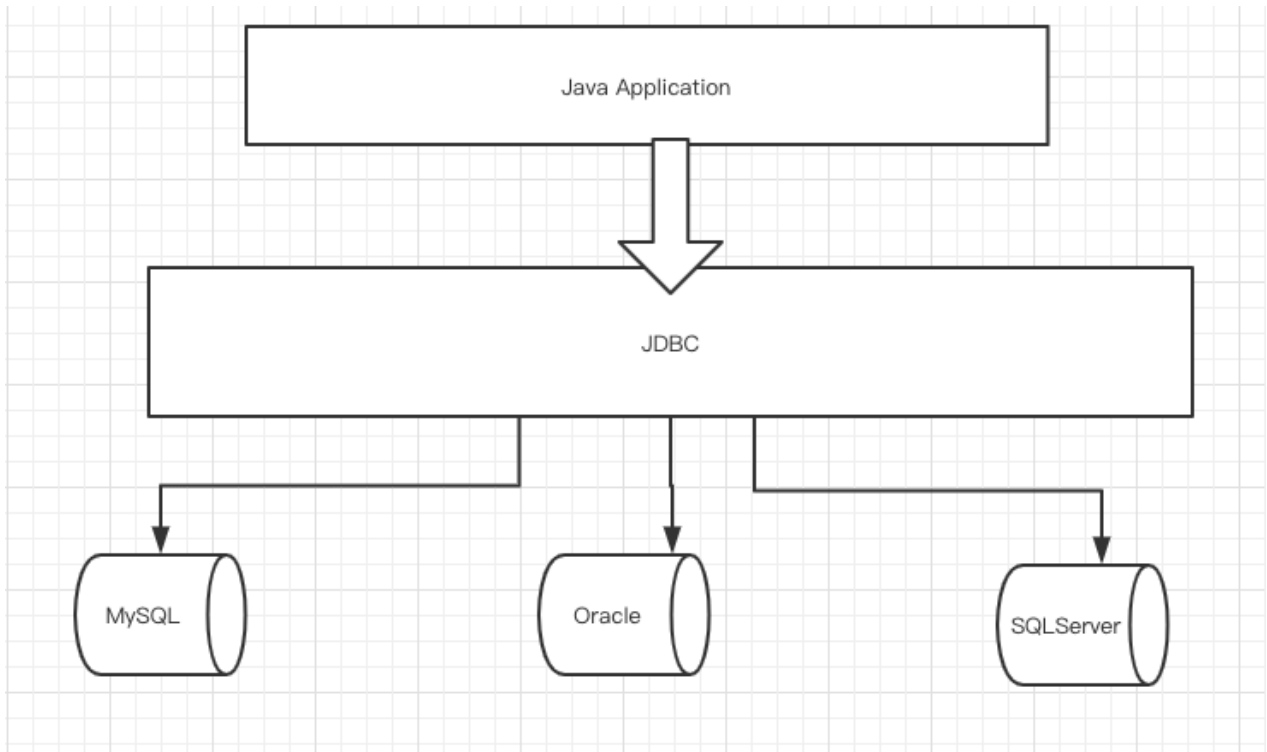


JDBC

Java DataBase Connectivity 是一个独立于特定数据库的管理系统，通用的 SQL 数据库存取和操作的公共接口。

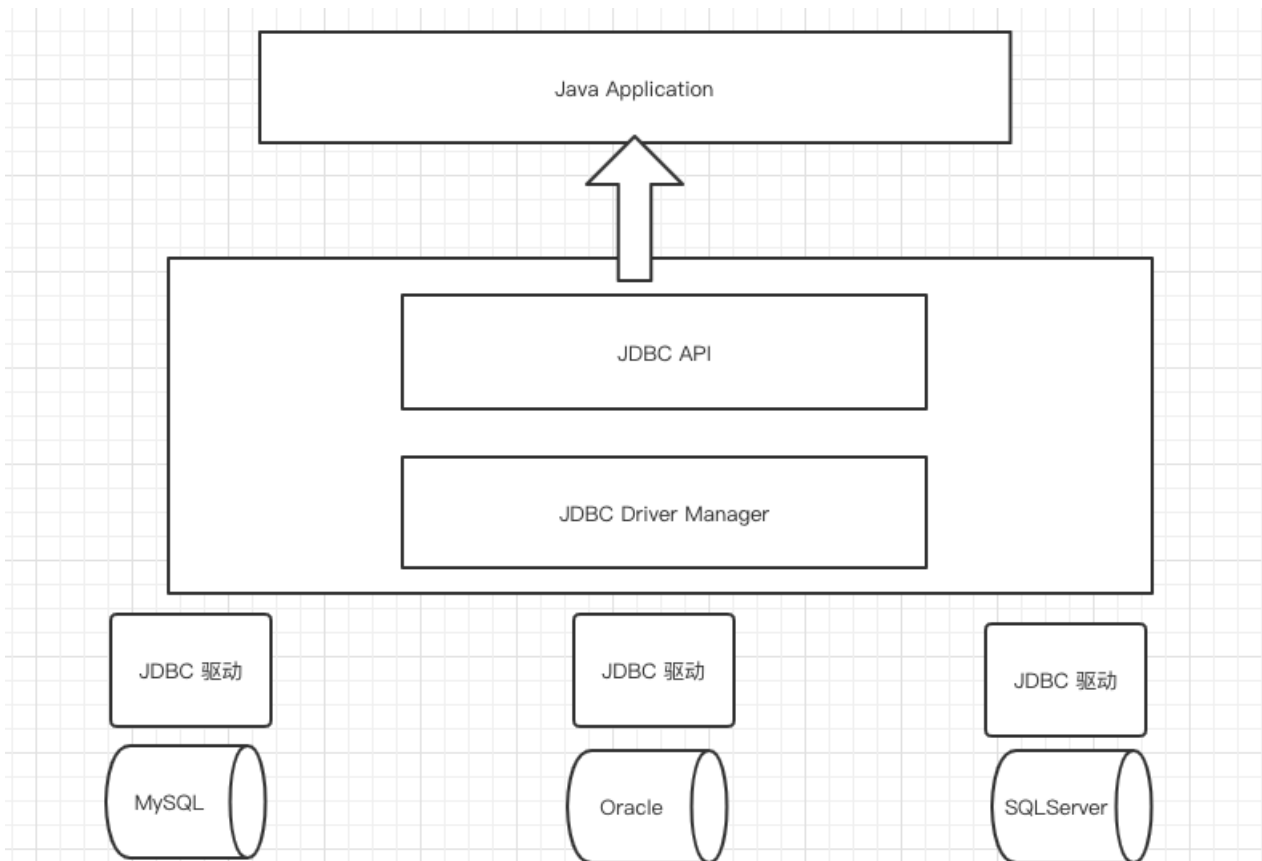
定义了一组标准，为访问不同数据库提供了统一的途径。



JDBC 体系结构

JDBC 接口包括两个层面：

- 面向应用的 API，供程序员调用
- 面向数据库的 API，供厂商开发数据库的驱动程序



JDBC API

提供者：Java 官方

内容：供开发者调用的接口

java.sql 和 javax.sql

- DriverManager 类
- Connection 接口
- Statement 接口
- ResultSet 接口

DriverManager

提供者：Java 官方

作用：管理不同的 JDBC 驱动

JDBC 驱动

提供者：数据库厂商

作用：负责连接不同的数据库

JDBC 的使用

- 1、加载数据库驱动，Java 程序和数据库之间的桥梁。
- 2、获取 Connection，Java 程序与数据库的一次连接。

3、创建 Statement 对象，由 Connection 产生，执行 SQL 语句。

4、如果需要接收返回值，创建 ResultSet 对象，保存 Statement 执行之后所查询到的结果。

```
package com.southwind.test;

import java.sql.*;
import java.util.Date;

public class Test {
    public static void main(String[] args) {
        try {
            //加载驱动
            Class.forName("com.mysql.cj.jdbc.Driver");
            //获取连接
            String url = "jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8";
            String user = "root";
            String password = "root";
            Connection connection =
DriverManager.getConnection(url,user,password);
            //      String sql = "insert into student(name,score,birthday) values('李
四',78,'2019-01-01')";
            //      String sql = "update student set name = '李四'";
            //      String sql = "delete from student";
            //      Statement statement = connection.createStatement();
            //      int result = statement.executeUpdate(sql);

            String sql = "select * from student";
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            while (resultSet.next()){
                Integer id = resultSet.getInt("id");
                String name = resultSet.getString(2);
                Double score = resultSet.getDouble(3);
                Date date = resultSet.getDate(4);
                System.out.println(id+"-"+name+"-"+score+"-"+date);
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e){
            e.printStackTrace();
        }
    }
}
```

PreparedStatement

Statement 的子类，提供了 SQL 占位符的功能

使用 Statement 进行开发有两个问题：

- 1、需要频繁拼接 String 字符串，出错率较高。
- 2、存在 SQL 注入的风险。

SQL 注入：利用某些系统没有对用户输入的信息进行充分检测，在用户输入的数据中注入非法的 SQL 语句，从而利用系统的 SQL 引擎完成恶意行为的做法。

```
String url = "jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8";
String user = "root";
String password = "root";
Connection connection = DriverManager.getConnection(url,user,password);
String username = "lisi";
String mypassword = "000";
String sql = "select * from t_user where username = ? and password = ?";
System.out.println(sql);
PreparedStatement preparedStatement = connection.prepareStatement(sql);
preparedStatement.setString(1,username);
preparedStatement.setString(2,mypassword);
ResultSet resultSet = preparedStatement.executeQuery();
if(resultSet.next()){
    System.out.println("登录成功");
}else{
    System.out.println("登录失败");
}
} catch (ClassNotFoundException e) {
    e.printStackTrace();
} catch (SQLException e){
    e.printStackTrace();
}
```