**实验报告七**

|  |  |
| --- | --- |
| 实验序号：7 | 实验名称：数据库应用编程 |
| 学号：2023015509 | 实验日期：2025年6月12日 |
| 姓名：胡林森 | 实验教室：C4 420机房 |

**一、实验目的**

1．了解数据库的数据访问接口技术，如：ODBC、JDBC、OLE DB、ADO、ADO.NET等；

2．掌握简易的数据库应用软件的开发。

**二、实验学时**

2学时

**三、实验内容**

1．查阅资料，了解并理解数据库的数据访问接口技术，如：ODBC、JDBC、OLE DB、ADO、ADO.NET等，并形成一个简要的数据访问接口技术总结；

2．选用合适的软件开发工具和数据访问接口开发简易的采油厂油水井作业成本管理系统（建议应用软件可视化），包括如下功能：

⑴ 数据管理模块：提供数据的增加、修改和删除功能。

⑵ 数据查询模块：提供实验三中实验内容2的查询功能。

**四、实验报告**

目 录

[一、实验步骤 3](#_Toc200724581)

[（一）环境准备 3](#_Toc200724582)

[1.技术栈 3](#_Toc200724583)

[2.开发工具 3](#_Toc200724584)

[（二）JDBC安装 3](#_Toc200724585)

[（三）需求分析 3](#_Toc200724586)

[架构图： 4](#_Toc200724587)

[ER图： 4](#_Toc200724588)

[二、实验结果 5](#_Toc200724589)

[（一）可视化页面 5](#_Toc200724590)

[（二）数据管理 6](#_Toc200724591)

[（三）查询操作 6](#_Toc200724592)

[二、实验收获 10](#_Toc200724593)

[三、核心代码 10](#_Toc200724594)

[JDBCUtil.java部分代码 10](#_Toc200724595)

[QueryManager.java部分代码 10](#_Toc200724596)

[DataManager.java部分代码 12](#_Toc200724597)

# 一、实验步骤

# （一）环境准备

## 1.技术栈

**编程语言：**Java

**JDK:** 22

**界面框架：**Java Swing

**构建工具：**Maven

## 2.开发工具

**IDE:** IntelliJ IDEA

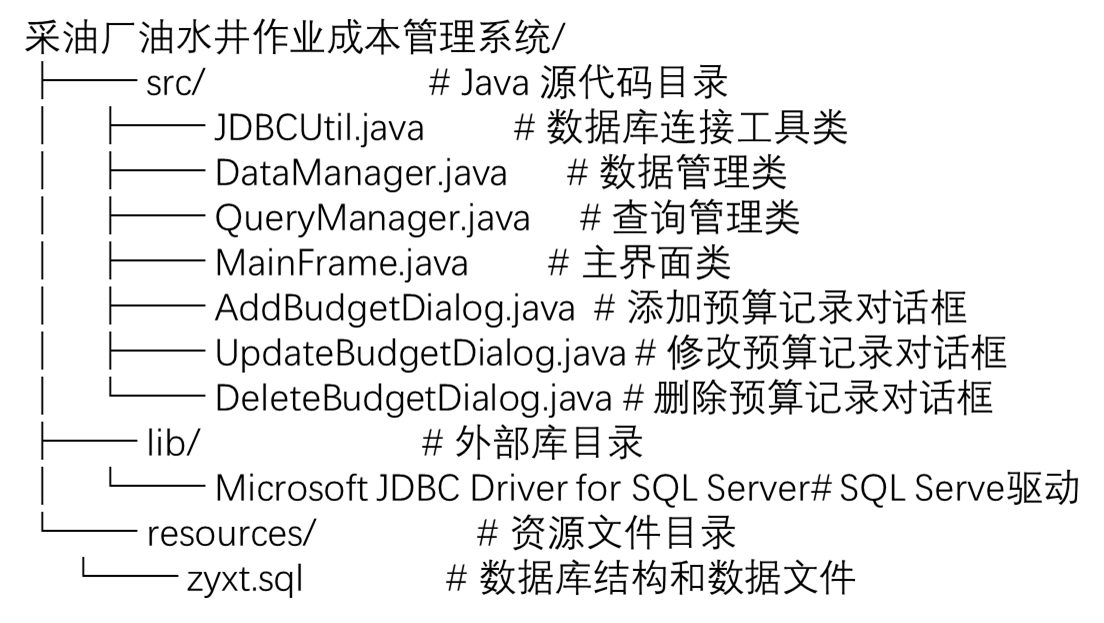
# （二）JDBC安装

从官网下载对应版本Microsoft JDBC Driver for SQL Server。

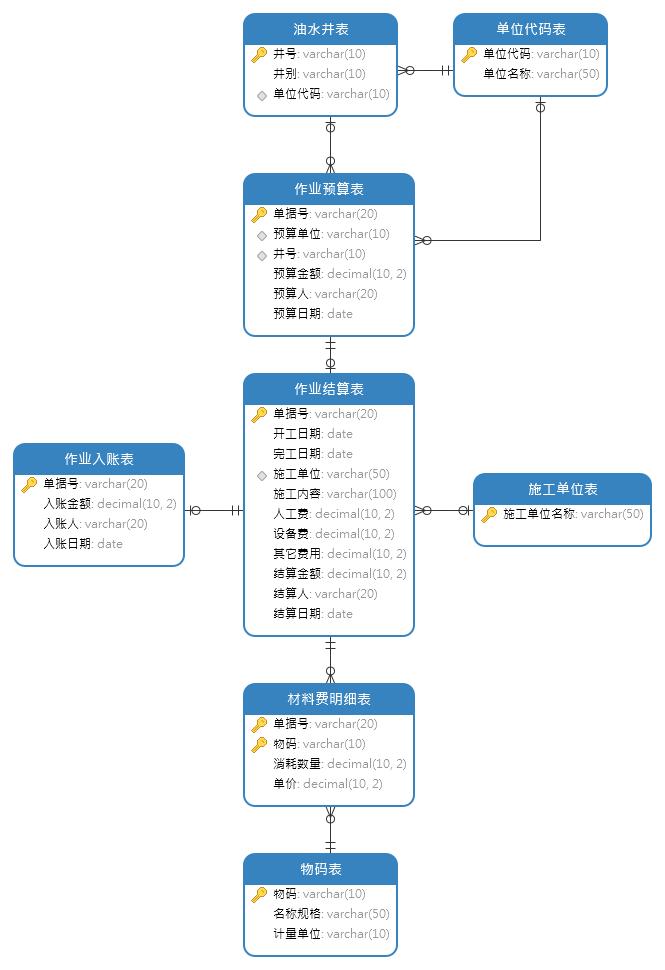
# （三）需求分析

* **JDBCUtil.java**
  + **内容**：数据库连接工具类，提供数据库连接获取和关闭的方法。
  + **作用**：管理与 MySQL 数据库的连接，为其他模块提供数据库访问的基础功能。
* **DataManager.java**
  + **内容**：数据管理类，实现数据的增加、修改和删除功能。
  + **作用**：提供对作业预算表等数据的增删改操作，支持系统的数据管理模块。
* **QueryManager.java**
  + **内容**：查询管理类，实现各种查询功能。
  + **作用**：封装了所有查询逻辑，包括简单的数据查询和复杂的统计、汇总查询。
* **MainFrame.java**
  + **内容**：系统的主界面类，使用 Swing 构建图形用户界面。
  + **作用**：提供系统的可视化操作界面，包括菜单栏和选项卡，整合各功能模块。
* **AddBudgetDialog.java**
  + **内容**：添加预算记录的对话框类。
  + **作用**：提供用户输入界面，用于添加新的预算记录。
* **UpdateBudgetDialog.java**
  + **内容**：修改预算记录的对话框类。
  + **作用**：提供用户输入界面，用于修改现有预算记录。
* **DeleteBudgetDialog.java**
  + **内容**：删除预算记录的对话框类。
  + **作用：**提供用户输入界面，用于删除预算记录。

## 架构图：

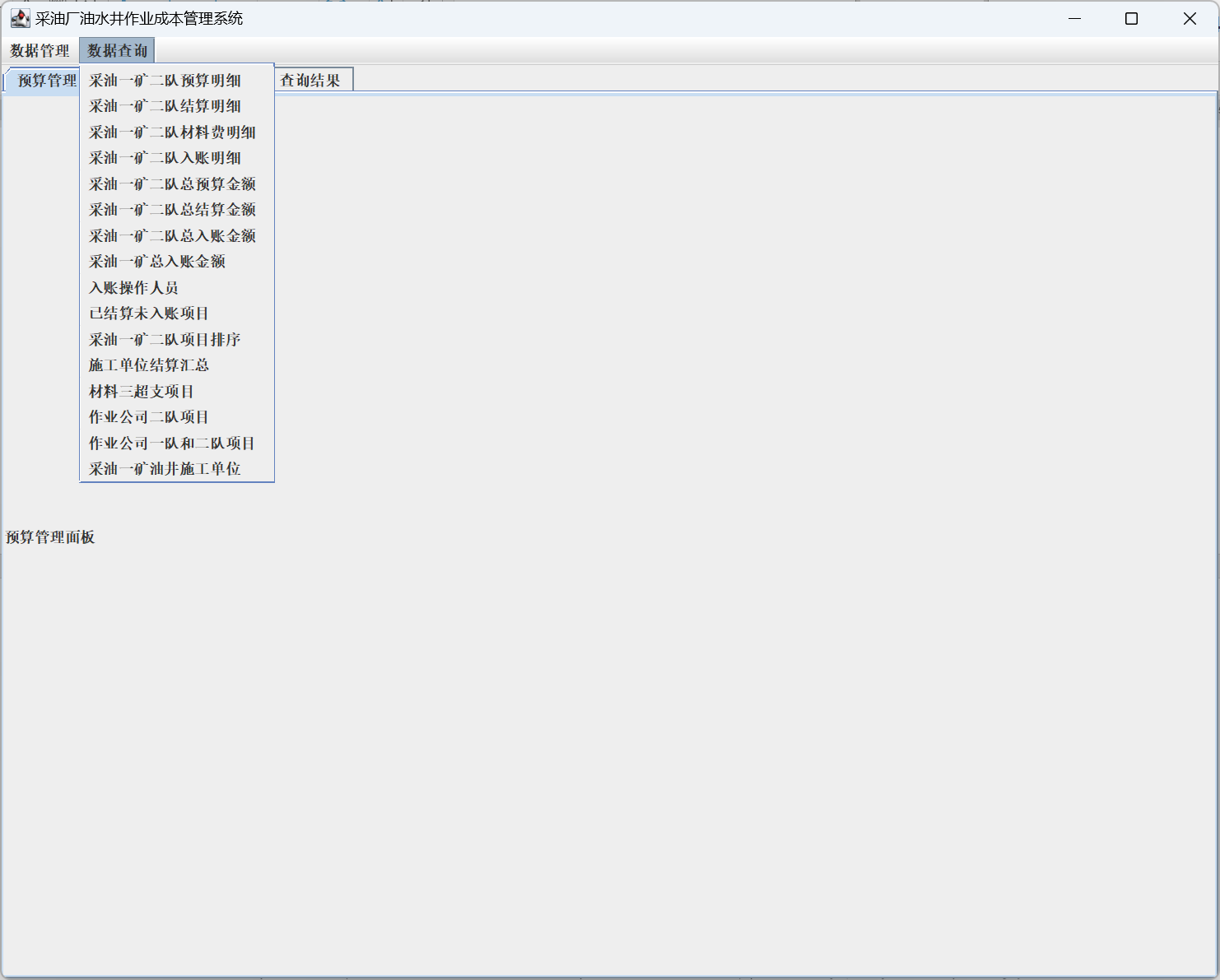


## ER图：



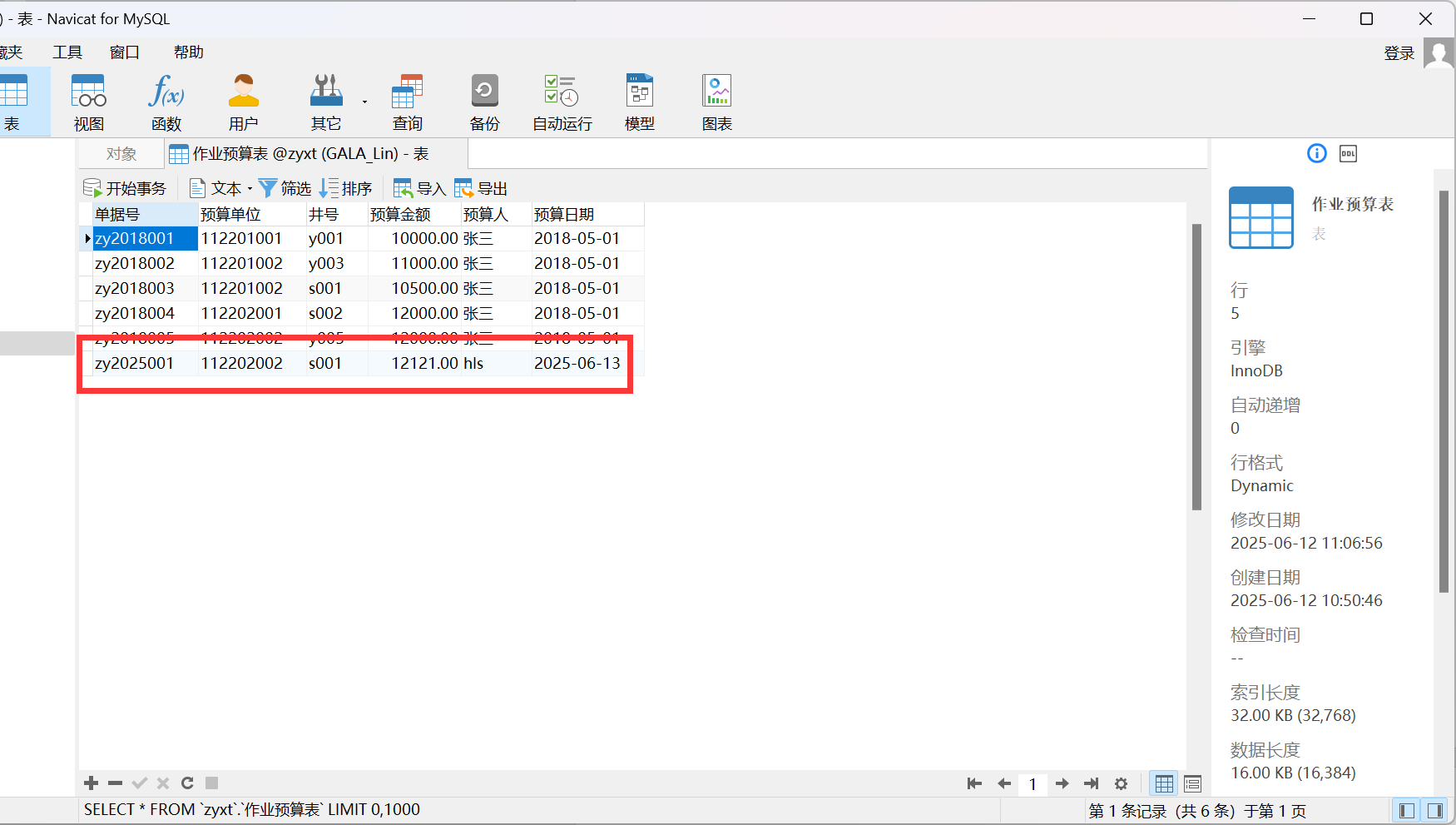
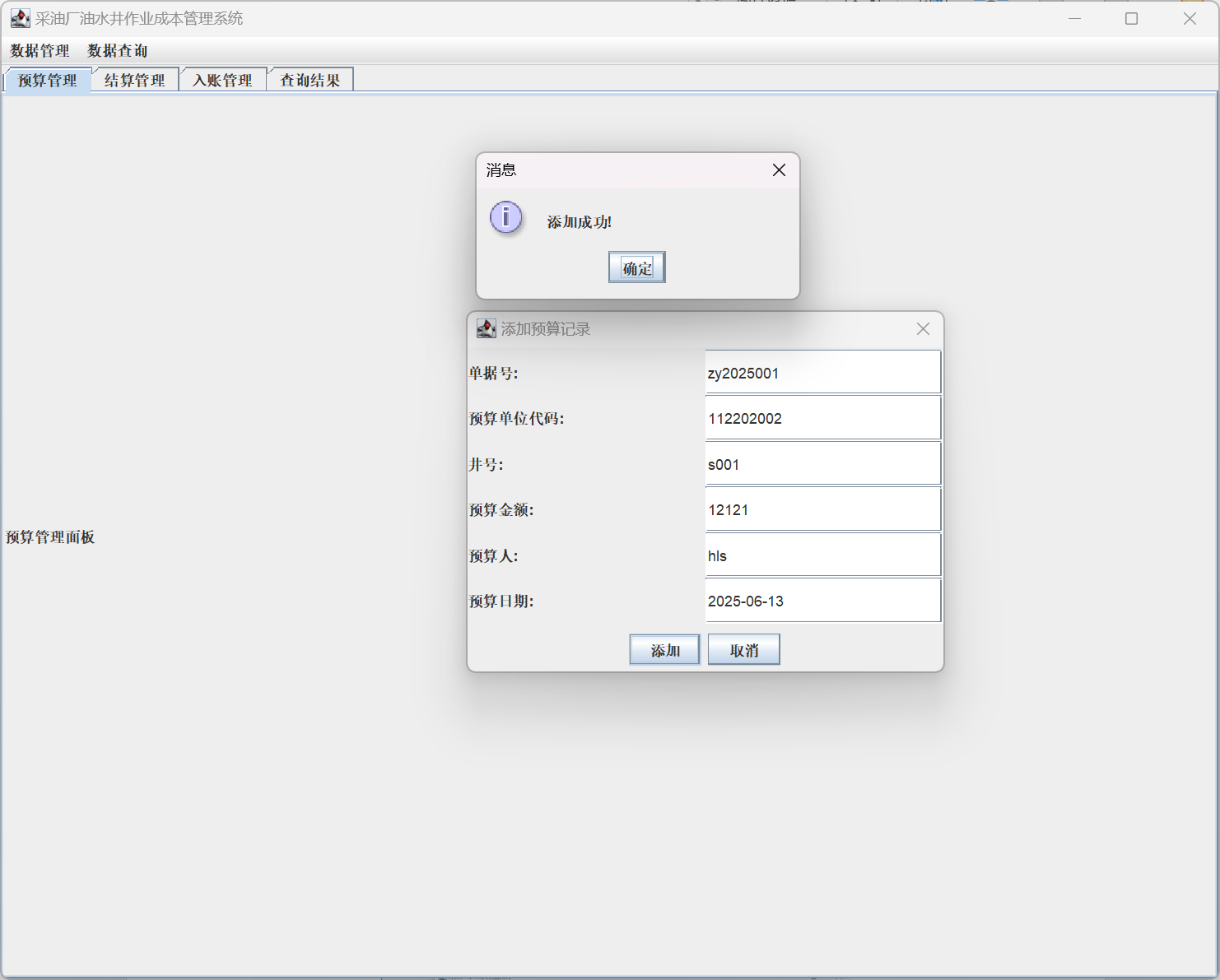
# 二、实验结果

# （一）可视化页面

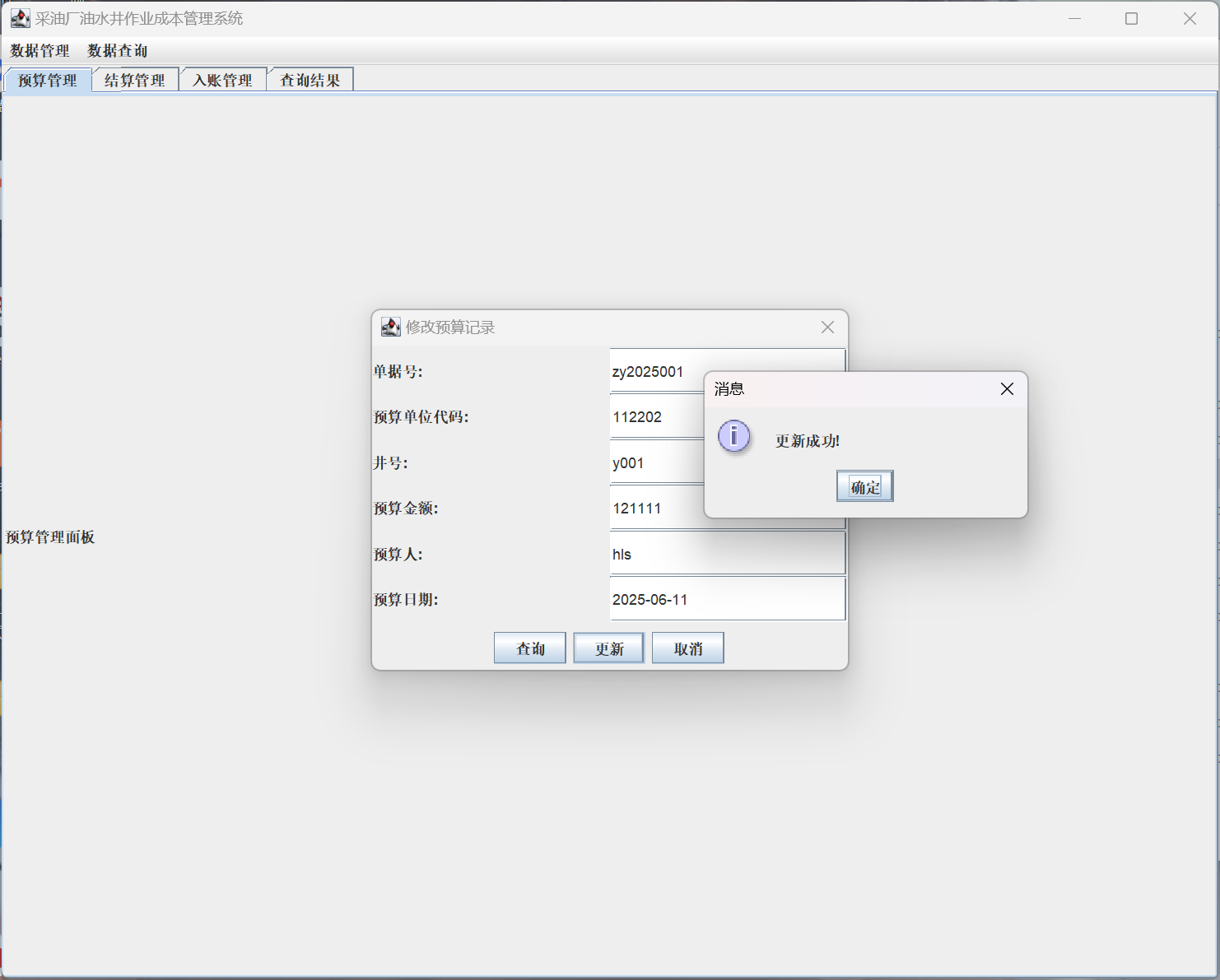


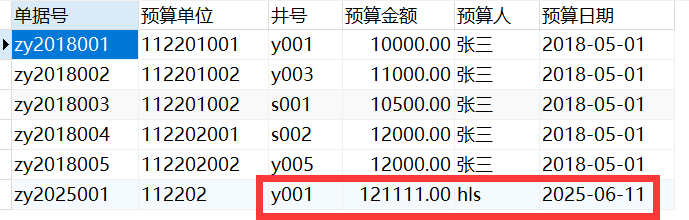
# （二）数据管理

### ⑴增

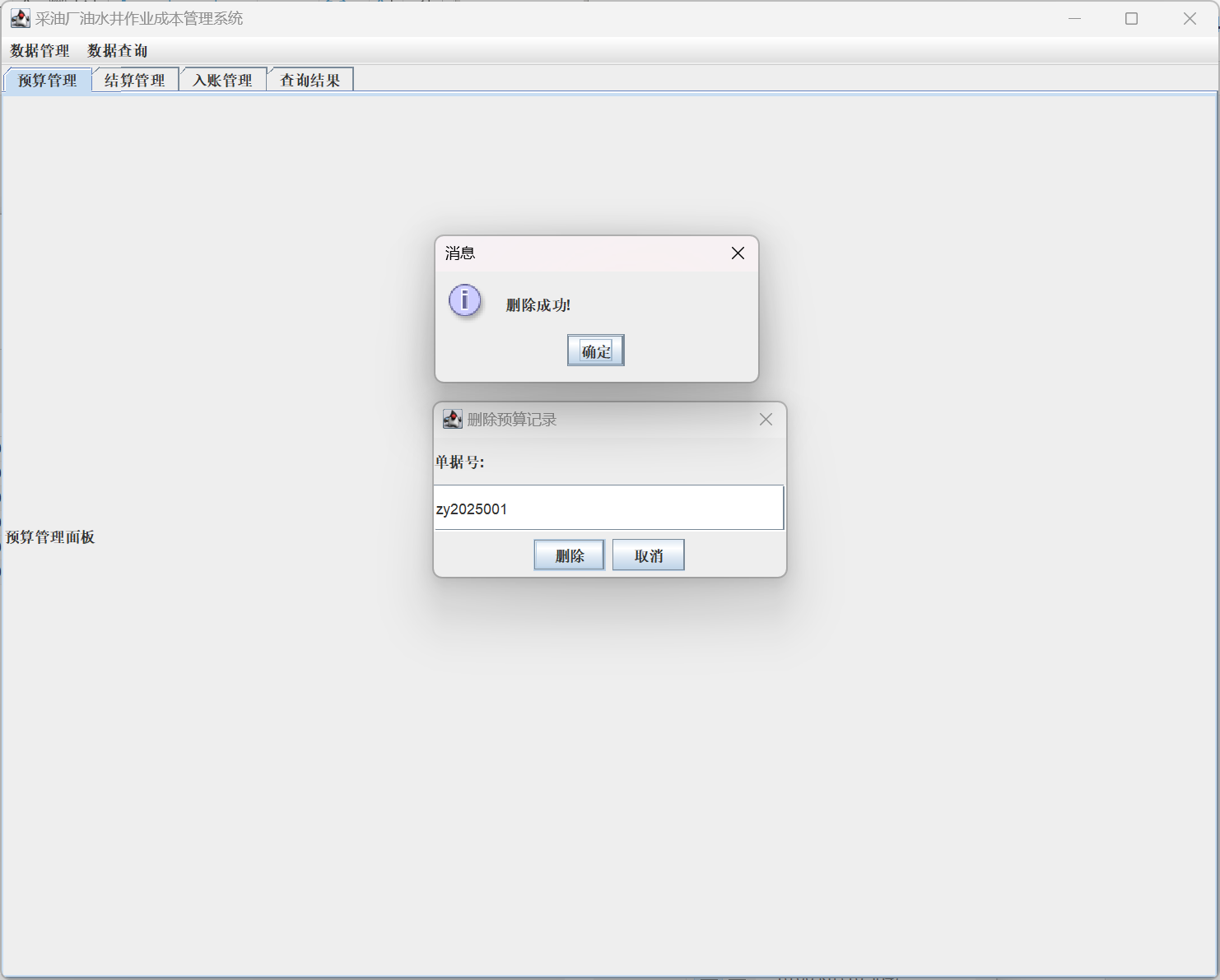


### ⑵改





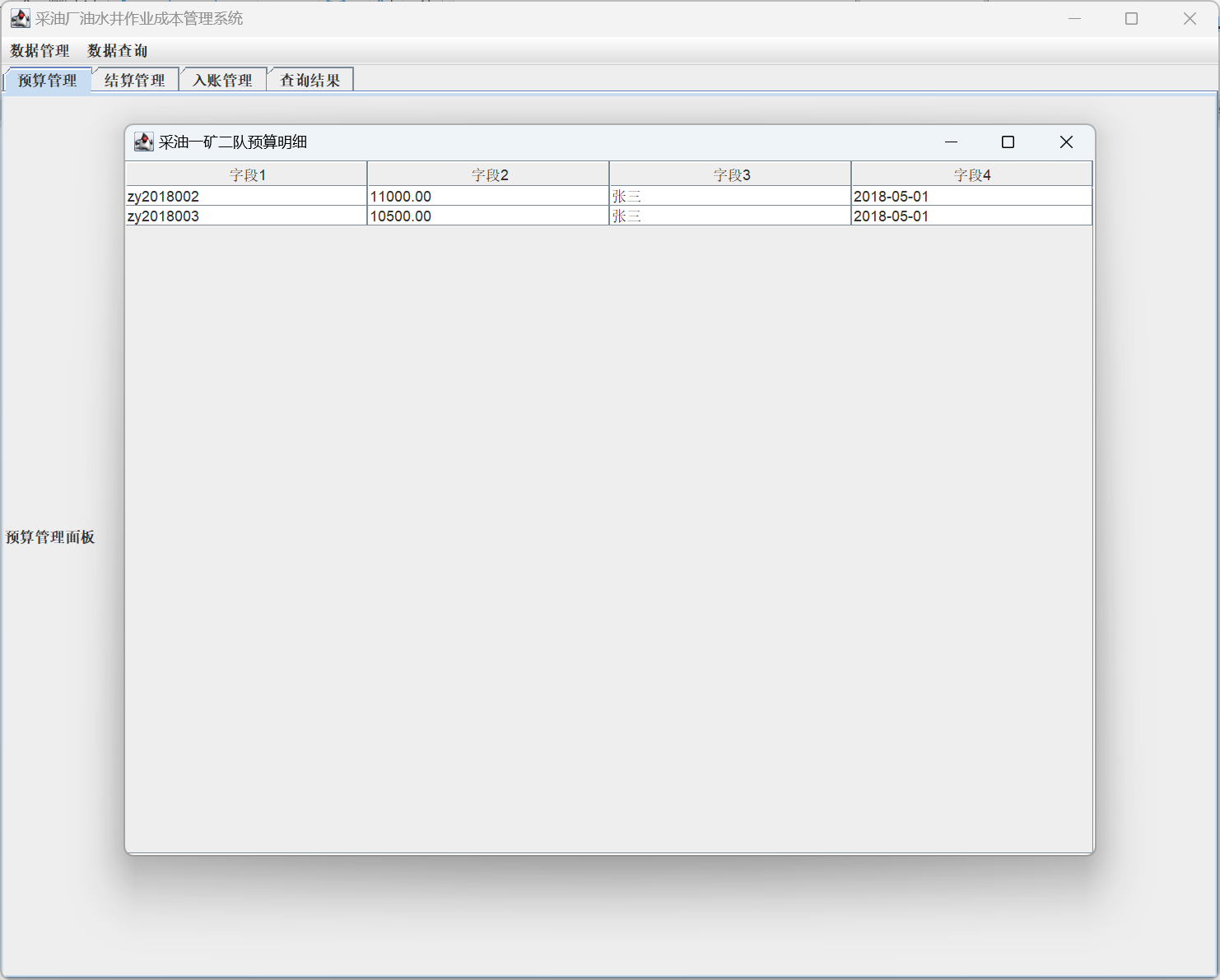
### ⑶删



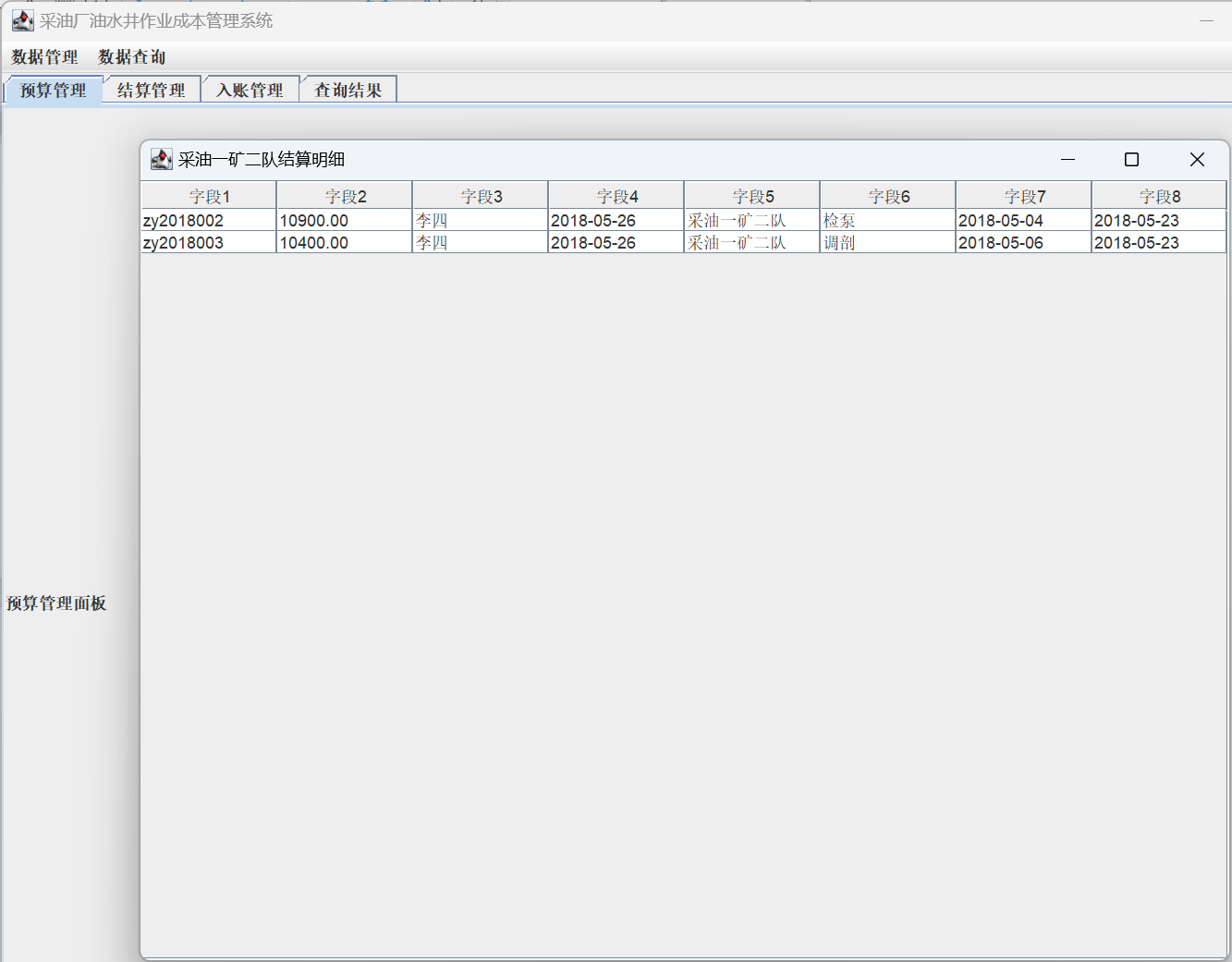


# （三）查询操作

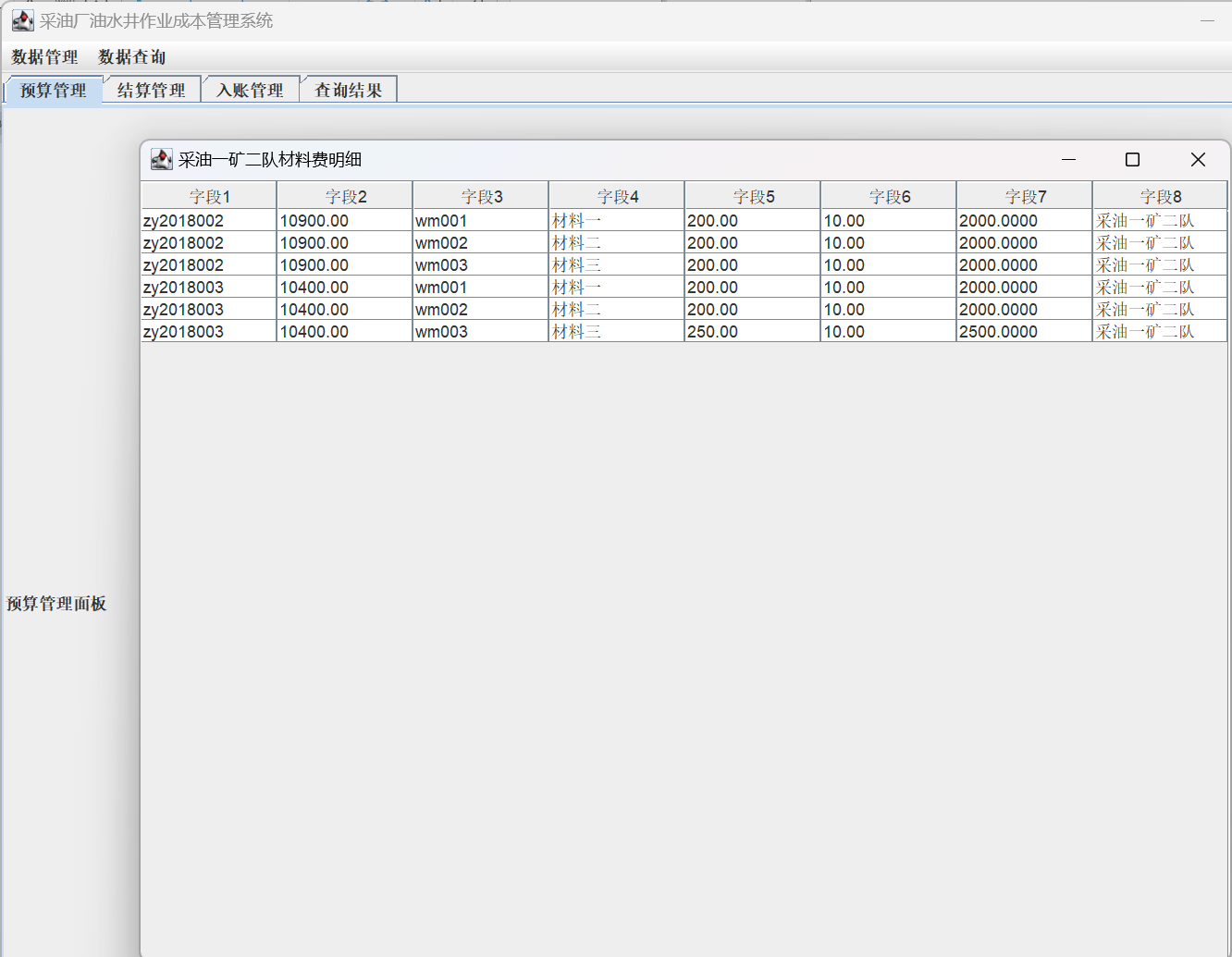
### ⑴ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了预算，列出相应明细。



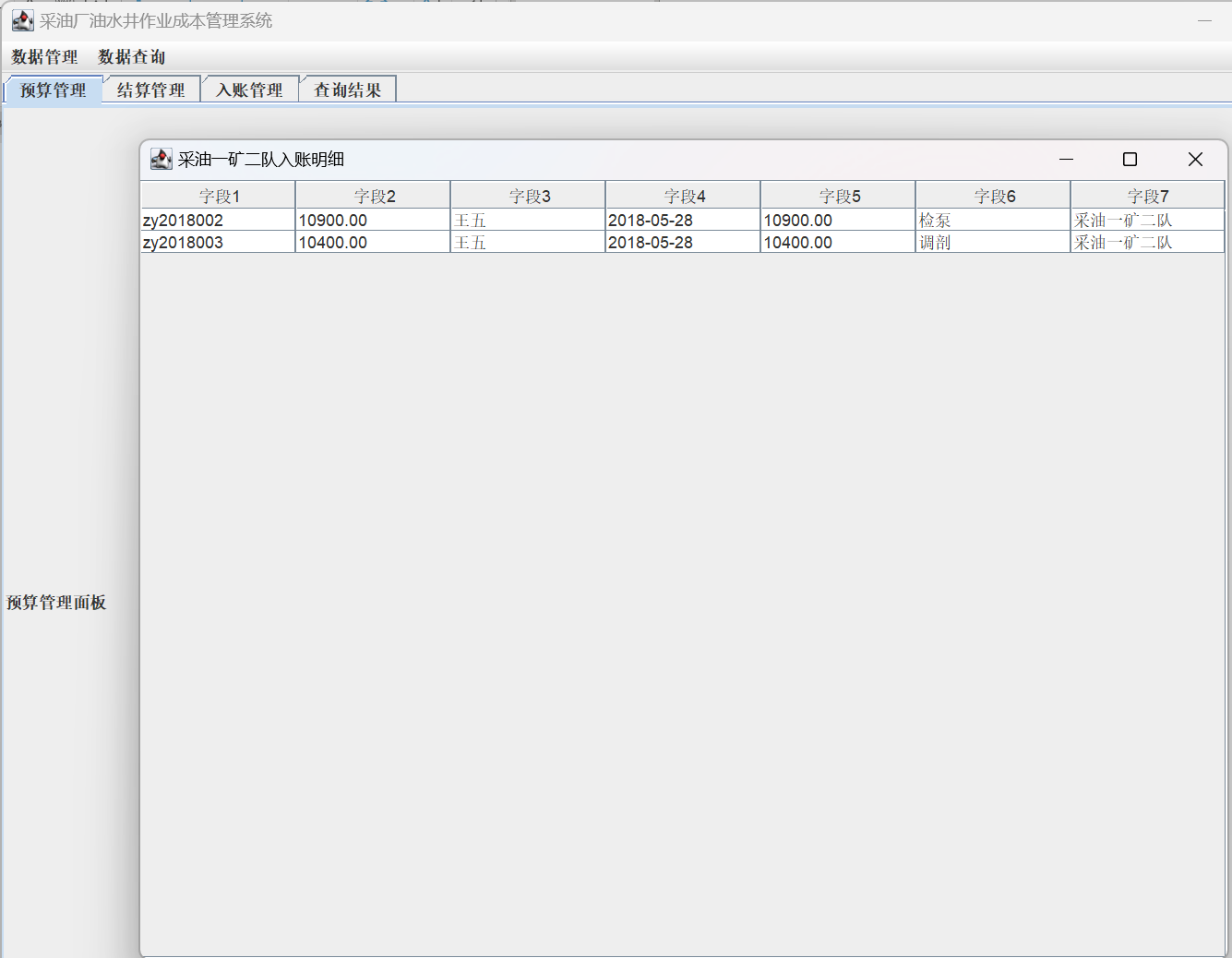
### ⑵ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了结算，列出相应明细。



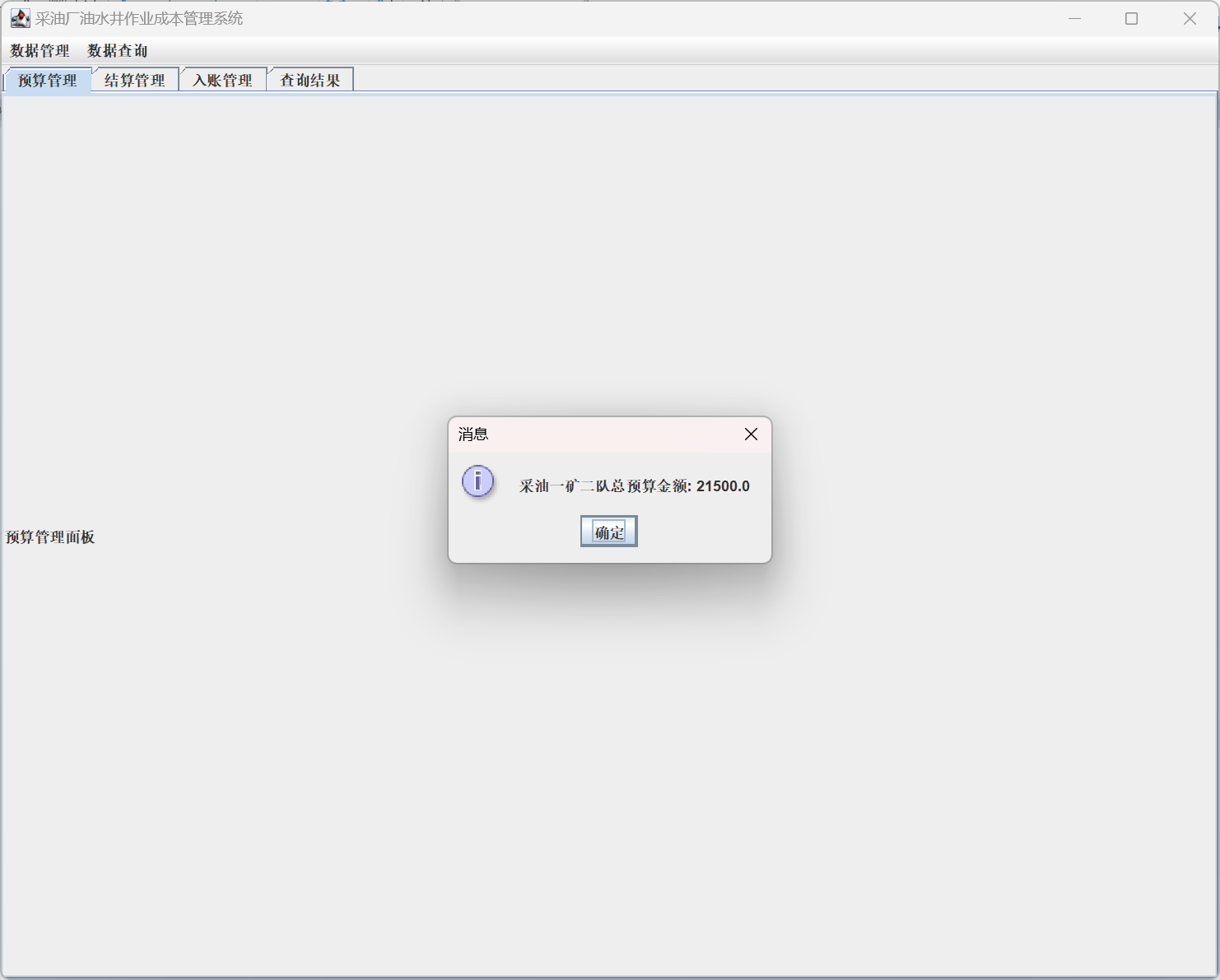
### ⑶ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了结算，列出相应的材料费消耗明细。



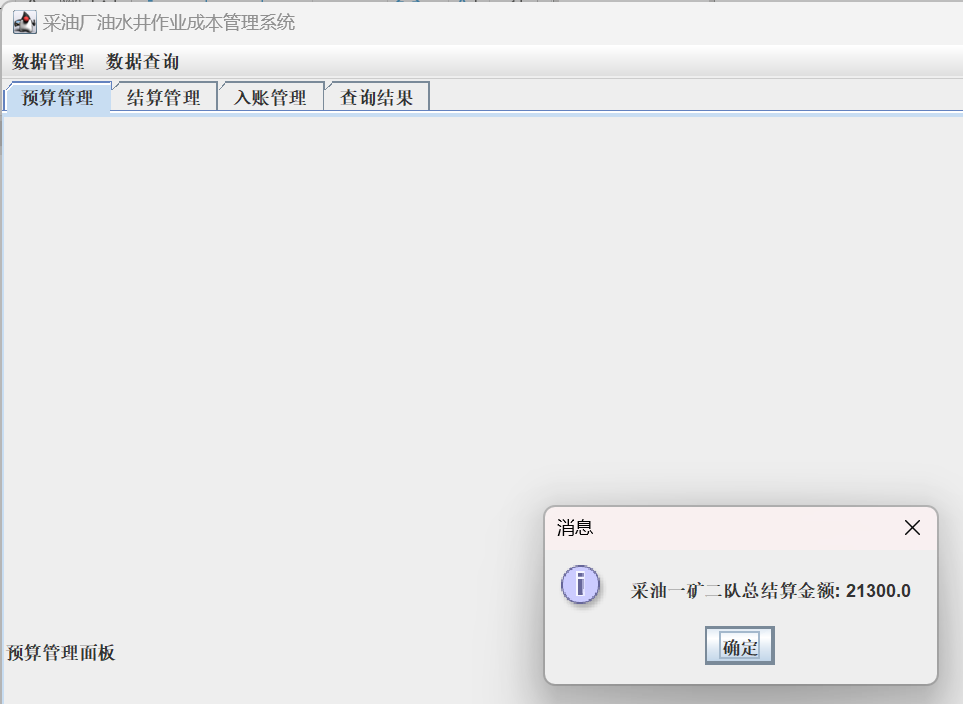
### ⑷ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了入账，列出相应明细。



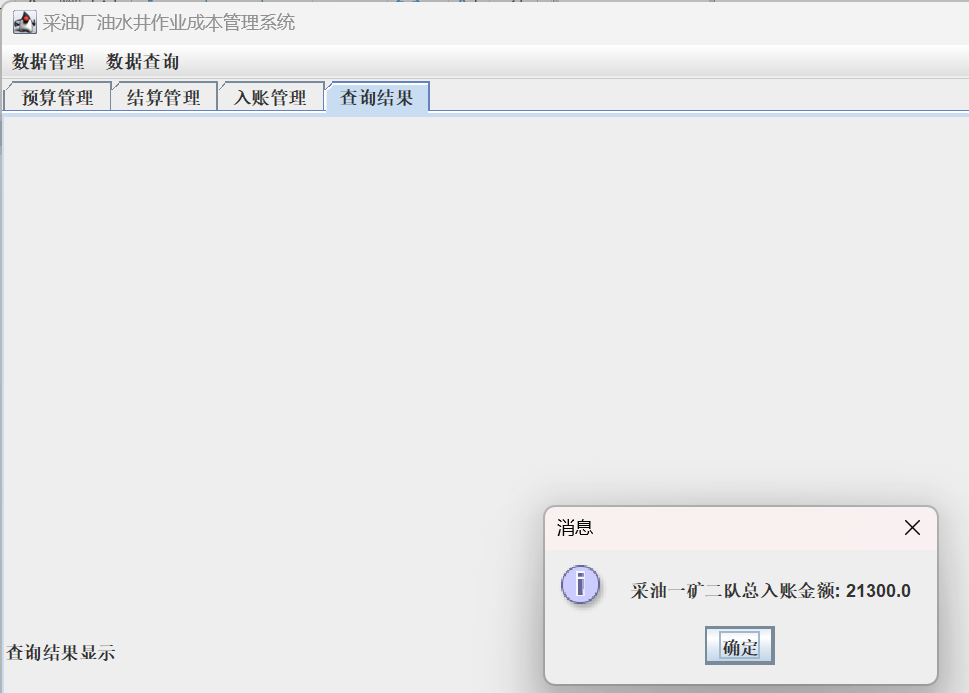
### ⑸ 列出采油一矿二队2018-5-1到2018-5-28总的预算金额。



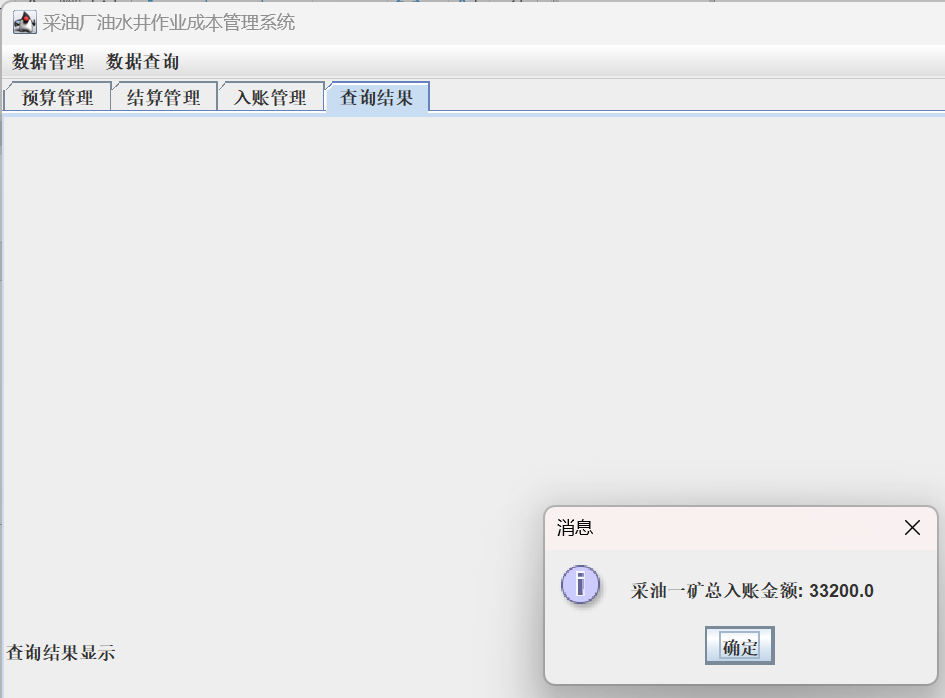
### ⑹ 列出采油一矿二队2018-5-1到2018-5-28总的结算金额。



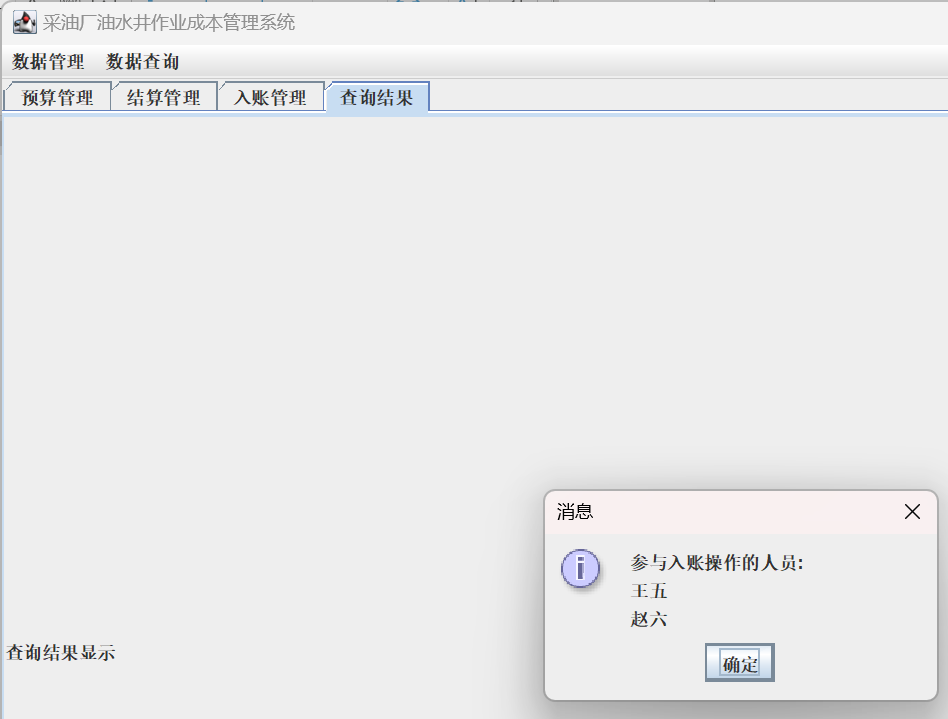
### ⑺ 列出采油一矿二队2018-5-1到2018-5-28总的入账金额。



### ⑻ 列出采油一矿2018-5-1到2018-5-28总的入账金额。

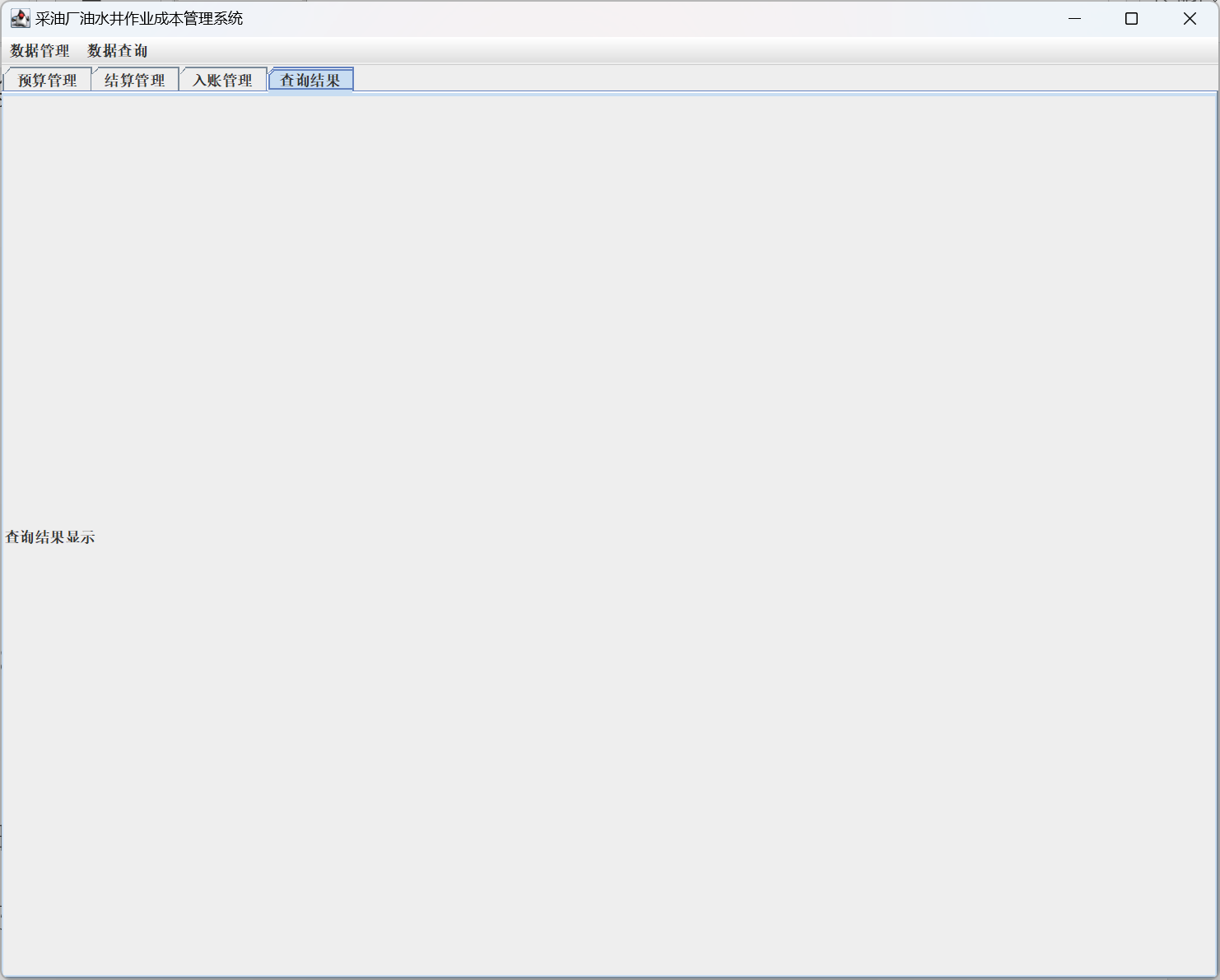


### ⑼ 有哪些人员参与了入账操作。

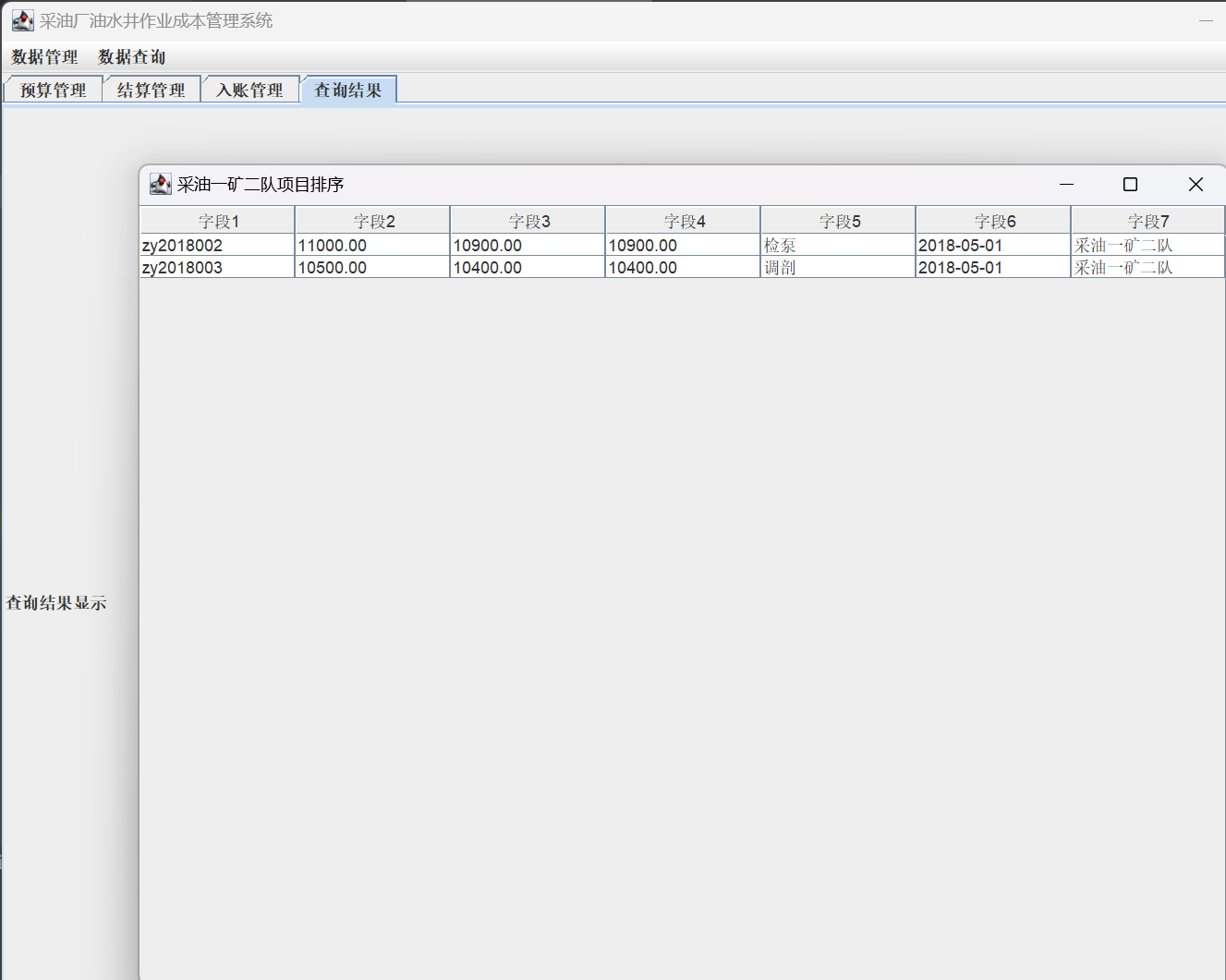


### ⑽ 列出2018-5-1到2018-5-28进行了结算但未入账的项目。

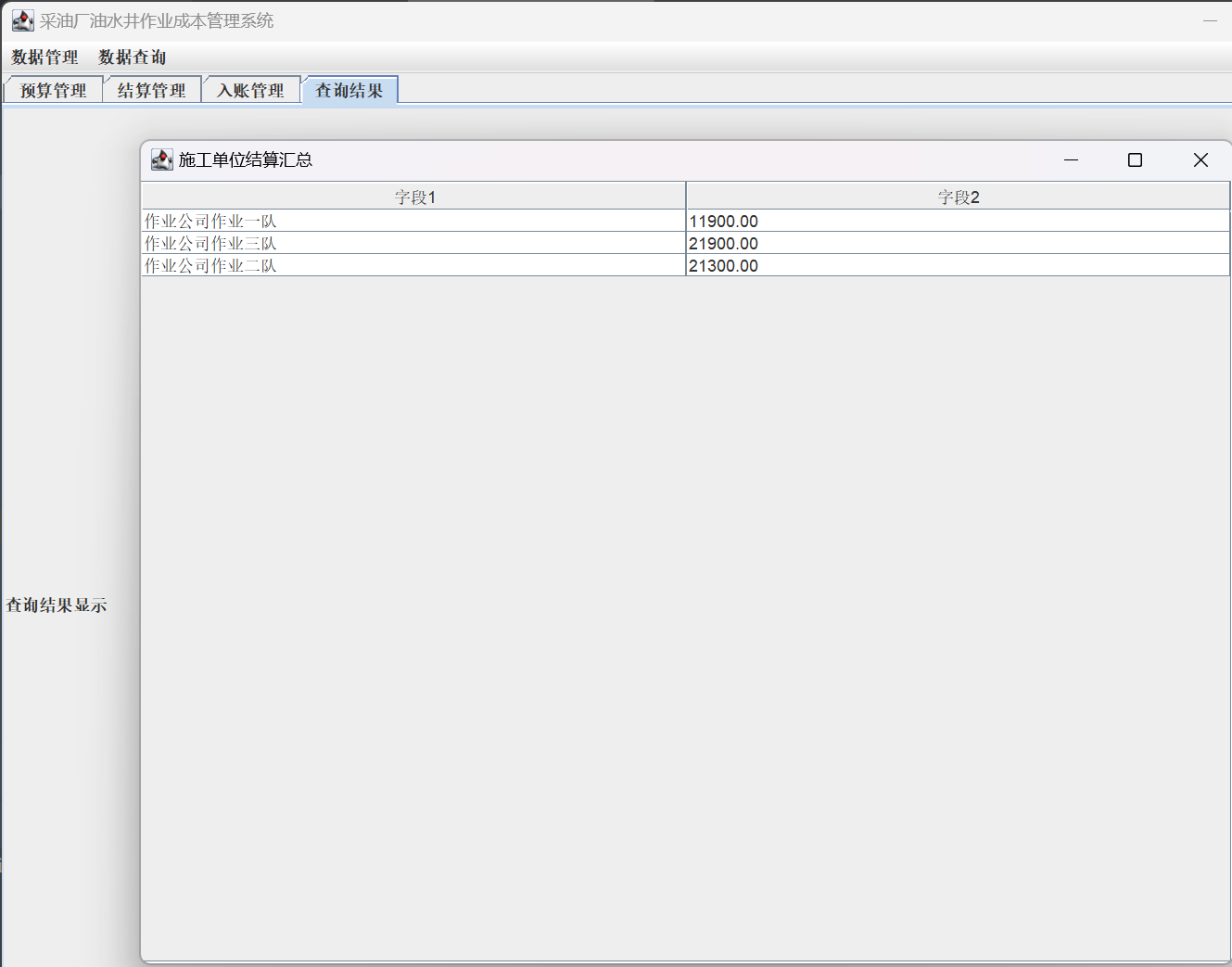
有zy2018005结算但未入账但结算时间为2018-06-01。故不存在2018-5-1到2018-5-28结算但未入账，故返回值均为N/A。



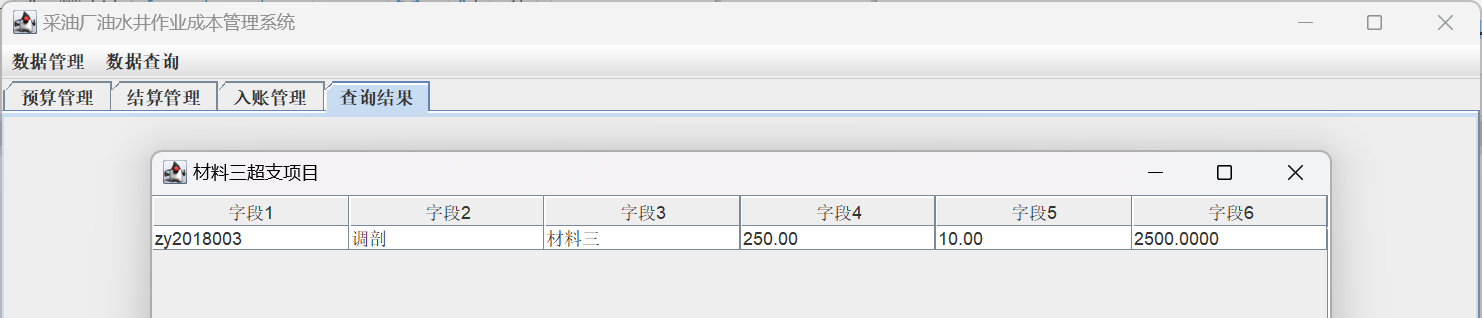
### ⑾ 列出采油一矿二队的所有项目，按入账金额（字段4）从高到低排列。



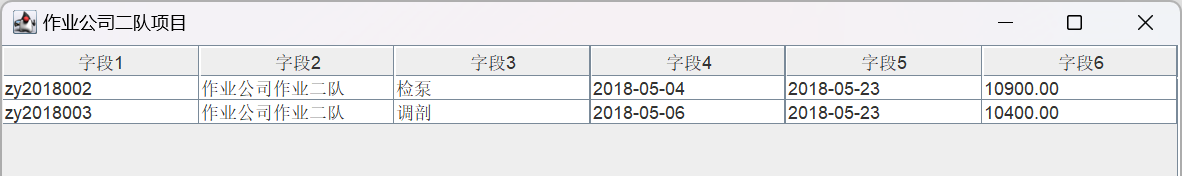
### ⑿ 列出有哪些施工单位实施了项目，并计算各单位所有项目结算金额总和。



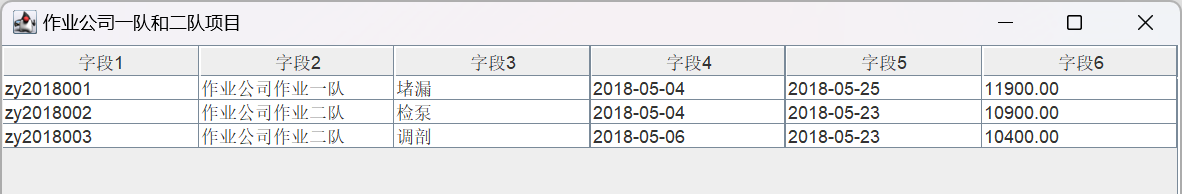
### ⒀ 找出消耗了材料三且消耗超过了2000元的项目，列出相应消耗明细(利用子查询)。



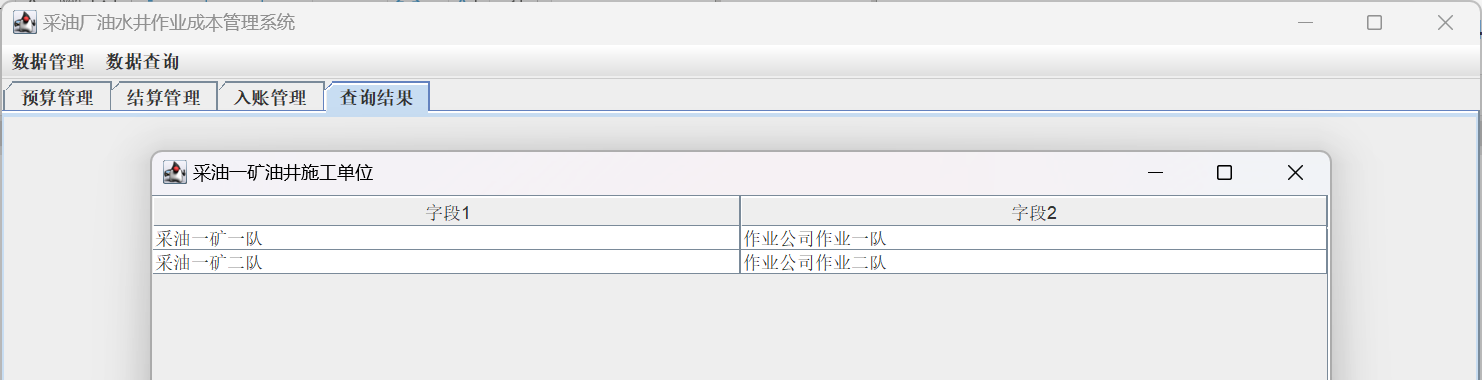
### ⒁ 作业公司二队参与了哪些项目。



### ⒂ 作业公司一队和二队参与了哪些项目（利用union）。



### ⒃ 采油一矿的油井是哪些作业队参与施工的。



# 二、实验收获

通过实验，我系统掌握了JDBC（Java Database Connectivity）技术的核心原理，包括数据库连接的建立、SQL 语句的执行以及结果集的处理，深刻体会到 JDBC在Java生态中的跨平台优势和标准化接口设计的便利性。

本次实验将理论知识（数据库原理、Java 编程）与工程实践深度结合，不仅掌握了JDBC等数据访问技术的落地应用，更培养了从需求分析到系统实现的完整开发思维。通过实现16项查询功能，数据增删改查，熟练掌握了SQL语句的高级用法：多表关联查询（JOIN）在整合作业预算表、结算表与单位代码表时的关键作用；子查询（如材料三超支项目）和UNION操作符在复杂业务逻辑中的灵活运用；聚合函数（SUM、COUNT）与分组（GROUP BY）在统计总预算、结算金额时的高效性。同时，通过 ER 图设计（单位代码表、作业预算表等），理解了数据库表结构与业务逻辑的映射关系，强化了范式设计的意识。

# 三、核心代码

完整代码仓库：[GALA-Lin/CUPK-DB-practice: CUPK 数据库实践结课设计](https://github.com/GALA-Lin/CUPK-DB-practice)(https://github.com/GALA-Lin/CUPK-DB-practice)

## JDBCUtil.java部分代码

import java.sql.\*;

public class JDBCUtil {

    private static final String URL = "jdbc:mysql://localhost:3306/zyxt?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";

    private static final String USER = "root";

    private static final String PASSWORD = "\*\*\*\*\*\*";

        } catch (SQLException e) {

            e.printStackTrace();

        }

## QueryManager.java部分代码

    // ⑴ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了预算

    public static List<Object[]> queryBudgetDetails() {

        String sql = "SELECT `单据号`, `预算金额`, `预算人`, `预算日期`"+

                "FROM `作业预算表`" +

                "JOIN `单位代码表` ON `预算单位` = `单位代码表`.`单位代码` " +

                "WHERE `单位名称` = '采油一矿二队' " +

                "AND `预算日期` BETWEEN '2018-05-01' AND '2018-05-28'";

        return executeQuery(sql);

    }

    // ⑵ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了结算，列出相应明细

    public static List<Object[]> querySettlementDetails() {

        String sql = "SELECT `作业结算表`.`单据号`, `结算金额`, `结算人`, `结算日期`, " +

                "`单位名称` AS `预算单位名称`, `施工内容`, " +

                "`开工日期`, `完工日期` " +

                "FROM `作业结算表`" +

                "JOIN `作业预算表` ON `作业预算表`.`单据号` = `作业结算表`.`单据号`" +

                "JOIN `单位代码表` ON `预算单位` = `单位代码`" +

                "WHERE `单位名称` = '采油一矿二队' " +

                "AND `结算日期` BETWEEN '2018-05-01' AND '2018-05-28'";

        return executeQuery(sql);

    }

    // ⑶ 采油一矿二队2018-5-1到2018-5-28有哪些项目完成了结算，列出相应的材料费消耗明细

    public static List<Object[]> queryMaterialCostDetails() {

        String sql = "SELECT `作业结算表`.`单据号`, `结算金额`, `材料费明细表`.`物码`, `名称规格`, `消耗数量`, `单价`," +

                "(`消耗数量` \* `单价`) AS 材料费用,`单位名称` " +

                "FROM `作业结算表`" +

                "JOIN `材料费明细表` ON `材料费明细表`.`单据号` = `作业结算表`.`单据号`" +

                "JOIN `物码表` ON `物码表`.`物码` = `材料费明细表`.`物码`" +

                "JOIN `作业预算表` ON `作业预算表`.`单据号` = `作业结算表`.`单据号`" +

                "JOIN `单位代码表` ON `作业预算表`.`预算单位` = `单位代码表`.`单位代码`" +

                "WHERE `单位名称` = '采油一矿二队'" +

                "AND `结算日期` BETWEEN '2018-05-01' AND '2018-05-28'";

        return executeQuery(sql);

    }

## DataManager.java部分代码

 // 添加数据 - 以作业预算表为例

    public static void addBudgetRecord(String billNo, String unitCode, String wellNo,

                                       double budgetAmount, String budgetPerson, Date budgetDate) {

        String sql = "INSERT INTO 作业预算表 (单据号, 预算单位, 井号, 预算金额, 预算人, 预算日期) " +

                "VALUES (?, ?, ?, ?, ?, ?)";

        Connection conn = null;

        PreparedStatement ps = null;

        try {

            conn = JDBCUtil.getConnection();

            ps = conn.prepareStatement(sql);

            ps.setString(1, billNo);

            ps.setString(2, unitCode);

            ps.setString(3, wellNo);

            ps.setDouble(4, budgetAmount);

            ps.setString(5, budgetPerson);

            ps.setDate(6, budgetDate);

            ps.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            JDBCUtil.close(conn, ps, null);

        }

    }

    // 更新数据 - 以作业预算表为例

    public static void updateBudgetRecord(String billNo, String unitCode, String wellNo,

                                          double budgetAmount, String budgetPerson, Date budgetDate) {

        String sql = "UPDATE 作业预算表 SET 预算单位 = ?, 井号 = ?, 预算金额 = ?, " +

                "预算人 = ?, 预算日期 = ? WHERE 单据号 = ?";

        Connection conn = null;

        PreparedStatement ps = null;

        try {

            conn = JDBCUtil.getConnection();

            ps = conn.prepareStatement(sql);

            ps.setString(1, unitCode);

            ps.setString(2, wellNo);

            ps.setDouble(3, budgetAmount);

            ps.setString(4, budgetPerson);

            ps.setDate(5, budgetDate);

            ps.setString(6, billNo);

            ps.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            JDBCUtil.close(conn, ps, null);

        }

    }

    // 删除数据 - 以作业预算表为例

    public static void deleteBudgetRecord(String billNo) {

        String sql = "DELETE FROM 作业预算表 WHERE 单据号 = ?";

        Connection conn = null;

        PreparedStatement ps = null;

        try {

            conn = JDBCUtil.getConnection();

            ps = conn.prepareStatement(sql);

            ps.setString(1, billNo);

            ps.executeUpdate();

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            JDBCUtil.close(conn, ps, null);

        }

    }