

**《数据库应用实践》**

实验报告/上机作业

**院 （系）：** 石油学院计算机系

**专业年级：** 数据科学与大数据技术23级

**学 号：** 2023015509

**姓 名：** 胡林森

**开课时间：** 2024-2025春季学期

石油学院计算机系

**2025年6月**

**填 写 说 明**

1. 实验报告A4纸，字体为4号宋体，单倍行距；

2. 指定实验报告序号和实验项目名称不能更改；

3. 每个实验项目填写一份实验报告，页面不够，可以加页。

4. 首次实验，实验报告1连同前面的1、2、3、4页一起提交，从第二次实验开始，只交单独的实验报告2、3…

《数据库应用实践》实验成绩评分标准

每次实验报告100分包括实验表现、现场提问回答20分。

|  |  |  |
| --- | --- | --- |
| **评分项** | **评分依据** | **分值** |
| 实验完成情况 | 按时、独立完成（10分）  实验结果完整程度（10分） | 20 |
| 实验完成质量 | 实验分析思路正确（10分）  实验结果完整、代码规范（20分）  完成内容步骤描述清晰（10分） | 40 |
| 实验报告质量 | 结果完整，实验报告文本规范（10分）  对本次实验进行了总结，有收获体会（10分） | 20 |
| 实验表现 | 包括现场提问和回答的正确性 | 20 |
| 总 分 | | 100 |

**实验项目及成绩登记表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验序号** | **实验日期** | **实验项目名称** | | **实验成绩** | **备 注** |
| **1** | 4.29 | 认识SQL Server | |  |  |
| **2** | 5.08 | SQL Server建库（表） | |  |  |
| **3** | 5.13 | 索引的建立、删除及使用 | |  |  |
| **4** | 5.20 | 表的删除与修改 | |  |  |
| **5** | 5.29 | 其它数据库对象的管理 | |  |  |
| **6** | 5.29 | 安全机制 | |  |  |
| **7** | 6.12 | 数据库应用编程 | |  |  |
| **8** | 6.17 | 综合实验 | |  |  |
| 实 验 报 告 最 终 成 绩：  （各次成绩总分的平均分） | | | 指导教师签名： | | |

注：

1.实验日期与实验项目名称须统一，不可修改；

2.实验成绩以百分制计分，不可涂改，确有涂改的，应在备注栏说明原因；

3.实验报告最终成绩为所有实验项目成绩的平均值，由实验指导教师红笔手写并签名。

**目 录**

[**一、实验环境** 6](#_Toc201057846)

[**（一）硬件环境** 6](#_Toc201057847)

[**（二）软件环境** 6](#_Toc201057848)

[**二、SQL Server 安装** 6](#_Toc201057849)

[**（一）准备工作** 6](#_Toc201057850)

[**（二）启动安装程序** 7](#_Toc201057851)

[**（三）Navicat介绍** 7](#_Toc201057852)

[**三、数据库设计** 8](#_Toc201057853)

[**（一）ER图** 8](#_Toc201057854)

[**（二）建表分析** 8](#_Toc201057855)

[**（三）数据插入** 12](#_Toc201057856)

[**四、数据查询** 12](#_Toc201057857)

[**五、数据操作** 13](#_Toc201057858)

[**六、视图创建与权限管理** 16](#_Toc201057859)

[**七、数据库应用编程** 17](#_Toc201057860)

[**（一）环境准备** 17](#_Toc201057861)

[**（二）JDBC安装** 17](#_Toc201057862)

[**（三）项目架构** 17](#_Toc201057863)

[**（四）查询页面** 17](#_Toc201057864)

[**（五）插入、修改、删除页面** 18](#_Toc201057865)

[**八、实验心得** 19](#_Toc201057866)

[**致 谢** 20](#_Toc201057867)

[**附 录：第七次实验代码** 21](#_Toc201057868)

[**JDBCUtil.java部分代码** 21](#_Toc201057869)

[**QueryManager.java部分代码** 21](#_Toc201057870)

[**DataManager.java部分代码** 22](#_Toc201057871)

**一、实验环境**

**（一）硬件环境**

**本次实验使用VMware workstation搭建虚拟机。**配置如下：

CPU：AMD R9 7940H；2核；4GHz

内存：2GB DDR4 2666MHz

硬盘：60GB固态硬盘

**（二）软件环境**

操作系统：Windows 10 教育版，64 位，版本号 22H2

安装软件：SQL Server 2022

**二、SQL Server 安装**

**（一）准备工作**

检查计算机硬件配置是否满足 SQL Server 2022 的最低安装要求。SQL Server 2022 ，本次实验使用的计算机硬件配置均满足要求。

关闭计算机上正在运行的其他应用程序，以避免安装过程中出现冲突。

以管理员身份登录 Windows 操作系统，确保具有足够的权限进行软件安装。



**（二）启动安装程序**

双击下载好的文件，将其加载到虚拟光驱中。在资源管理器中打开虚拟光驱，找到并双击 Setup.exe 文件，启动 SQL Server 安装中心。

在 SQL Server 安装中心窗口中，选择安装选项卡，然后点击全新 SQL Server 独立安装或向现有安装添加功能选项。

**（三）Navicat介绍**

Navicat for SQL Server 是一款专为 Microsoft SQL Server 设计的图形化数据库管理与开发工具，支持本地及远程数据库连接，并兼容主流云数据库（如 Amazon RDS、Microsoft Azure 等）。其直观的界面和丰富的功能使其成为数据库管理、数据迁移及开发的高效工具，尤其适用于实验环境中对数据库的快速操作与可视化分析。

核心功能有：

**1. 对象设计器**

用户可直接创建、修改表、视图、索引等数据库对象，无需手动编写复杂 SQL 语句。

**2. 数据操作与查询**

网格/表单视图：支持以表格形式编辑数据，提供排序、筛选及外键关联下拉选择功能。

SQL 编辑器：内置代码自动补全与语法高亮功能，简化复杂查询的编写。实验中复杂查询通过可视化 SQL 生成器快速实现。

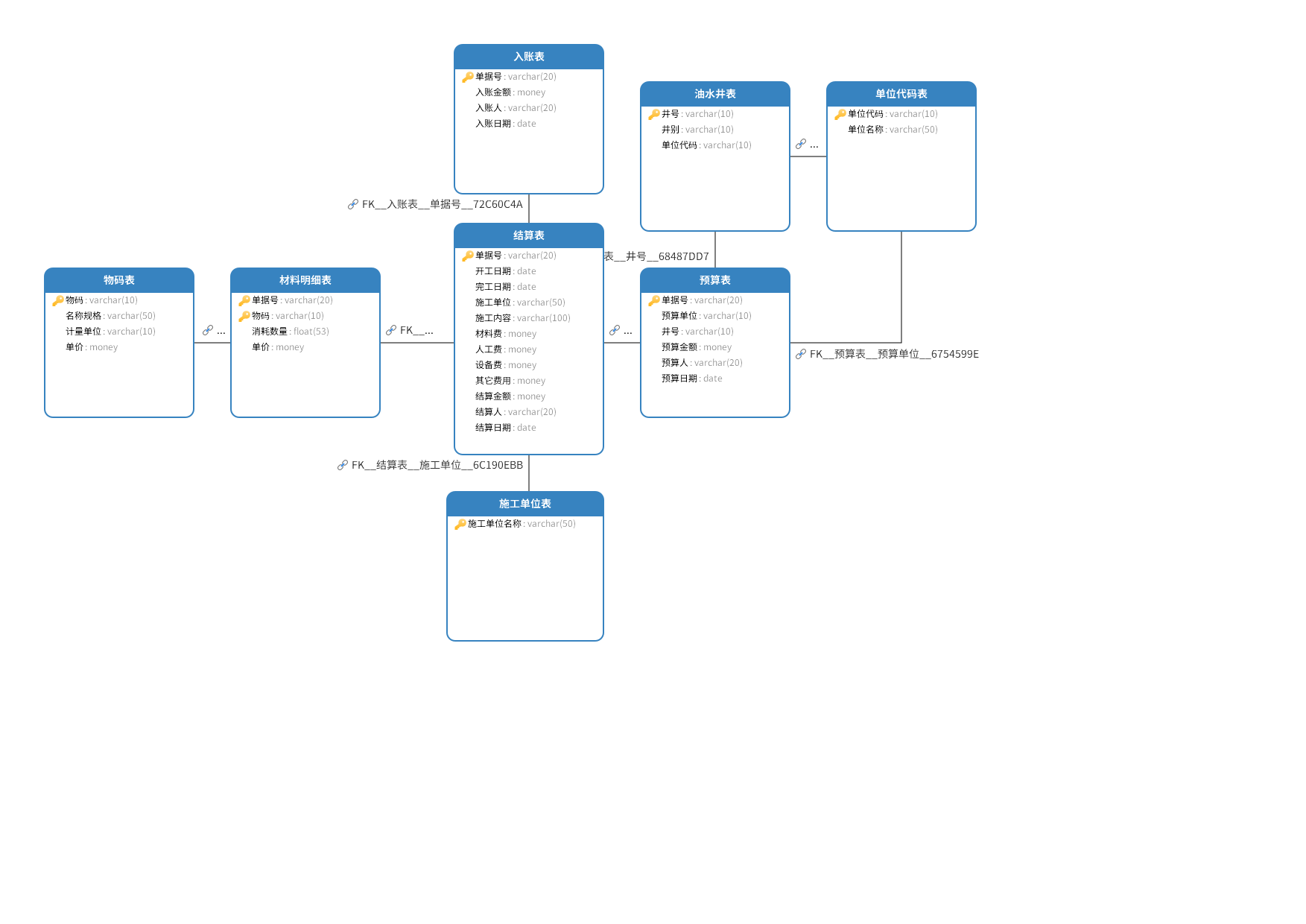
**3. 可视化分析与设计**

**生成ER 图表：**通过逆向工程生成数据库模型，直观展示表间关系；

**数据图表**：将查询结果转换为柱状图或折线图，辅助分析课程成绩分布等实验数据。

**三、数据库设计**

**（一）ER图**



**（二）建表分析**

**1. 单位代码表**

**属性：**

单位代码（VARCHAR(10)）：唯一标识单位。

单位名称（VARCHAR(50)）：单位全称，不可为空。

**主键：**单位代码

**外键：**无

**2. 油水井表**

**属性：**

井号（VARCHAR(10)）：唯一标识油/水井。

井别（VARCHAR(10)）：井的类型，**通过“check”函数仅允许油井或水井**。

单位代码（VARCHAR(10)）：所属单位的代码。

**主键：**井号

**外键：**

单位代码 引用 单位代码表的单位代码。

**3. 施工单位表**

**属性：**施工单位名称（VARCHAR(50)）：唯一标识施工单位。

**主键：**施工单位名称

**外键：**无

**4. 物码表**

**属性：**

物码（VARCHAR(10)）：唯一标识物料。

名称规格（VARCHAR(50)）：物料名称及规格。

计量单位（VARCHAR(10)）：如“吨”、“件”等。

单价（MONEY）：默认为 10。

**主键：**物码

**外键：**无

**5. 预算表**

**属性：**

单据号（VARCHAR(20)）：唯一标识预算单据。

预算单位（VARCHAR(10)）：编制预算的单位代码。

井号（VARCHAR(10)）：关联的油/水井编号。

预算金额（MONEY）：预算总金额。

预算人（VARCHAR(20)）：编制人姓名。

预算日期（DATE）：编制日期。

**主键：**单据号

**外键：**

预算单位 引用 单位代码表的单位代码。

井号 引用 油水井表的井号。

**6. 结算表**

**属性：**

单据号（VARCHAR(20)）：关联的预算单据号。

开工日期（DATE）：施工开始日期。

完工日期（DATE）：施工结束日期。

施工单位（VARCHAR(50)）：施工单位名称。

施工内容（VARCHAR(100)）：施工描述。

材料费（MONEY）、人工费（MONEY）、设备费（MONEY）、其它费用（MONEY）：分项费用。

结算金额（MONEY）：总结算金额。结算人（VARCHAR(20)）：结算操作人。

结算日期（DATE）：结算操作日期。

**主键：**单据号

**外键：**

单据号 引用 预算表的单据号。

施工单位 引用 施工单位表的施工单位名称。

**7. 材料明细表**

**属性：**单据号（VARCHAR(20)）：关联的结算单据号。

物码（VARCHAR(10)）：使用的物料代码。

消耗数量（FLOAT）：物料消耗数量。

单价（MONEY）：实际结算单价。

**主键：**单据号, 物码（联合主键）

**外键：**

单据号 引用 结算表的单据号。

物码 引用 物码表的物码。

**8. 入账表**

**属性：**

单据号（VARCHAR(20)）：关联的结算单据号。

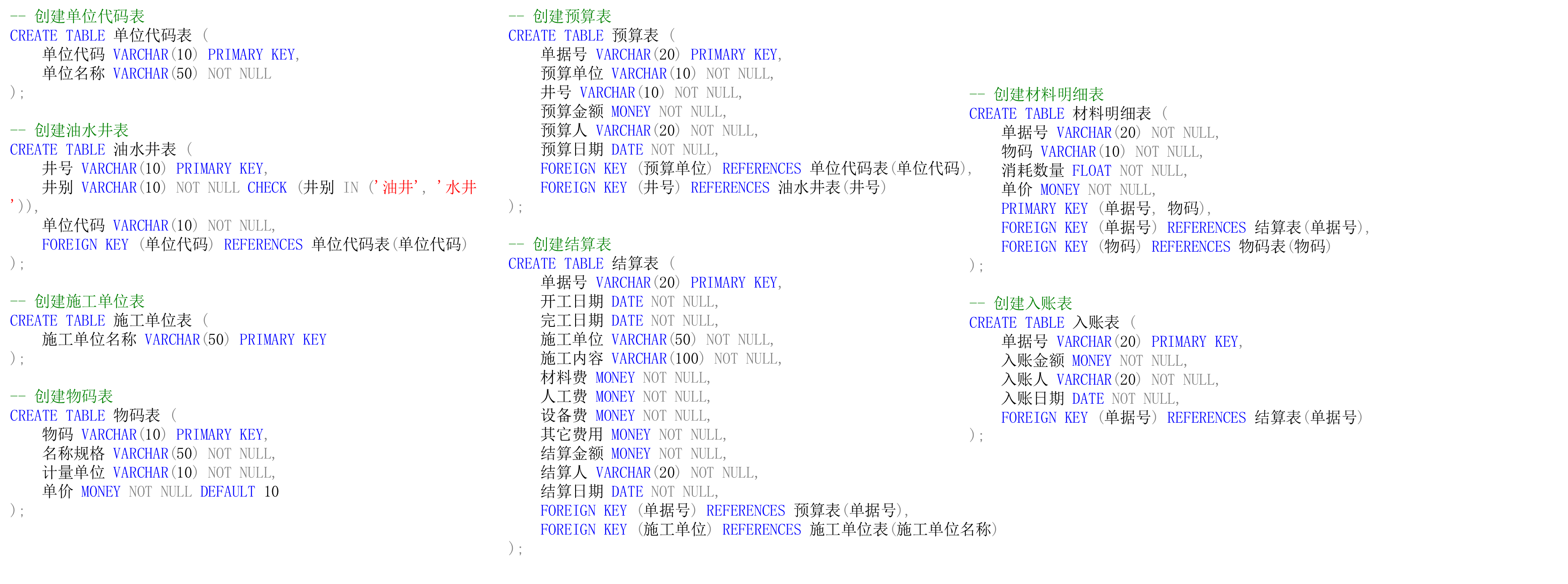
入账金额（MONEY）：实际入账金额。

入账人（VARCHAR(20)）：入账操作人。

入账日期（DATE）：入账日期。

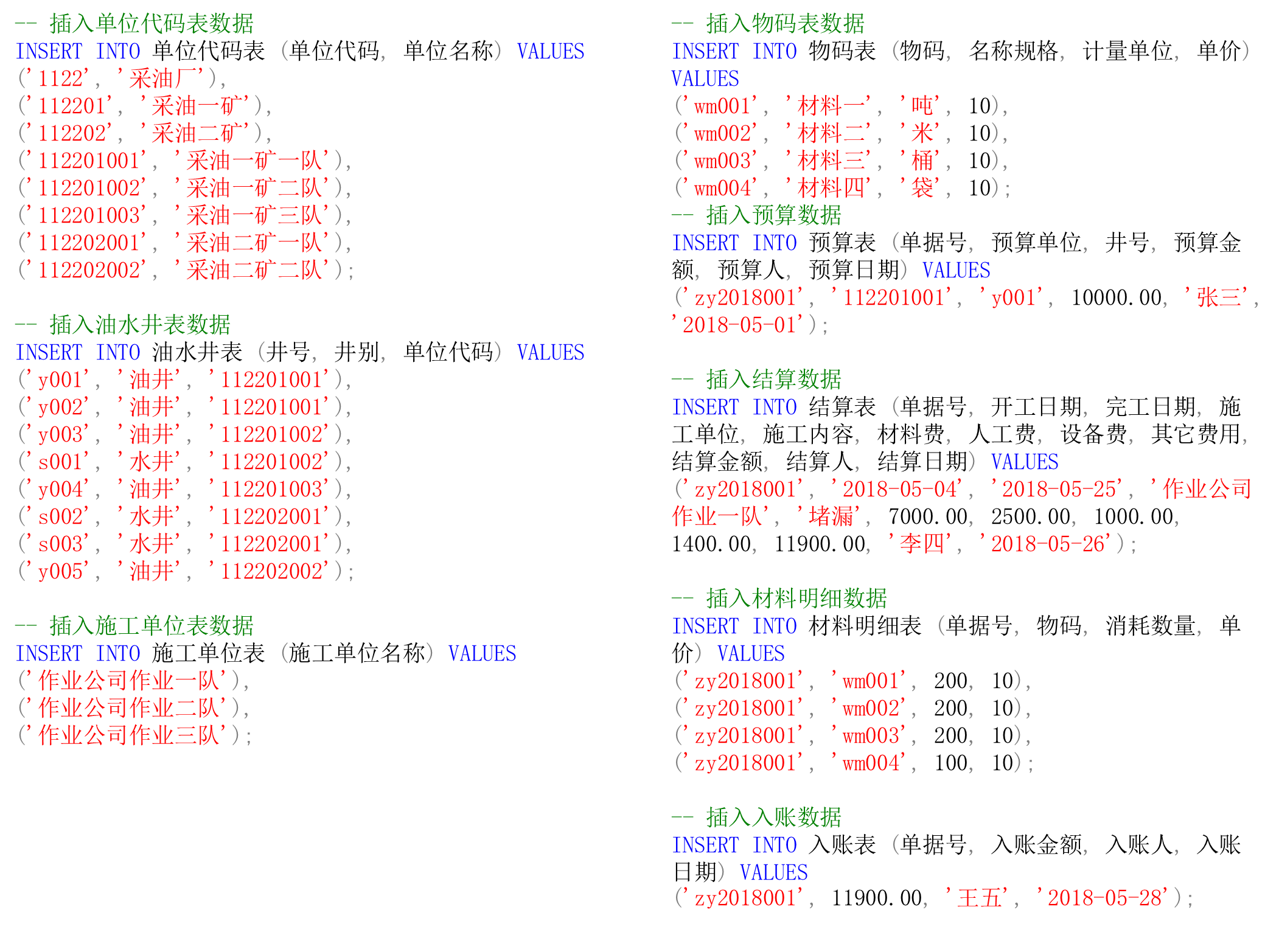
**主键：**单据号

**外键：**单据号 引用 结算表的单据号。



**（三）数据插入**

由于代码过长，业务数据以zy2018001为例：



**四、数据查询**

**如：**列出采油一矿二队的所有项目，按入账金额从高到低排列。



解释：通过单据号依次关联四张表：入账表 → 结算表 → 预算表 → 单位代码表以获得单位名称。ORDER BY 入账表.入账金额 DESC：按入账金额从高到低降序排列。

ASC：升序（默认值）。DESC：降序

**结果：**



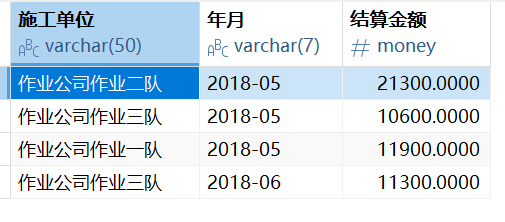
**五、数据操作**

插入数据：



解释：显然完成实验内容要求无需使用子查询，第二段代码更优：直接使用GROUP BY一步完成聚合，无需JOIN和冗余分组，逻辑清晰且性能更高。

结果：

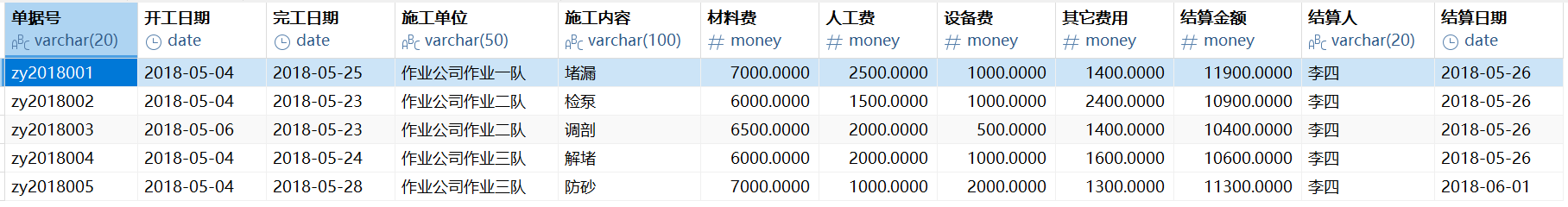


修改、删除数据

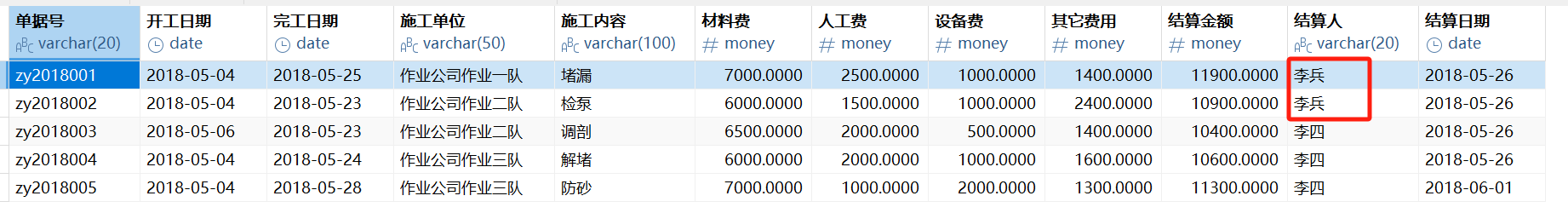




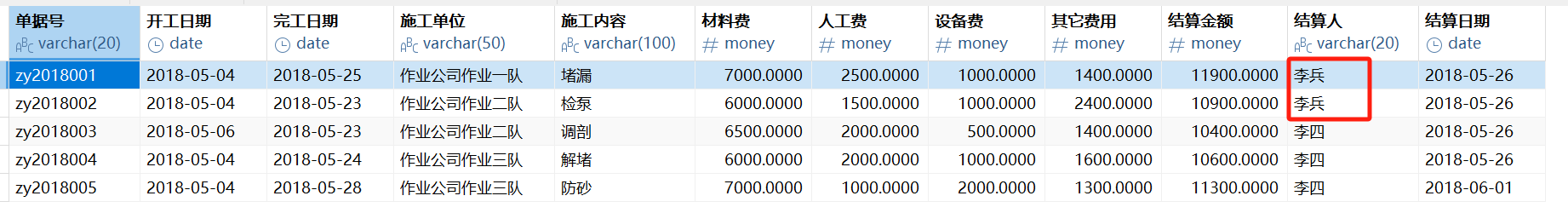
修改前：



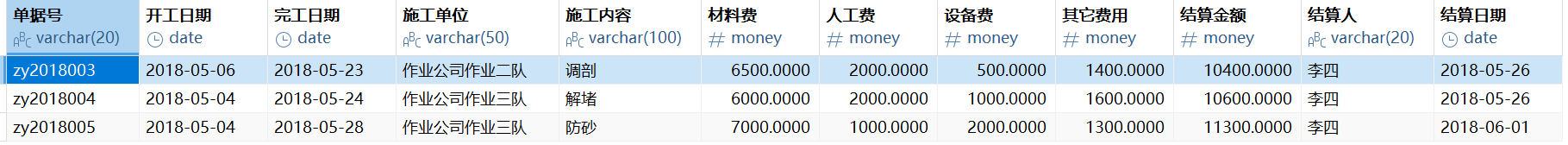
修改后：



删除前：



删除后：



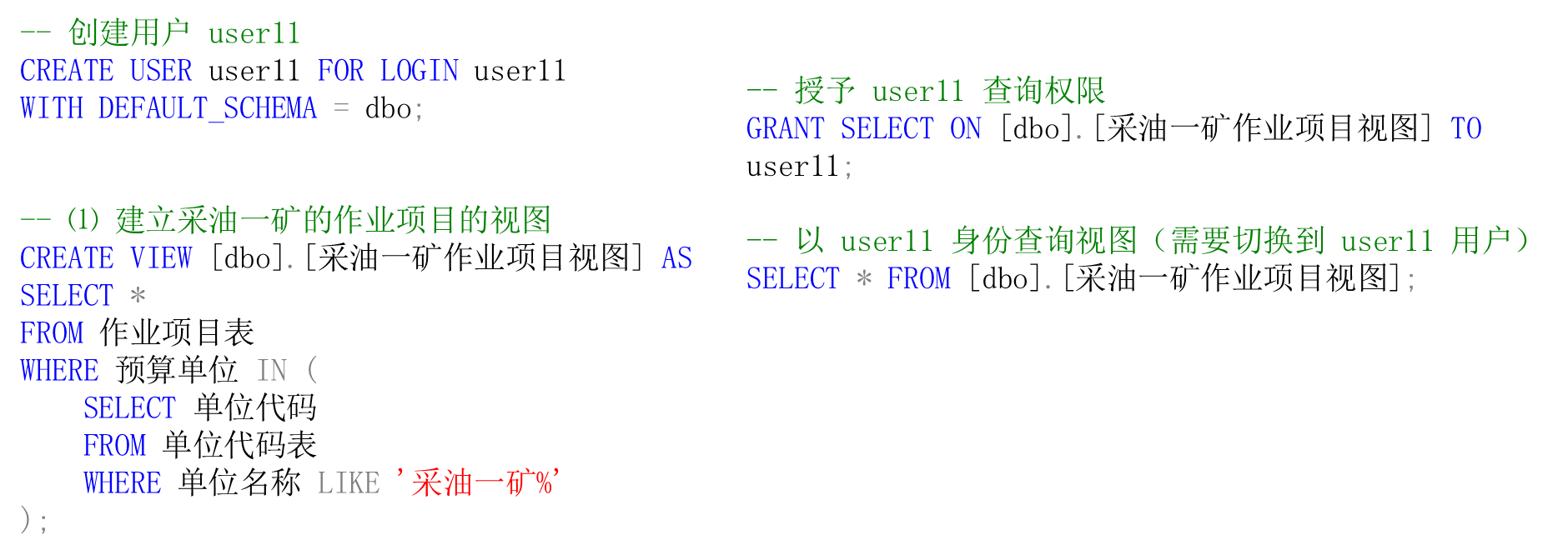
**六、视图创建与权限管理**

编写 SQL 语句，创建一个名为 采油一矿作业项目视图 的视图，该视图仅包含采油一矿相关的作业项目。

在 SQL Server 中创建一个名为 user11 的新用户。

授予 user11 对 采油一矿作业项目视图 的查询权限。

切换到 user11 用户身份，执行查询语句，验证是否能够成功查询视图。





**七、数据库应用编程**

**（一）环境准备**

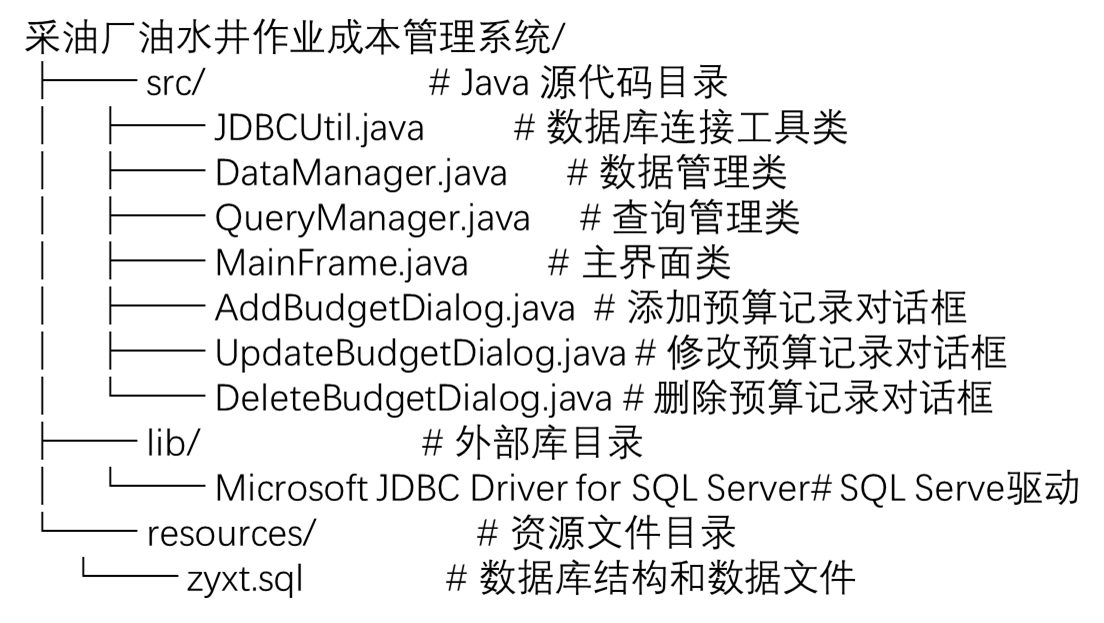
**1.技术栈**

|  |  |
| --- | --- |
| **编程语言：** | Java |
| **JDK：** | 22 |
| **界面框架：** | Java Swing |
| **构建工具：** | Maven |
| **IDE** | IntelliJ IDEA |

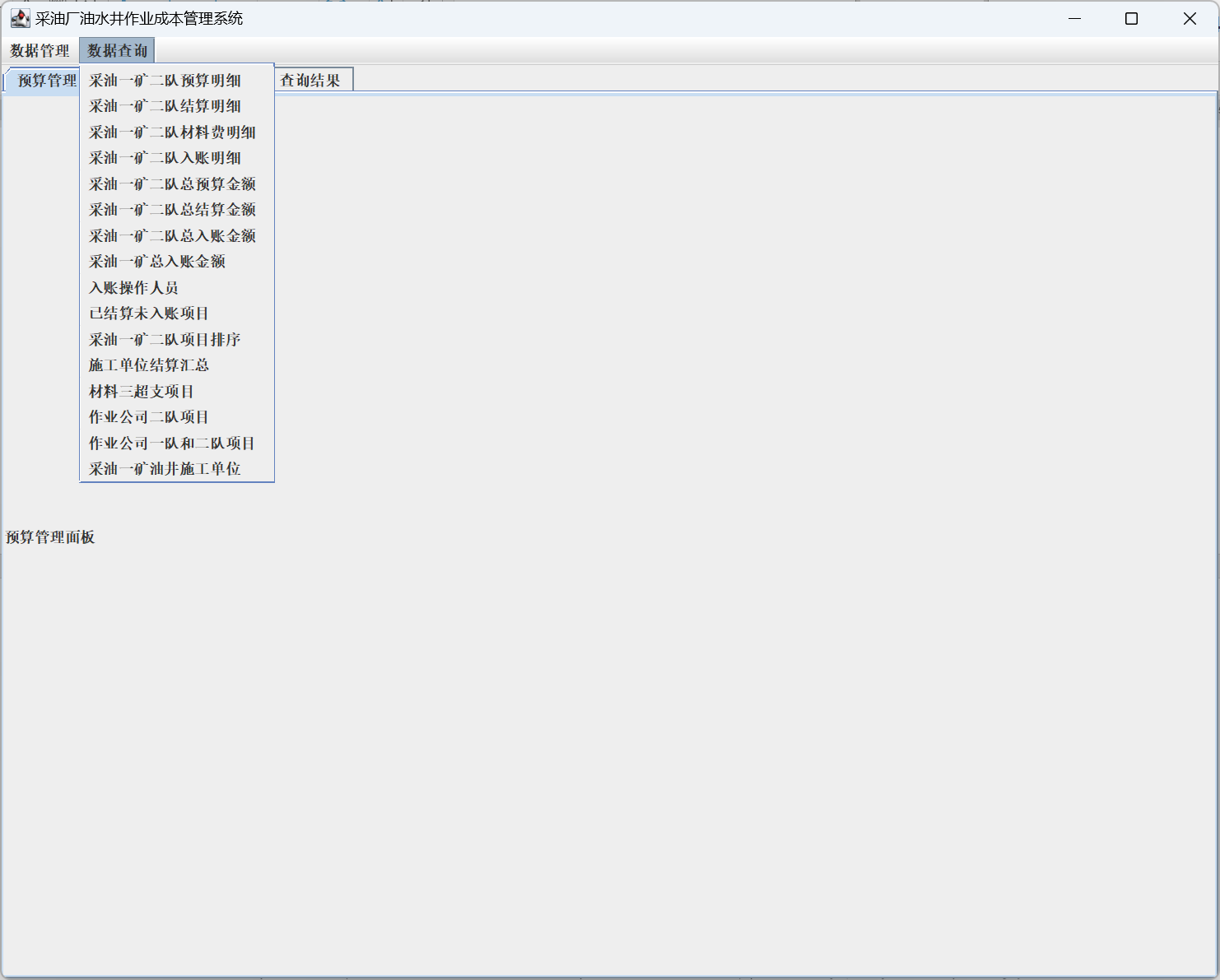
**（二）JDBC安装**

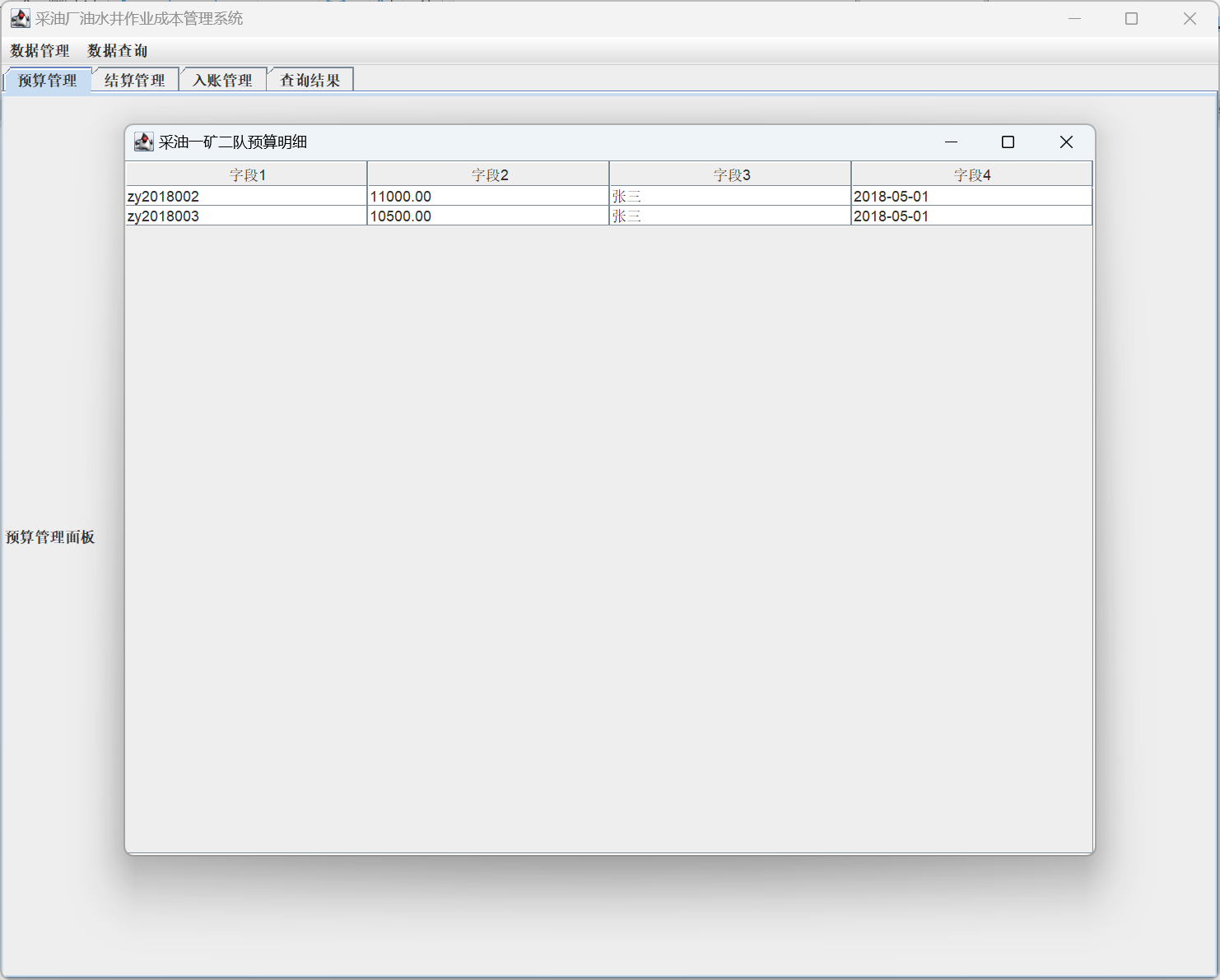
从官网下载对应版本Microsoft JDBC Driver for SQL Server。

**（三）项目架构**

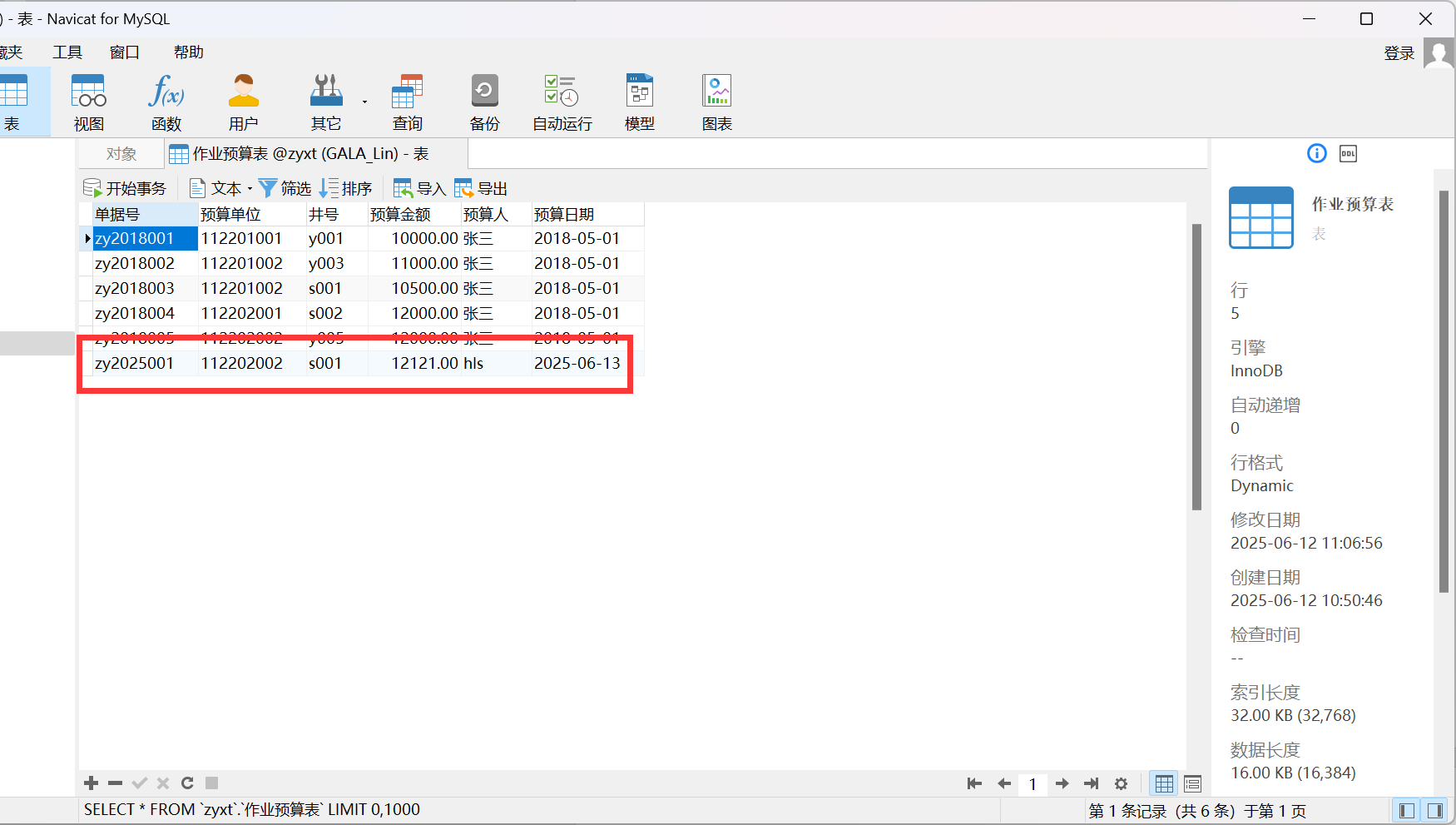
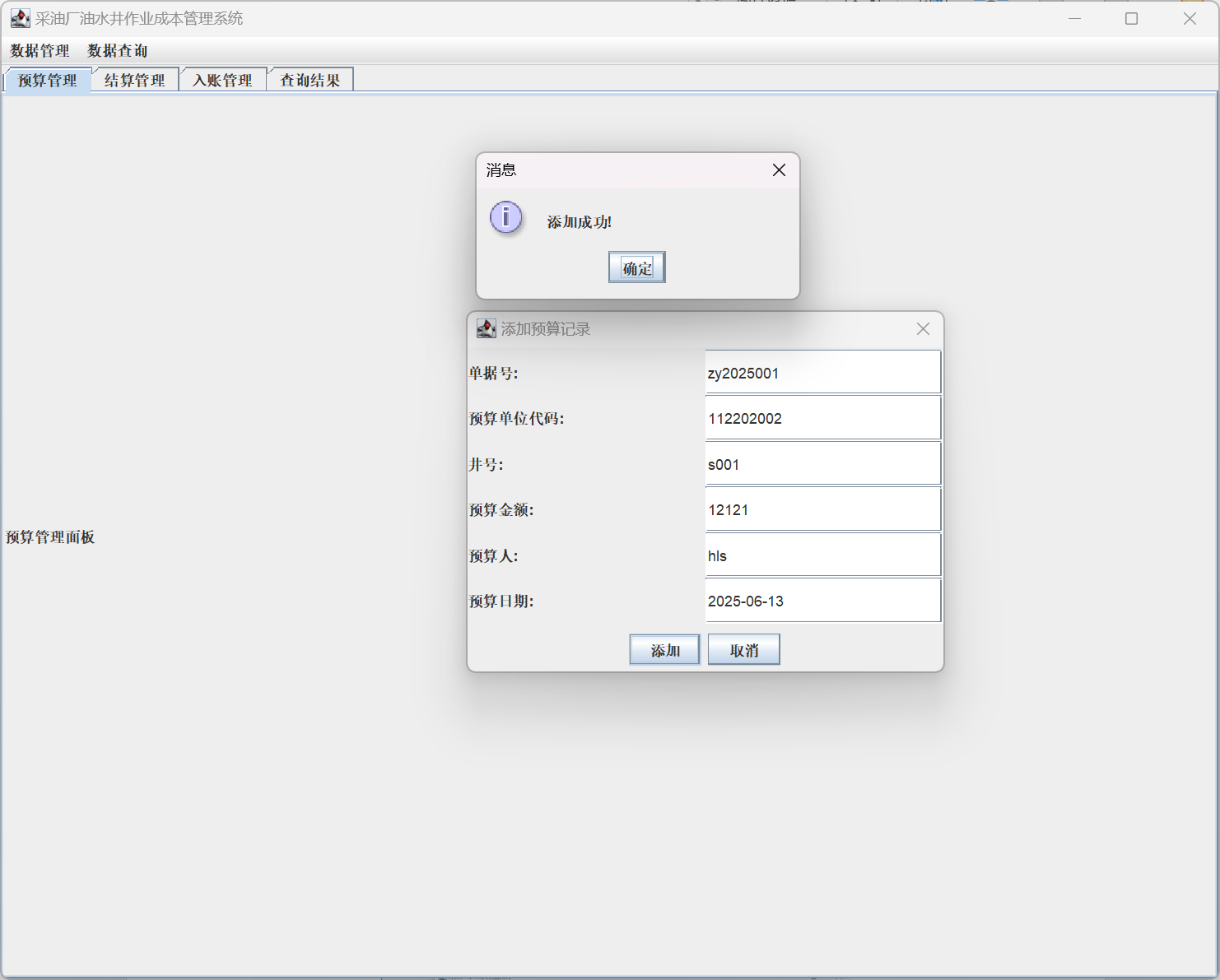


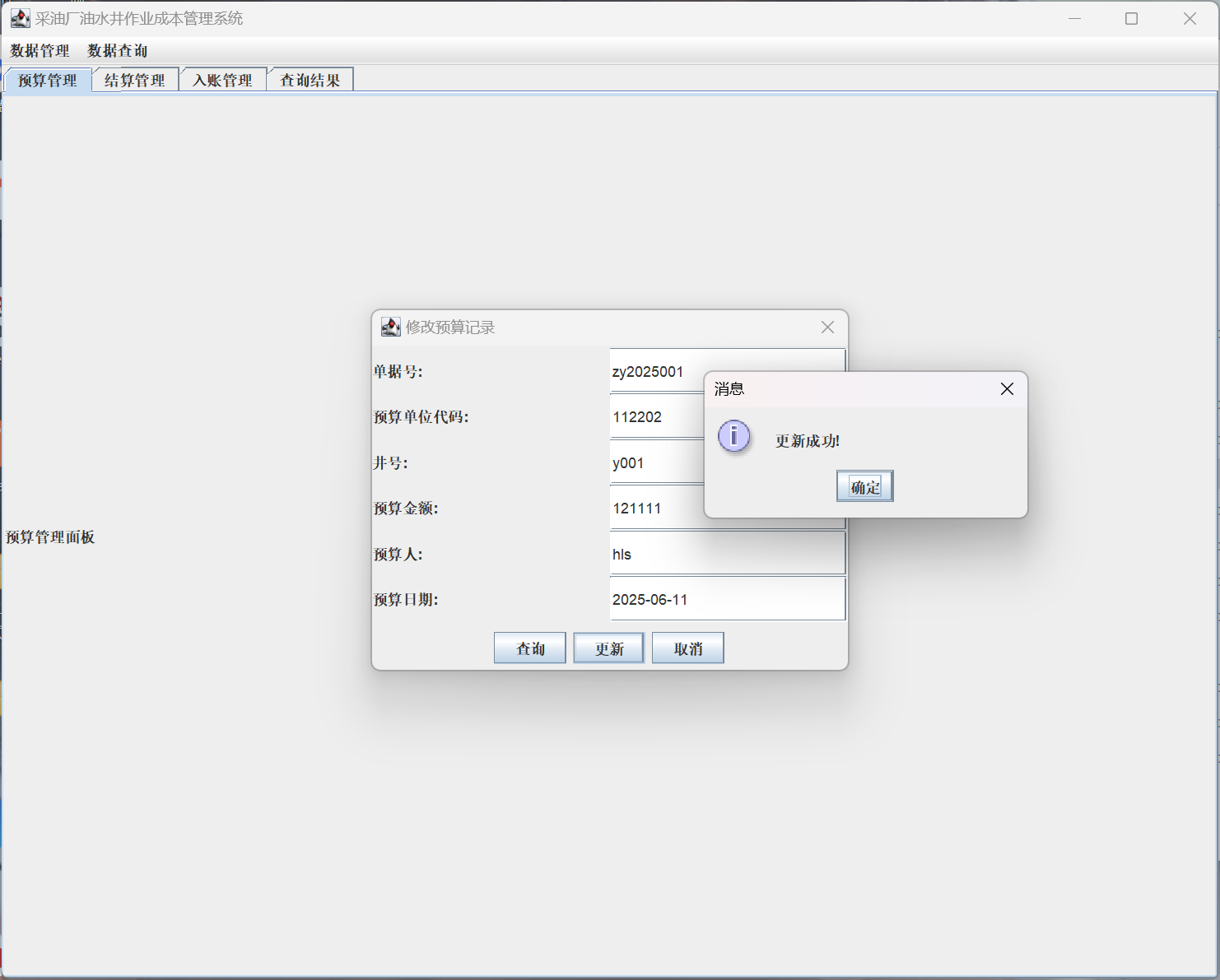
**（四）查询页面**

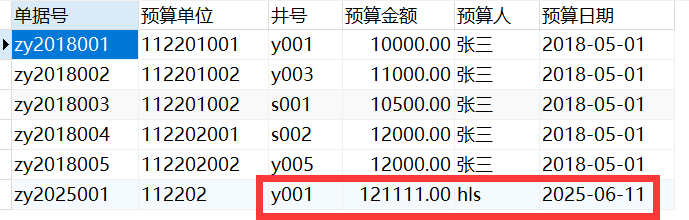


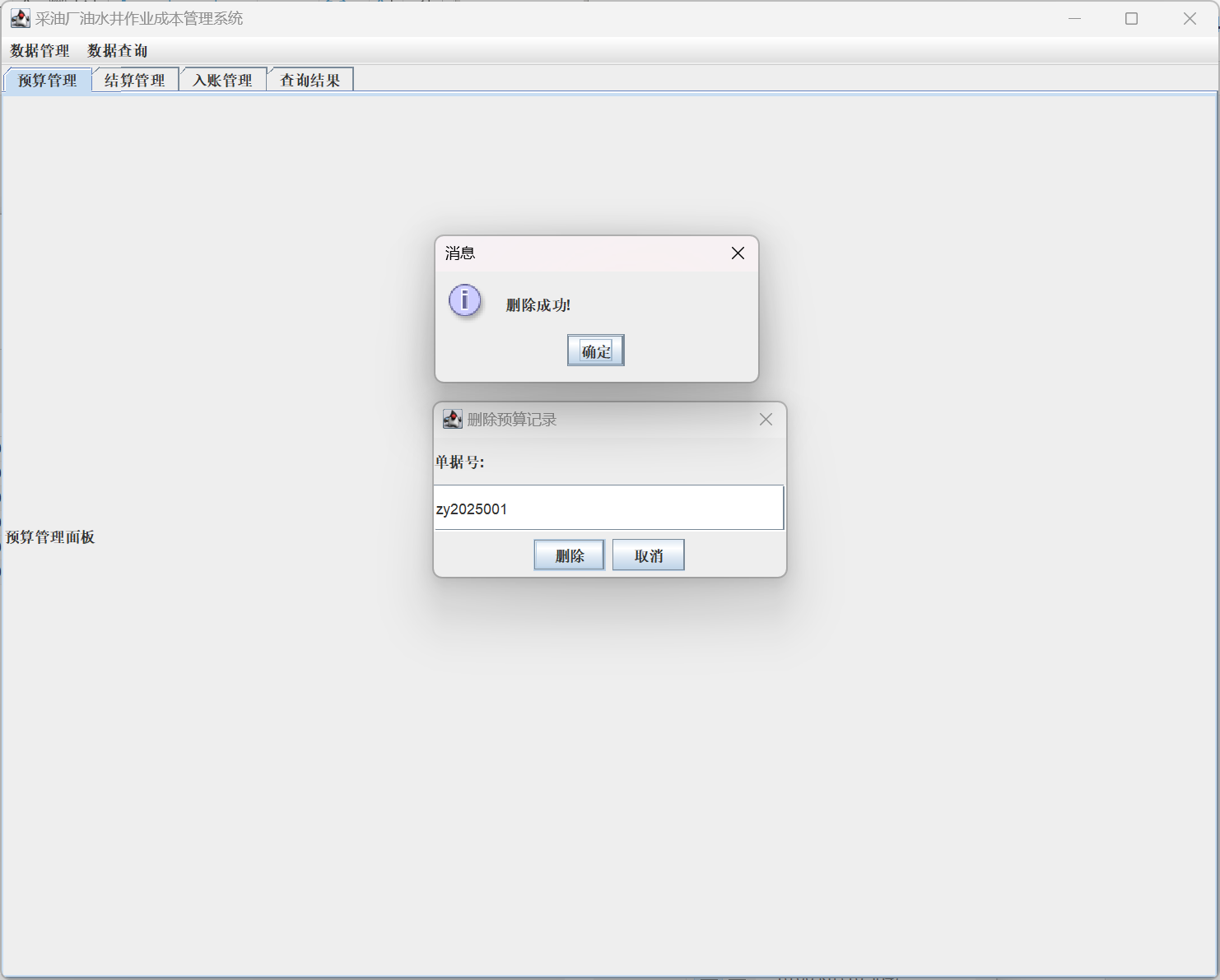


**（五）插入、修改、删除页面**











**八、实验心得**

在本次《数据库应用实践》实验中，我完成了从 SQL Server 环境搭建到数据库应用编程的全流程实践，这让我对数据库系统的认知实现了从课本理论到实际应用的质变。

我通过VMware虚拟机搭建起实验环境，主要采用Navicat工具，尤其是其可视化的ER图生成功能让抽象的表间关系变得直观可触，这种将概念模型转化为物理模型的过程，让我深刻体会到数据库设计“先建模后实现”的严谨逻辑。

查询操作中，我深入了解了JOIN操作符背后蕴含着对业务逻辑的深度解构。以 “采油一矿二队项目查询” 为例，通过单据号串联入账表、结算表、预算表和单位代码表的过程，本质上是对作业流程中预算-结算-入账业务逻辑的映射。

在JDBC编程部分，项目架构的分层设计让我体会到软件工程的模块化思想。JDBCUtil工具类对数据库连接的封装，QueryManager 与DataManager的职责分离，这些看似繁琐的代码结构，在后续维护和扩展时展现出强大优势。

期间也遇到了许多问题，如外键约束下数据增删改的逻辑曾经困扰我，在邱老师的教学与自我摸索下我逐渐解决问题，不惧挫折，，获得了成就感。

**致 谢**

这一学期以来学习的数据库课程让我受益匪浅，特别是实践阶段。感谢邱老师在实验过程中的悉心指导！每当我在SQL语句调试、表结构设计遇到困惑时，老师总能耐心讲解关键要点，帮助我理清思路，也让我对数据库技术有了更系统的认知。

**附 录：第七次实验代码**

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);

        }

    }