MySQL数据库

1.MySQL常用操作

1.MySQL连接

```
mysql -u root -p
Enter password:
```

2.MySQL创建数据库

```
create database stu;
```

3.MySQL创建学生表

```
CREATE TABLE table_name (column_name column_type);
```

4.MySQL插入数据

```
INSERT INTO table_name ( field1, field2,...fieldN )
                       VALUES
                       ( value1, value2,...valueN );
```

如果数据是字符型，必须使用单引号或者双引号，如："value"。

5.MySQL查询数据

```
SELECT column_name,column_name
FROM table_name
[WHERE Clause]
[OFFSET M ][LIMIT N]
```

6.MySQL where 子句

我们知道从MySQL表中使用SQL SELECT 语句来读取数据。

如需有条件地从表中选取数据，可将 WHERE 子句添加到 SELECT 语句中。

以下是SQL SELECT 语句使用 WHERE 子句从数据表中读取数据的通用语法：

```
SELECT field1, field2,...fieldN FROM table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

- 查询语句中你可以使用一个或者多个表，表之间使用逗号(,)分割，并使用WHERE语句来设定查询条件。
- 你可以在WHERE子句中指定任何条件。
- 你可以使用AND或者OR指定一个或多个条件。

- WHERE子句也可以运用于SQL的 DELETE 或者 UPDATE 命令。
- WHERE 子句类似于程序语言中的if条件，根据 MySQL 表中的字段值来读取指定的数据。

以下为操作符列表，可用于 WHERE 子句中。

下表中实例假定 A为10 B为20

操作符	描述	实例
=	等号，检测两个值是否相等，如果相等返回true	(A = B) 返回 false。
<> 或 !=	不等于，检测两个值是否相等，如果不相等返回true	(A != B) 返回 true。
>	大于号，检测左边的值是否大于右边的值, 如果左边的值大于右边的值返回true	(A > B) 返回 false。
<	小于号，检测左边的值是否小于右边的值, 如果左边的值小于右边的值返回true	(A < B) 返回 true。
>=	大于等于号，检测左边的值是否大于或等于右边的值, 如果左边的值大于或等于右边的值返回true	(A >= B) 返回 false。
<=	小于等于号，检测左边的值是否小于于或等于右边的值, 如果左边的值小于或等于右边的值返回true	(A <= B) 返回 true。

如果我们想再MySQL数据表中读取指定的数据，WHERE 子句是非常有用的。

使用主键来作为 WHERE 子句的条件查询是非常快速的。

如果给定的条件在表中没有任何匹配的记录，那么查询不会返回任何数据。

7.MySQL UPDATE 查询

以下是 UPDATE 命令修改 MySQL 数据表数据的通用SQL语法：

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

可以同时更新一个或多个字段，可以在 WHERE 子句中指定任何条件，可以在一个单独表中同时更新数据。

2.JDBC连接前修改

1.在maven工程中添加mysql-jdbc依赖包

打开File -> Project Structure -> modules -> 右侧栏的 "+" -> JARs or directories.. -> 添加mysql-jdbc -> Apply

2.虚拟机上修改

1. vim /etc/mysql/mysql.conf.d/mysqld.cnf 注释掉 bind-address = 127.0.0.1
2.
    - 进入mysql

- use mysql;
- update user set host = ' %' where user = 'root';
- flush privileges;
- 退出mysql
- service mysql restart

2.JDBC

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。

1.MySQL的连接与关闭

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

public class JDBCUtils {
    private static final String DRIVER="com.mysql.cj.jdbc.Driver";
    private static final String DB = "jdbc:mysql://192.168.6.130:3306/opt";
    private static final String USER = "root";
    private static final String PASSWORD = "root";
    // 通过参数连接数据库
    static {
        try {
            Class.forName(JdbcUtils.DRIVER);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    /**
     * 获取链接对象connection
     * @return
     */
    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(JdbcUtils.DB, JdbcUtils.USER,
JdbcUtils.PASSWORD);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
    /**
     * 关闭资源
     * @param conn
     * @param st
     * @param rs
     */
    public static void close(Connection conn,PreparedStatement ps,ResultSet rs)
{
        if(rs != null) {
            try {
                rs.close();
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        if(rs != null) {
            try {
                st.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        if(conn != null) {
            try {
                conn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

2.MySQL的查询

使用Statement容易产生SQL注入问题，故一般使用PreparedStatement来代替Statement。PreparedStatement是对SQL语句预编译。

```
import org.junit.Test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class mysqlQuery {
   @Test
    public void Query() {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            conn = JDBCUtils.getConnection();
            String sql = "select * from stu";
            ps = conn.prepareStatement(sql);
            rs = ps.executeQuery();
            while (rs.next()) {
                int id = rs.getInt("Id");
                String sex = rs.getString("Sex");
                String name = rs.getString("Name");

                System.out.println("id :" + id);
                System.out.println("sex :" + sex);
                System.out.println("name :" + name);
            }
        }catch (Exception e) {
            e.printStackTrace();
        }finally {
            JDBCUtils.close(conn, ps, rs);
```

```
        }
    }
}
```