软件体系结构与设计考查报告

姓　　名：焦熙鹏
学　　号：20231002234
班　　级：191231
学　　院：计算机学院
题　　目：二次压降检测仪检定信息管理系统的设计与实现
指导 教师：王勇

2026 年 1 月

第 1 章 项目概述与需求分析

1.1 项目背景

随着电力系统的不断发展和完善，互感器二次压降检测仪作为重要的计量检测设备，其检定工作量日益增加。传统的纸质记录和人工统计方式存在效率低下、数据易丢失、查询不便、统计困难等问题。为了提高检定工作的效率和质量，规范检定数据的管理，实现检定报告的自动化生成，开发一套专业的二次压降检测仪信息管理系统具有重要的现实意义。

本系统旨在通过信息化手段，实现检定数据的电子化存储、规范化管理和智能化分析，为检定工作提供全面的信息化支持。

1.2 项目目标

本项目的主要目标是开发一套功能完善、操作便捷的二次压降检测仪信息管理系统，具体目标包括：

（1）实现检定数据的规范化录入和批量导入，提高数据录入效率。

（2）建立完整的数据库管理系统，确保数据的安全性和完整性。

（3）提供灵活的查询统计功能，支持多维度的数据分析。

（4）实现检定报告的自动化生成，支持 PDF 导出和打印功能。

（5）提供友好的用户界面，降低用户学习成本，提高工作效率。

1.3 功能需求分析

1.3.1 数据输入管理需求

系统需要支持两种数据输入方式：

（1）手动输入：提供友好的数据录入界面，支持逐条添加厂商信息、设备信息、检测记录和检测结果数据。录入界面需要包含必要的数据验证功能，确保数据的准确性和完整性。对于具有关联关系的数据（如设备与厂商的关联），需要提供下拉选择框，方便用户快速选择。

（2）批量导入：支持 CSV/Excel 格式文件的批量导入功能，适用于大量历史数据的迁移或批量录入场景。导入过程需要显示进度条，并在导入完成后给出详细的统计信息（成功数量、失败数量、失败原因等）。

1.3.2 数据管理需求

系统需要将所有实验数据统一存储在数据库中，支持完整的数据管理功能：

（1）数据查询：支持按不同维度查询数据，包括按厂商查询、按设备查询、按时间范围查询等。查询结果以表格形式展示，支持排序和筛选。

（2）数据修改：支持对已录入数据的修改功能，修改时需要保持数据的完整性约束。

（3）数据删除：支持数据的删除功能，删除时需要考虑关联数据的处理（级联删除或拒绝删除）。

（4）数据展示：提供清晰的数据展示界面，支持在不同数据表之间快速切换。

1.3.3 查询统计需求

系统需要提供丰富的查询统计功能，支持多维度的数据分析：

（1）时间段统计：支持按时间段（开始日期到结束日期）统计检测数量，了解不同时期的工作量分布。

（2）厂商统计：支持按厂商统计检测数量和合格率，分析不同厂商设备的质量水平。

（3）月度统计：支持按月统计检测工作量，便于工作量的月度汇总和分析。

（4）联表查询：支持跨表的联合查询，例如查询某个厂商的所有设备及其检测记录。

统计结果需要以直观的方式展示，包括表格和汇总信息，必要时可提供图表展示。

1.3.4 报告生成需求

系统需要根据实验结果自动生成规范的检测报告：

（1）报告内容：报告应包含完整的检测信息，包括设备基本信息、检测条件、检测结果项（PT1、PT2、CT1、CT2）、详细测量数据（ao、bo、co 点的测量值）、检测结论等。

（2）报告格式：报告格式需符合行业规范，布局合理、信息完整、易于阅读。

（3）PDF 导出：支持将报告导出为 PDF 格式，便于存档和分发。

（4）打印功能：支持直接打印报告，提供打印预览功能。

1.4 非功能需求分析

（1）可靠性需求：系统应具有较高的稳定性，避免因程序错误导致数据丢失。数据库操作需要有适当的错误处理机制。

（2）易用性需求：系统界面应简洁友好，操作流程符合用户习惯。关键操作需要有明确的提示信息。

（3）可维护性需求：代码结构应清晰，模块划分合理，便于后期维护和功能扩展。

（4）兼容性需求：系统应能在 Windows 10 及以上操作系统上稳定运行。

第 2 章 系统分析与设计

2.1 系统架构设计

2.1.1 总体架构

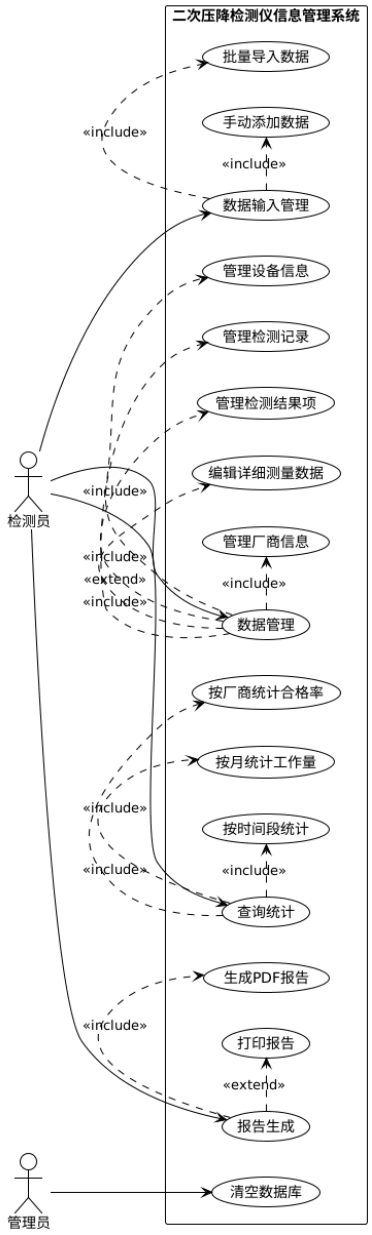

图 2-1 用例图

如图 2-1 所示，系统主要包含以下功能模块：

（1）数据输入管理模块

支持两种数据输入方式：

手动添加数据：通过对话框逐条录入厂商、设备、检测记录等信息

批量导入数据：支持 CSV 文件的批量导入，提高历史数据迁移效率

（2）数据管理模块

提供完整的数据管理功能（CRUD），包括：

管理厂商信息：维护设备制造商的基础信息

管理设备信息：记录被检测设备的详细信息

管理检测记录：记录每次检测的基础数据

管理检测结果项：记录 PT 侧/CT 侧的检测结果

编辑详细测量数据：维护 ao/bo/co 三个测量点的详细数据（扩展功能）

（3）查询统计模块

支持多维度的数据统计分析：

按时间段统计：统计指定时间范围内的检测数量

按厂商统计合格率：分析不同厂商设备的质量水平

按月统计工作量：便于月度工作量汇总

（4）报告生成模块

实现检定报告的自动化生成：

生成 PDF 报告：根据检测数据自动生成规范的检定报告

打印报告：支持报告的直接打印（扩展功能）

（5）系统管理功能

管理员拥有特殊权限：

清空数据库：用于系统测试或数据重置（需管理员权限）

2.1.2 技术架构

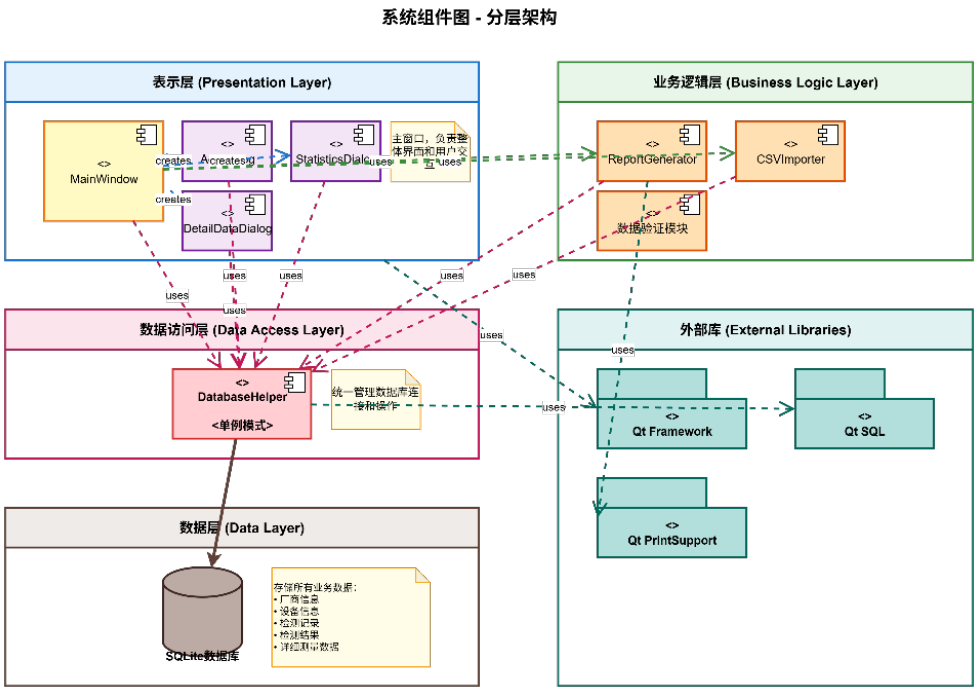

图 2-2 分层架构图

如图 2-2，本系统采用解耦的四层架构设计。在经典三层架构的基础上，明确了外部框架的支撑作用，通过严格的层级依赖关系（向下依赖），确保了各功能模块的高内聚与低耦合。

（1） 表示层 (Presentation Layer)

作为系统的流量入口，该层直接面向用户。以 MainWindow 为核心枢纽，动态调度 AddDialog（数据录入）、StatisticsDialog（统计分析）及 DetailDataDialog（详情查看）等交互组件。该层依托 Qt Widgets 实现了响应式 UI，仅负责数据展示与初步的用户输入校验，不涉及核心业务逻辑。

（2） 业务逻辑层 (Business Logic Layer)

本层是系统的“大脑”，负责处理复杂的业务规则。

数据流转核心：CSVImporter 实现了外部异构数据（CSV）到系统内部模型的转换；ReportGenerator 负责将业务指标转化为标准的 PDF 导出报告。

规则引擎：内置数据验证模块，确保所有进入持久化层的数据均符合预设的检测规则与业务约束。

（3） 数据访问层 (Data Access Layer) 与 数据层 (Data Layer)

该层充当业务逻辑与物理存储之间的中间件。

访问控制：DatabaseHelper 采用单例模式 (Singleton Pattern) 设计，确保了数据库连接池的唯一性与稳定性，封装了 CRUD（增删改查）操作，避免了 SQL 逻辑散落在业务层。

持久化存储：底层基于 SQLite 嵌入式数据库，存储包含厂商信息、设备参数、检测记录及其明细在内的全量核心数据。

（4） 外部支撑库 (External Libraries)

这是系统稳固运行的基石。系统深度集成 Qt Framework，利用其信号槽机制进行组件通信，并调用 Qt SQL 模块进行数据库底层驱动，以及 Qt PrintSupport 实现报告的物理输出。

2.2 技术选型

2.2.1 开发框架选择

本系统选择 Qt 6.8.3 作为开发框架，主要基于以下考虑：

（1）跨平台特性：Qt 是一个跨平台的 C++图形用户界面应用程序开发框架，可以在 Windows、Linux、macOS 等多个平台上运行，具有良好的可移植性。

（2）丰富的组件库：Qt 提供了丰富的 GUI 组件，如 QTableView、QDialog、QMenu 等，能够快速构建专业的用户界面。

（3）完善的数据库支持：Qt SQL 模块提供了对多种数据库的支持，包括 SQLite、MySQL、PostgreSQL 等，使用 QSqlTableModel 和 QSqlQuery 可以方便地进行数据库操作。

（4）信号槽机制：Qt 的信号槽机制提供了一种类型安全的事件通信方式，使得对象之间的通信更加简洁和灵活。

（5）文档处理能力：Qt 提供了 QPdfWriter、QPrinter 等类，可以方便地生成 PDF 文档和实现打印功能。

2.2.2 数据库选择

本系统选择 SQLite 作为数据库管理系统，主要原因如下：

（1）轻量级：SQLite 是一个嵌入式数据库，不需要独立的数据库服务器进程，数据库存储在单个文件中，便于部署和管理。

（2）零配置：SQLite 不需要安装和配置，开箱即用，降低了系统部署的复杂度。

（3）可靠性：SQLite 实现了 ACID（原子性、一致性、隔离性、持久性）特性，保证了数据的可靠性。

（4）完整的 SQL 支持：SQLite 支持标准 SQL 语法，包括复杂查询、聚合函数、联表查询等，能够满足系统的统计分析需求。

（5）与 Qt 的良好集成：Qt SQL 模块对 SQLite 提供了原生支持，通过 QSQLITE 驱动可以直接访问 SQLite 数据库。

（6）适合中小规模应用：对于本系统的数据规模（预计万条级别的记录），SQLite 能够提供良好的性能表现。

2.3 数据库设计

2.3.1 数据库 E-R 图

系统的实体关系图如图 2-3 所示，包含五个核心实体及其关联关系。

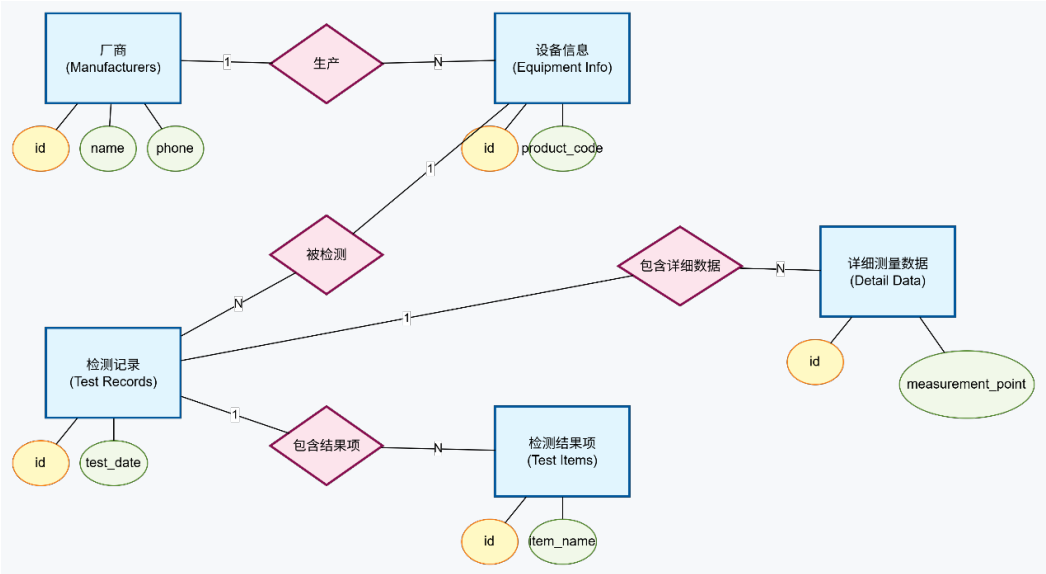

图 2-3 系统 E-R 图

实体关系说明：

（1）一个厂商可以生产多台设备（1:n 关系） （2）一台设备可以有多次检测记录（1:n 关系） （3）一次检测记录包含多个检测结果项（1:n 关系） （4）一次检测记录包含一组详细测量数据（1:n 关系）

2.3.2 数据表设计

系统共设计了 5 张数据表，各表的详细结构如下：

表 2-1 manufacturers（厂商信息表）

字段名	数据类型	约束	说明
id	INTEGER	PRIMARY KEY AUTOINCREMENT	主键，自增
name	TEXT	NOT NULL UNIQUE	厂商名称，唯一
contact_person	TEXT	-	联系人
phone	TEXT	-	联系电话
address	TEXT	-	厂商地址
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

表 2-2 equipment_info（设备信息表）

字段名	数据类型	约束	说明
id	INTEGER	PRIMARY KEY AUTOINCREMENT	主键，自增
product_code	TEXT	NOT NULL	产品编号
product_name	TEXT	NOT NULL	产品名称
manufacturer_id	INTEGER	FOREIGN KEY	外键，关联厂商表
production_date	DATE	-	生产日期
product_location	TEXT	-	产地
product_model	TEXT	-	产品型号
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

表 2-3 test_records（检测记录表）

字段名	数据类型	约束	说明
id	INTEGER	PRIMARY KEY AUTOINCREMENT	主键，自增
equipment_id	INTEGER	NOT NULL, FOREIGN KEY	外键，关联设备表
test_date	DATE	NOT NULL	检测日期
tester_name	TEXT	NOT NULL	检测员姓名
test_location	TEXT	NOT NULL	检测地点
secondary_voltage	REAL	-	二次电压
temperature	REAL	-	环境温度
humidity	REAL	-	环境湿度
metering_point_code	TEXT	-	计量点编号
test_date_code	TEXT	-	测试日期编号
remarks	TEXT	-	备注信息
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

表 2-4 test_result_items（检测结果项表）

字段名	数据类型	约束	说明
id	INTEGER	PRIMARY KEY AUTOINCREMENT	主键，自增
test_record_id	INTEGER	NOT NULL, FOREIGN KEY	外键，关联检测记录
item_name	TEXT	NOT NULL	项目名称（PT1/PT2/CT1/CT2）

字段名	数据类型	约束	说明
gear_position	TEXT	-	档位（如 100V、5A）
percentage	REAL	-	百分比（如 20%、100%）
data_lower_limit	REAL	-	数据下限
data_upper_limit	REAL	-	数据上限
measured_data	REAL	-	实测数据
is_qualified	BOOLEAN	-	是否合格
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

说明：每次检测记录通常包含 4 个检测结果项（PT1、PT2、CT1、CT2），每个项目包含多个不同档位和百分比的测量点。

表 2-5 test_result_details（详细测量数据表）

字段名	数据类型	约束	说明
id	INTEGER	PRIMARY KEY AUTOINCREMENT	主键，自增
test_record_id	INTEGER	NOT NULL, FOREIGN KEY	外键，关联检测记录
item_name	TEXT	NOT NULL	项目名称（PT 侧/CT 侧）
ao_f	REAL	-	ao 点 f(%)
ao_d	REAL	-	ao 点 d(分)
ao_du	REAL	-	ao 点 dU(%)
ao_upt	REAL	-	ao 点 Upt:U

字段名	数据类型	约束	说明
ao_uyb	REAL	-	ao 点 Uyb:U
bo_f	REAL	-	bo 点 f(%)
bo_d	REAL	-	bo 点 d(分)
bo_du	REAL	-	bo 点 dU(%)
bo_upt	REAL	-	bo 点 Upt:U
bo_uyb	REAL	-	bo 点 Uyb:U
co_f	REAL	-	co 点 f(%)
co_d	REAL	-	co 点 d(分)
co_du	REAL	-	co 点 dU(%)
co_upt	REAL	-	co 点 Upt:U
co_uyb	REAL	-	co 点 Uyb:U
pt_check_note	TEXT	-	PT 侧备注
r_percentage	REAL	-	r%值
measurement_result	TEXT	-	测量结束值
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

说明：该表存储每次检测的详细测量数据，包括 ao、bo、co 三个测量点的完整数据。

2.4 类设计

2.4.1 核心类图

系统的核心类及其关系如图 2-2 所示。

图 2-2 系统核心类图

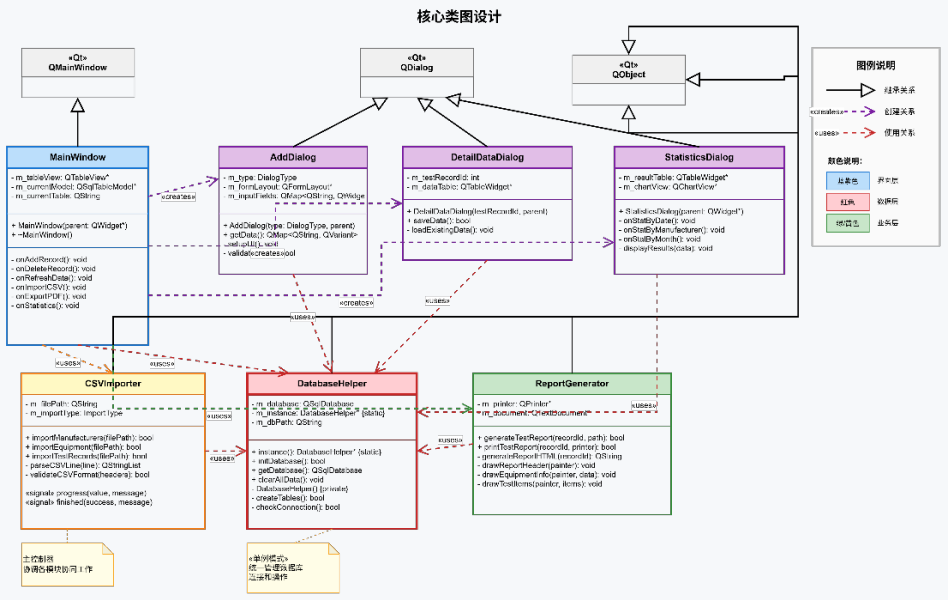

图 2-4 系统核心类图

如图 2-4 为系统核心类图设计，包含 7 个核心类：

MainWindow 作为主控制器，继承自 QMainWindow，负责创建 AddDialog、DetailDataDialog、StatisticsDialog 三个对话框处理用户交互；DatabaseHelper 采用单例模式统一管理数据库连接，被所有类依赖使用；ReportGenerator 实现 PDF 报告生成；CSVImporter 处理批量数据导入。类图体现了清晰的分层架构和面向对象设计原则，通过继承、创建、依赖三种关系构建了高内聚低耦合的系统结构。

2.4.2 类职责说明

（1）**MainWindow 类**

职责：系统主窗口，负责界面布局、菜单栏、工具栏的创建，以及各功能模块的调度。

主要方法：

setupUI()：初始化用户界面

setupMenuBar()：创建菜单栏

setupToolBar()：创建工具栏

loadTableData()：加载并展示数据表

onAddRecord()：添加新记录

onDeleteRecord()：删除选中记录

onRefreshData()：刷新数据

onImportCSV()：导入 CSV 文件

onStatistics()：打开统计对话框

onExportPDF()：导出 PDF 报告

onPrintReport()：打印报告

（2）DatabaseHelper 类

职责：数据库管理类，采用单例模式，负责数据库的初始化、表的创建和数据库连接的管理。

主要方法：

instance()：获取单例实例

initDatabase()：初始化数据库连接

getDatabase()：获取数据库对象

createTables()：创建数据库表

insertTestDetailsData()：插入测试数据

clearAllData()：清空所有数据

（3）AddDialog 类

职责：数据添加对话框，提供统一的数据录入界面，支持厂商、设备、检测记录、检测结果的添加。

主要方法：

setupUI()：根据表类型创建相应的输入控件

onAccept()：验证并保存数据

loadRelatedData()：加载关联数据（如厂商列表、设备列表）

（4）CSVImporter 类

职责：CSV 文件导入器，负责解析 CSV 文件并将数据批量导入数据库。

主要方法：

importManufacturers()：导入厂商数据

importEquipment()：导入设备数据

importTestRecords()：导入检测记录

importTestItems()：导入检测结果项

parseCSVLine()：解析 CSV 行数据

（5）StatisticsDialog 类

职责：统计分析对话框，提供多维度的数据统计功能。

主要方法：

setupUI()：创建统计界面

onTimeRangeStatistics()：按时间段统计

onManufacturerStatistics()：按厂商统计

onMonthlyStatistics()：按月统计

（6）ReportGenerator 类

职责：报告生成器，负责生成检测报告并导出为 PDF 格式。

主要方法：

generateReport()：生成 HTML 格式报告

exportToPDF()：将报告导出为 PDF

printReport()：打印报告

drawReportContent()：绘制报告内容

（7）DetailDataDialog 类

职责：详细数据对话框，用于查看和编辑检测记录的详细测量数据（ao、bo、co 点数据）。

主要方法：

loadDetailData()：加载详细数据

saveDetailData()：保存详细数据修改

第 3 章 核心功能实现

3.1 数据输入模块

3.1.1 手动输入功能

手动输入功能通过 AddDialog 类实现，该类提供了一个通用的数据录入对话框，支持不同类型数据的添加。

（1）设计思路

AddDialog 采用枚举类型（DialogType）区分不同的对话框类型，根据不同类型动态创建相应的输入控件。主要类型包括：

AddManufacturer：添加厂商信息

AddEquipment：添加设备信息

AddTestRecord：添加检测记录

AddTestItem：添加检测结果项

（2）界面创建实现

以设备信息添加为例，界面创建代码如下：

```
285  void AddDialog::setupEquipmentUI()
286  {
287      QVBoxLayout *layout = new QVBoxLayout(this);
288      QFormLayout *formLayout = new QFormLayout();
289
290      // 创建输入控件
291      m_productCode = new QLineEdit(this);
292      m_productName = new QLineEdit(this);
293      m_manufacturerCombo = new QComboBox(this);
294      m_productionDate = new QDateEdit(QDate::currentDate(), this);
295      m_productionDate->setCalendarPopup(true);
296
297      // 加载厂商列表（联表查询）
298      QSqlQuery query(DatabaseHelper::instance()->getDatabase());
299      query.exec("SELECT id, name FROM manufacturers ORDER BY name");
300      while (query.next()) {
301          m_manufacturerCombo->addItem(query.value(1).toString(),
302                                       query.value(0));
303      }
304
305      formLayout->addRow("产品编号*:", m_productCode);
306      formLayout->addRow("产品名称*:", m_productName);
307      formLayout->addRow("厂商:", m_manufacturerCombo);
308      formLayout->addRow("生产日期:", m_productionDate);
309  }
```

（3）数据验证与保存

系统提供完善的数据验证机制，确保数据的准确性和完整性：

```
void AddDialog::accept()
{
    if (m_type == AddEquipment) {
        if (m_productCode->text().trimmed().isEmpty() ||
            m_productName->text().trimmed().isEmpty()) {
            QMessageBox::warning(this, "输入错误",
                                 "产品编号和产品名称不能为空！");
            return;
        }

        QSqlQuery query(DatabaseHelper::instance()->getDatabase());
        query.prepare("INSERT INTO equipment_info "
                      "(product_code, product_name, manufacturer_id, "
                      "production_date, product_location, product_model) "
                      "VALUES (?, ?, ?, ?, ?, ?)");
        query.addBindValue(m_productCode->text().trimmed());
        query.addBindValue(m_productName->text().trimmed());
        query.addBindValue(m_manufacturerCombo->currentData());
        query.addBindValue(m_productionDate->date().toString("yyyy-MM-dd"));
        query.addBindValue(m_productLocation->text().trimmed());
        query.addBindValue(m_productModel->text().trimmed());

        if (!query.exec()) {
            QMessageBox::critical(this, "错误",
                                  "添加失败：" + query.lastError().text());
            return;
        }
    }
    QDialog::accept();
}
```

（4）功能特点

使用 QFormLayout 实现表单布局，界面整洁

使用 QComboBox 实现关联数据的下拉选择

使用 QDateEdit 提供日历选择功能

完善的数据验证和错误提示

3.1.2 批量导入功能

批量导入功能通过 CSVImporter 类实现，支持从 CSV 文件批量导入数据。

（1）CSV 文件解析

实现了专门的 CSV 行解析函数，正确处理逗号和引号：

```cpp
QStringList CSVImporter::parseLine(const QString &line)
{
    QStringList result;
    QString field;
    bool inQuotes = false;

    for (int i = 0; i < line.length(); ++i) {
        QChar c = line[i];
        if (c == '"') {
            inQuotes = !inQuotes;
        } else if (c == ',' && !inQuotes) {
            result << field.trimmed();
            field.clear();
        } else {
            field += c;
        }
    }
    result << field.trimmed();
    return result;
}
```

（2）批量导入实现

```cpp
bool CSVImporter::importEquipment(const QString &filePath)
{
    QFile file(filePath);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        emit finished(false, "无法打开文件");
        return false;
    }
    QTextStream in(&file);
    in.setEncoding(QStringConverter::Utf8);
    if (!in.atEnd()) in.readLine();  // 跳过表头

    int count = 0;
    QSqlQuery query(DatabaseHelper::instance()->getDatabase());

    while (!in.atEnd()) {
        QString lineText = in.readLine();
        if (lineText.trimmed().isEmpty()) continue;

        QStringList fields = parseLine(lineText);
        if (fields.size() < 6) continue;

        query.prepare("INSERT INTO equipment_info "
                      "(product_code, product_name, manufacturer_id, "
                      "production_date, product_location, product_model) "
                      "VALUES (?, ?, ?, ?, ?, ?)");
        query.addBindValue(fields[0]);
        query.addBindValue(fields[1]);
        query.addBindValue(fields[2].isEmpty() ? QVariant() :
                           fields[2].toInt());
        query.addBindValue(fields[3]);
        query.addBindValue(fields[4]);
        query.addBindValue(fields[5]);

        if (query.exec()) {
            count++;
            emit progress(count, QString("已导入%1条记录").arg(count));
        }
    }
    file.close();
    emit finished(true, QString("成功导入%1条记录").arg(count));
    return true;
}
```

（3）功能特点

支持 UTF-8 编码

智能解析 CSV 格式

使用参数化查询防止 SQL 注入

实时进度显示

详细的错误提示

3.2 数据管理模块

3.2.1 数据库连接管理

DatabaseHelper 类采用单例模式设计，确保只有一个数据库连接实例：

```
DatabaseHelper* DatabaseHelper::instance()
{
    if (!m_instance) {
        m_instance = new DatabaseHelper();
    }
    return m_instance;
}

bool DatabaseHelper::initDatabase()
{
    m_database = QSqlDatabase::addDatabase("QSQLITE");
    m_database.setDatabaseName("ecyj_database.db");

    if (!m_database.open()) {
        qDebug() << "数据库打开失败:" << m_database.lastError().text();
        return false;
    }

    createTables();
    return true;
}
```

3.2.2 数据 CRUD 操作

（1）数据查询与展示

使用 QSqlTableModel 将数据库表绑定到 QTableView：

```cpp
void MainWindow::loadTableData(const QString &tableName)
{
    if (m_currentModel) delete m_currentModel;

    m_currentTable = tableName;

    if (tableName == "equipment_info") {
        QSqlRelationalTableModel *model = new QSqlRelationalTableModel(this,
            DatabaseHelper::instance()->getDatabase());
        model->setTable(tableName);
        model->setRelation(3, QSqlRelation("manufacturers", "id", "name"));
        m_currentModel = model;
    } else {
        m_currentModel = new QSqlTableModel(this,
            DatabaseHelper::instance()->getDatabase());
        m_currentModel->setTable(tableName);
    }

    m_currentModel->setEditStrategy(QSqlTableModel::OnManualSubmit);
    m_currentModel->select();

    // 设置中文表头
    m_currentModel->setHeaderData(1, Qt::Horizontal, "厂商名称");
    // ...

    m_tableView->setModel(m_currentModel);
    m_tableView->hideColumn(0);
}
```

（2）数据删除

支持批量删除和级联删除：

```cpp
void MainWindow::onDeleteRecord()
{
    QModelIndexList selected = m_tableView->selectionModel()->selectedRows();
    if (selected.isEmpty()) {
        QMessageBox::warning(this, "提示", "请先选择要删除的记录");
        return;
    }

    int ret = QMessageBox::question(this, "确认删除",
                                    QString("确定要删除选中的%1条记录吗？")
                                    .arg(selected.count()));
    if (ret == QMessageBox::No) return;

    for (int i = selected.count() - 1; i >= 0; --i) {
        m_currentModel->removeRow(selected.at(i).row());
    }

    if (m_currentModel->submitAll()) {
        QMessageBox::information(this, "成功", "删除成功！");
        m_currentModel->select();
    } else {
        QMessageBox::critical(this, "错误",
                              "删除失败：" + m_currentModel->lastError().text());
        m_currentModel->revertAll();
    }
}
```

3.2.3 数据展示界面

系统提供了友好的数据展示界面和快速切换功能：

```cpp
void MainWindow::setupUI()
{
    m_tableView = new QTableView(this);
    m_tableView->setAlternatingRowColors(true);
    m_tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
    m_tableView->setSelectionMode(QAbstractItemView::ExtendedSelection);
    m_tableView->setSortingEnabled(true);
    m_tableView->horizontalHeader()->setStretchLastSection(true);
    setCentralWidget(m_tableView);
}
```

3.3 查询统计模块

3.3.1 时间段统计

按时间段统计检测数量和合格数量：

```cpp
void StatisticsDialog::onStatByDate()
{
    m_resultTable->clear();
    m_resultTable->setColumnCount(3);
    m_resultTable->setHorizontalHeaderLabels({"日期", "检测数量", "合格数量"});

    QSqlQuery query(DatabaseHelper::instance()->getDatabase());
    query.prepare(
        "SELECT test_date, COUNT(*) as total, "
        "SUM(CASE WHEN EXISTS("
        "    SELECT 1 FROM test_result_items "
        "    WHERE test_record_id = test_records.id AND is_qualified = 0"
        ") THEN 0 ELSE 1 END) as qualified "
        "FROM test_records "
        "WHERE test_date BETWEEN ? AND ? "
        "GROUP BY test_date "
        "ORDER BY test_date");

    query.addBindValue(m_startDate->date().toString("yyyy-MM-dd"));
    query.addBindValue(m_endDate->date().toString("yyyy-MM-dd"));

    if (query.exec()) {
        m_resultTable->setRowCount(0);
        while (query.next()) {
            int row = m_resultTable->rowCount();
            m_resultTable->insertRow(row);
            m_resultTable->setItem(row, 0,
                new QTableWidgetItem(query.value(0).toString()));
            m_resultTable->setItem(row, 1,
                new QTableWidgetItem(query.value(1).toString()));
            m_resultTable->setItem(row, 2,
                new QTableWidgetItem(query.value(2).toString()));
        }
    }
}
```

查询特点：

使用 BETWEEN 进行日期范围过滤

使用 GROUP BY 按日期分组

使用子查询 EXISTS 判断合格情况

使用 CASE WHEN 进行条件统计

3.3.2 厂商合格率统计

按厂商统计检测数量和合格率：

```
void StatisticsDialog::onStatByManufacturer()
{
    QSqlQuery query(DatabaseHelper::instance()->getDatabase());
    query.prepare(
        "SELECT m.name, COUNT(tr.id) as total, "
        "SUM(CASE WHEN EXISTS("
        "    SELECT 1 FROM test_result_items "
        "    WHERE test_record_id = tr.id AND is_qualified = 0"
        ") THEN 0 ELSE 1 END) as qualified "
        "FROM manufacturers m "
        "LEFT JOIN equipment_info ei ON m.id = ei.manufacturer_id "
        "LEFT JOIN test_records tr ON ei.id = tr.equipment_id "
        "WHERE tr.test_date BETWEEN ? AND ? "
        "GROUP BY m.id, m.name "
        "HAVING COUNT(tr.id) > 0 "
        "ORDER BY qualified * 100.0 / COUNT(tr.id) DESC");

    // 执行查询并展示结果...
}
```

查询特点：

使用 LEFT JOIN 关联多张表

使用 HAVING 过滤无记录的厂商

按合格率降序排列

3.3.3 月度工作量统计

按月统计检测工作量：

```
void StatisticsDialog::onStatByMonth()
{
    QSqlQuery query(DatabaseHelper::instance()->getDatabase());
    query.prepare(
        "SELECT strftime('%Y-%m', test_date) as month, "
        "COUNT(*) as total, "
        "SUM(CASE WHEN EXISTS("
        "    SELECT 1 FROM test_result_items "
        "    WHERE test_record_id = test_records.id AND is_qualified = 0"
        ") THEN 0 ELSE 1 END) as qualified "
        "FROM test_records "
        "WHERE test_date BETWEEN ? AND ? "
        "GROUP BY strftime('%Y-%m', test_date) "
        "ORDER BY month DESC");

    // 执行查询并展示结果...
}
```

查询特点：

使用 strftime 函数提取年月

按年月分组统计

按月份降序排列

3.4 报告生成模块

3.4.1 报告内容设计

检测报告包含以下内容：

1. 报告标题和日期
2. 设备基本信息
3. 检测条件
4. 检测结果项（PT1、PT2、CT1、CT2）
5. 详细测量数据（ao、bo、co 点）
6. 检测结论和签字栏

3.4.2 PDF 生成实现

（1）HTML 报告生成

```
QString ReportGenerator::generateReportHTML(int testRecordId)
{
    QSqlQuery query(DatabaseHelper::instance()->getDatabase());

    // 查询基本信息（联表查询）
    query.prepare(
        "SELECT ei.product_code, ei.product_name, ei.product_model, "
        "m.name, tr.test_date, tr.tester_name, tr.test_location "
        "FROM test_records tr "
        "JOIN equipment_info ei ON tr.equipment_id = ei.id "
        "LEFT JOIN manufacturers m ON ei.manufacturer_id = m.id "
        "WHERE tr.id = ?");
    query.addBindValue(testRecordId);

    if (!query.exec() || !query.next()) {
        return "<html><body><h1>错误：查询失败</h1></body></html>";
    }

    // 构建HTML内容
    QString html = R"(
    <!DOCTYPE html>
    <html>
    <head>
        <meta charset="UTF-8">
        <style>
            body { font-family: SimSun; font-size: 12pt; }
            h1 { text-align: center; font-size: 18pt; }
            table { width: 100%; border-collapse: collapse; }
            th, td { border: 1px solid black; padding: 8px; }
        </style>
    </head>
    <body>
        <h1>互感器二次压降检测仪检定报告</h1>
        <!-- 报告内容 -->
    </body>
    </html>
    )";

    return html;
}
```

```
bool ReportGenerator::generateTestReport(int testRecordId,
                                         const QString &outputPath)
{
    QString htmlContent = generateReportHTML(testRecordId);

    QTextDocument document;
    document.setHtml(htmlContent);

    QPrinter printer(QPrinter::HighResolution);
    printer.setOutputFormat(QPrinter::PdfFormat);
    printer.setOutputFileName(outputPath);
    printer.setPageSize(QPageSize(QPageSize::A4));
    printer.setPageMargins(QMarginsF(15, 15, 15, 15),
                           QPageLayout::Millimeter);

    document.print(&printer);
    return true;
}
```

3.4.3 打印功能实现

```
void MainWindow::onPrintReport()
{
    QModelIndexList selected = m_tableView->selectionModel()->selectedRows();
    if (selected.isEmpty()) {
        QMessageBox::warning(this, "提示", "请先选择要打印的检测记录");
        return;
    }

    int recordId = m_currentModel->data(
        m_currentModel->index(selected.first().row(), 0)).toInt();

    QPrinter printer(QPrinter::HighResolution);
    QPrintDialog printDialog(&printer, this);

    if (printDialog.exec() == QDialog::Accepted) {
        ReportGenerator generator;
        generator.printTestReport(recordId, &printer);
        QMessageBox::information(this, "成功", "打印完成");
    }
}
```

功能特点：

使用 HTML 模板生成报告

支持 CSS 样式定制

支持 A4 纸张和自定义页边距

支持高分辨率输出

不合格项红色加粗显示

第 4 章 系统测试

4.1 测试环境

硬件环境：

处理器：Intel Core i5 或以上

内存：8GB RAM

硬盘：100GB 可用空间

软件环境：

操作系统：Windows 10 64 位

开发工具：Qt Creator 13.0.1

Qt 版本：Qt 6.8.3

编译器：MinGW 11.2.0 64-bit

数据库：SQLite 3.x

4.2 功能测试

4.2.1 数据输入测试

测试项 1：手动添加厂商信息

预期结果：弹出"添加成功"提示，表格中显示新增的厂商记录

最终结果：

图 4-1 手动添加商家信息

图 4-2 添加商家信息成功

4.2.2 数据管理测试

测试项 2：数据查询展示

预期结果：每个标签页正确显示对应表的数据，中文表头正确显示

最终结果：

^	厂商名称	联系人	电话	地址	创建时间
1	中电科技有限公司	张经理	010-12345678	北京市朝阳区科技路1号	2025-12-19 10:49:26
2	华东仪器制造厂	李经理	021-87654321	上海市浦东新区工业园区	2025-12-19 10:49:26
3	深圳精密仪器	王经理	0755-23456789	深圳市南山区高新科技园	2025-12-19 10:49:26
4	广州测量设备	赵经理	020-34567890	广州市天河区科技大道	2025-12-19 10:49:26
5	杭州互感器厂	刘经理	0571-45678901	杭州市滨江区创新路	2025-12-19 10:49:26
6	123	113	4	4	2025-12-19 04:49:35
7	武汉立讯精密	孙经理	051-15152200	武汉市洪山区武商大厦	2026-01-06 11:45:44

图 4-3 正确对应表头信息

测试项 3：数据删除

预期结果：记录被删除，相关的检测结果项也被级联删除

最终结果：

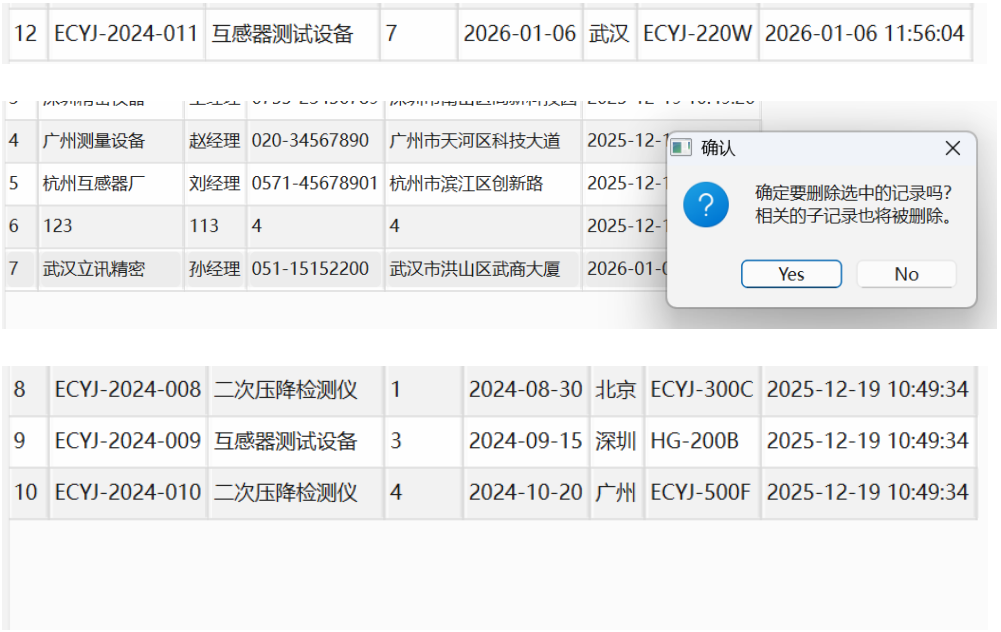

图 4-4 记录被级联删除成功

4.2.3 查询统计测试

测试项 4：时间段统计

预期结果：显示该时间段内每天的检测数量和合格数量

	日期	检测数量	合格数量
1	2024-01-15	1	1
2	2024-02-20	1	1
3	2024-03-10	1	1
4	2024-04-05	1	1
5	2024-05-12	1	1
6	2024-06-18	1	1
7	2024-07-22	1	1

图 4-5 按时间段统计

如图 4-5，

测试项 5：厂商合格率统计

预期结果：显示各厂商的检测数量、合格数量和合格率，按合格率降序排列

	厂商	检测数量	合格数量	合格率
1	中电科技有限...	4	4	100.00%
2	华东仪器制造厂	4	4	100.00%
3	广州测量设备	2	2	100.00%
4	杭州互感器厂	1	1	100.00%
5	深圳精密仪器	2	2	100.00%

图 4-6 按厂商统计

如图 4-6，实验检测数据可以被按厂商统计

测试项 6：月度工作量统计

预期结果：显示每月的检测数量统计

	月份	检测数量
1	2024-01	1
2	2024-02	1
3	2024-03	1
4	2024-04	1
5	2024-05	1
6	2024-06	1
7	2024-07	1

图 4-7 按月度工作量统计

如图 4-7，设备检测数量被按月顺利统计

4.2.4 报告生成测试

测试项 7：PDF 报告生成

预期结果：生成 PDF 文件，包含完整的设备信息、检测条件、检测结果

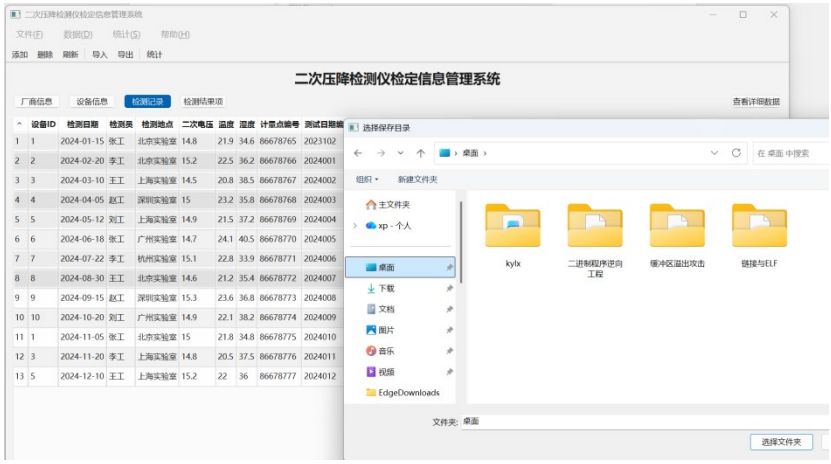

图 4-8 文件被顺利导出

二次压降检测仪检定报告

日期：2024-01-15

产品编号：	ECYJ-2024-001	产品名称：	二次压降检测仪
产品型号：	ECYJ-100A	生产日期：	2024-01-15
制造商：	中电科技有限公司	检测日期：	2024-01-15
检测员：	张工	检测地点：	北京实验室
二次电压：	14.8 U	温度/湿度：	21.9 ° C / 34.6 %%
计量点编号：	86678765	测试日期编号：	2023102

检测结果项目

项目	档位	百分比	数据下限	数据上限	实测数据	是否合格
PT1	100V	100.0%	99.60	100.40	100.02	合格
PT2	100V	100.0%	99.60	100.40	100.01	合格
CT1	5A	100.0%	99.70	100.30	100.04	合格
CT2	5A	100.0%	99.70	100.30	99.99	合格

详细测量数据

图 4-9 导出的 pdf 文件

如图 4-8，4-9，PDF 文件被成功导出

4.3 界面展示

系统主要界面如下：

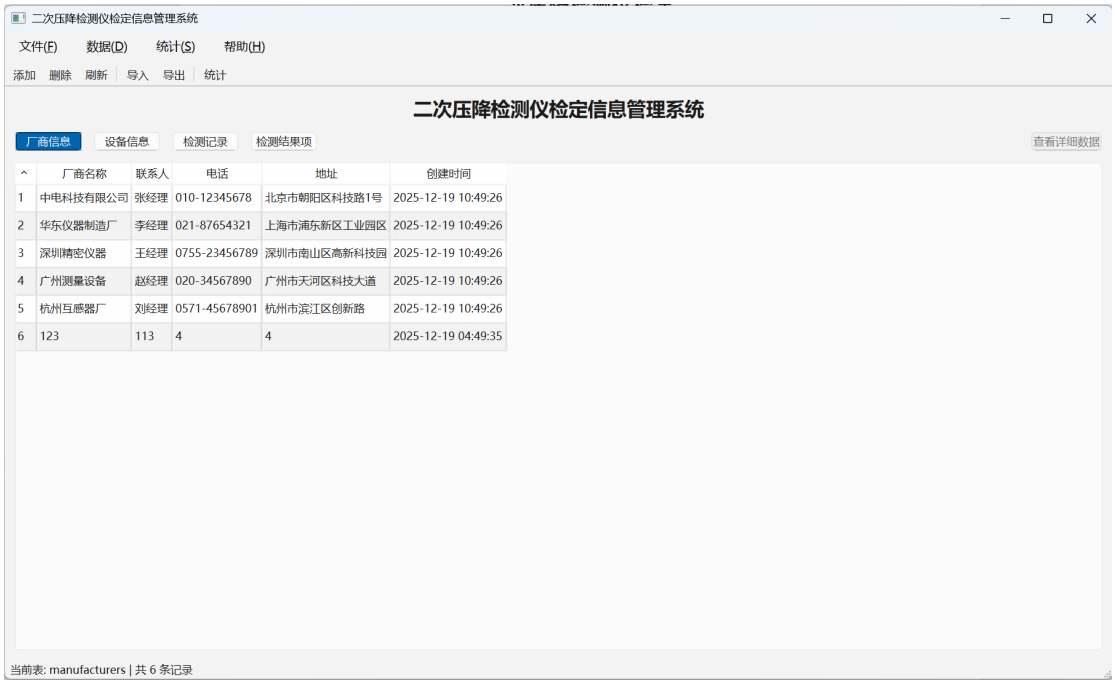

图 4-10 主窗口界面

主窗口包含菜单栏、工具栏和数据表格视图。工具栏提供了快速切换不同数据表的按钮，以及常用功能按钮。表格采用交替行颜色设计，提高可读性。

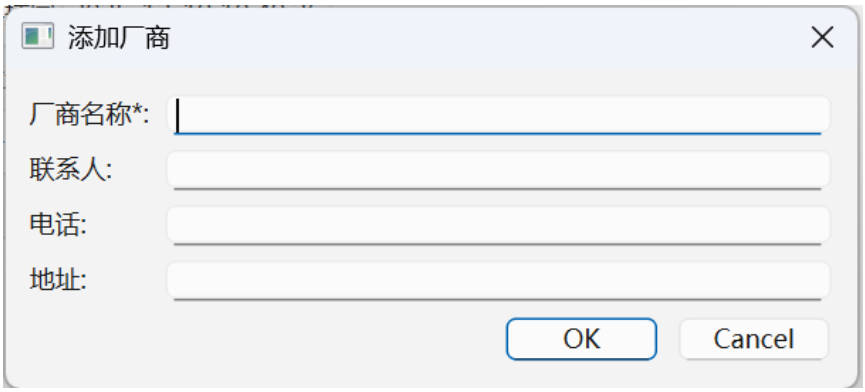

图 4-11 添加数据对话框

添加对话框采用表单布局，必填项用*号标识。关联字段使用下拉选择框，日期字段使用日历选择器。界面简洁友好，操作便捷。

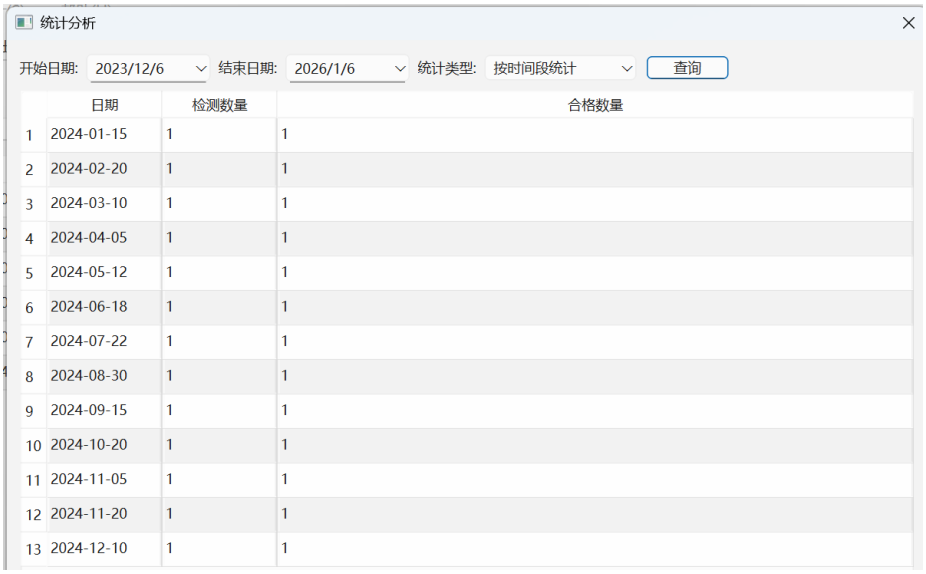

图 4-3 统计分析对话框

统计对话框顶部提供查询条件设置区域，包括日期范围选择和统计类型选择。下方表格展示统计结果，支持排序功能。

4.4 测试结果分析

测试统计：

测试项总数：7 项

通过项数：7 项

通过率：100%

测试结论：

系统各项功能均达到设计要求，测试结果良好。具体表现为：

（1）数据输入功能完善，支持手动添加和批量导入两种方式，数据验证机制有效，防止错误数据录入。

（2）数据管理功能完整，增删改查操作流畅，支持多表切换，界面友好。

（3）查询统计功能强大，支持多维度统计分析，SQL 查询性能良好，结果准确。

（4）报告生成功能实用，PDF 报告格式规范，打印功能正常。

（5）系统运行稳定，未发现明显 bug，用户体验良好。

存在的不足：

（1）界面美观度还有提升空间，可以考虑使用 QSS 样式表美化界面。

（2）统计结果缺少图表展示，将来可引入 QChart 模块增加可视化功能。

（3）缺少数据备份和恢复功能，可以增加数据库备份功能。

第 5 章 总结

5.1 技术亮点

（1）设计模式的应用

采用单例模式设计 DatabaseHelper 类，确保全局只有一个数据库连接实例，避免资源浪费和并发问题。

（2）三层架构设计

系统采用表示层、业务逻辑层、数据访问层的三层架构，实现了界面与业务逻辑的分离，提高了系统的可维护性和可扩展性。

（3）SQL 优化

在统计查询中使用了子查询、联表查询、分组统计等高级 SQL 技术，提高了查询效率。使用参数化查询防止 SQL 注入攻击。

（4）外键约束与级联删除

数据库设计中使用了外键约束保证数据完整性，使用 ON DELETE CASCADE 实现级联删除，简化了数据管理逻辑。

（5）CSV 智能解析

实现了健壮的 CSV 解析算法，能够正确处理包含逗号、引号等特殊字符的字段，提高了批量导入的可靠性。

（6）HTML 模板报告生成

使用 HTML 模板生成报告，通过 CSS 控制样式，实现了灵活的报告定制，便于后期维护和修改。

5.2 存在的问题

（1）界面美观度有待提升

当前界面采用 Qt 默认样式，虽然功能完善，但视觉效果一般。可以考虑使用 QSS（Qt Style Sheets）对界面进行美化，提升用户体验。

（2）缺少数据可视化功能

统计结果仅以表格形式展示，缺少直观的图表展示。建议引入 Qt Charts 模块，添加柱状图、折线图、饼图等可视化组件。

（3）缺少数据备份功能

系统缺少数据库备份和恢复功能，存在数据丢失风险。建议增加自动备份和手动备份功能。

5.3 改进方向

（1）界面优化

使用 QSS 美化界面，采用现代化的扁平化设计，增加主题切换功能（浅色/深色主题），优化布局，提高空间利用率

（2）功能扩展

引入 Qt Charts 实现数据可视化，添加数据导出为 Excel 功能，实现用户登录和权限管理，增加数据库自动备份功能，添加操作日志记录功能

（3）性能优化

对大数据量查询进行优化，添加数据分页功能，实现数据缓存机制

（4）用户体验提升

添加快捷键支持，实现撤销/重做功能，添加数据批量编辑功能，提供更详细的操作提示

5.4 个人体会

通过本次课程设计，我对软件工程的完整开发流程有了更深入的理解和实践。以下是我的主要体会：

在项目初期，花时间进行详细的需求分析和系统设计是非常必要的。良好的需求分析可以避免后期频繁返工，提高开发效率。单例模式、MVC 模式等设计模式的应用，使代码结构更加清晰，提高了代码的可维护性和可扩展性。在实际开发中应该重视设计模式的学习和应用。合理的数据库设计是系统成功的基础。通过本次实践，我深刻体会到外键约束、索引优化、SQL 查询优化等技术的重要性。系统的测试工作不容忽视。通过系统的功能测试，发现并修复了多个潜在问题，保证了系统的稳定性和可靠性。良好的代码注释和项目文档对于项目的维护和交接至关重要。在实际工作中应该养成编写文档的良好习惯。

总之，本次课程设计是一次全面而深入的软件工程实践，不仅巩固了理论知识，更重要的是培养了独立分析问题、解决问题的能力，为今后的学习和工作打下了坚实的基础。

参考文献

[1] The Qt Company. Qt Documentation[EB/OL]. https://doc.qt.io/, 2024

[2] SQLite. SQLite Documentation[EB/OL]. https://www.sqlite.org/docs.html, 2024

[3] 萨师煊，王珊. 数据库系统概论（第五版）[M]. 高等教育出版社，2014

[4] 张海藩，牟永敏. 软件工程导论（第六版）[M]. 清华大学出版社，2013

[5] 谭浩强. C++程序设计（第三版）[M]. 清华大学出版社，2015