



# 华中科技大学

## 数据库系统原理实践报告

综合设计题目： 租车信息管理系统

姓 名：

学 院： 计算机科学与技术学院

专 业： 物联网工程

班 级： IOT1601

学 号：

指导教师：

分数	
教师签名	

2018 年 6 月 14 日



## 目 录

<b>1 课程任务书 .....</b>	<b>1</b>
<b>2 软件功能学习部分 .....</b>	<b>6</b>
2.1 任务要求.....	6
2.2 完成过程.....	6
2.2.1 环境配置.....	6
2.2.2 数据备份.....	6
2.2.2 添加用户并配置权限.....	8
2.3 任务总结.....	8
<b>3 Sql 练习部分.....</b>	<b>9</b>
3.1 任务要求.....	9
3.2 完成过程.....	11
3.2.1 创建表格.....	11
3.2.2 插入数据.....	13
3.2.3 数据更新.....	15
2.2.4 数据查询.....	20
3.2.5 了解系统的查询性能分析功能（选做） .....	27
3.2.6 DBMS 函数及存储过程和事务（选做） .....	28
3.3 任务总结.....	28
<b>4 数据库应用系统设计.....</b>	<b>29</b>
4.1 系统设计目标.....	29
4.2 需求分析.....	29
4.2.1 任务文档要求.....	29
4.2.2 需求分析内容.....	29
4.3 总体设计.....	30
4.3.1 系统体系结构.....	30
4.3.2 总体功能模块划分.....	30
4.4 数据库设计.....	31
4.5 数据流图 and 业务流程图.....	34
4.6 详细设计与实现.....	37
4.6.1 运行环境.....	37
4.6.2 前端主要组件.....	37
4.6.3 数据库操作.....	41
4.7 系统测试.....	42
4.8 系统设计与实现总结.....	52

附录: .....54

    代码托管: .....54

    MainGUI.java: .....54

    DataBase.java: .....80

    DataNameUtils.java:.....93

## 1 课程任务书

### 1 软件功能学习部分（各教师可适当调整）

完成下列 1~2 题，并在实践报告中叙述过程，可适当辅以插图（控制在 A4 三页篇幅以内）

1) 练习 sqlserver 的两种完全备份方式：数据和日志文件的脱机备份、系统的备份功能。

2) 练习在新增的数据库上增加用户并配置权限的操作。

### 2 Sql 练习部分（各教师每年可适当调整）

#### 2.1 建表

##### 1) 创建下列跟电影相关的关系，包括主码和外码的说明

**电影表【电影编号，电影名称，电影类型，导演姓名，电影时长（以分钟计），是否 3D，用户评分】**

FILM(FID int, FNAME char(30), FTYPE char(10), DNAME char(30), length int, IS3D char(1), GRADE int)。

主码为电影编号，IS3D 取值为‘Y’表示是 3D 电影，‘N’表示不是，用户评分规定为 0~100 分之间或者为空值。

**演员表【演员编号，演员姓名，性别，出生年份】**

ACTOR(ACTID int, ANAME char(30), SEX char(2), BYEAR int)

主码为演员编号

**参演表【演员编号，电影编号，是否主角，用户对该演员在该电影中的评分】**

ACTIN(ACTID int, FID int, ISLEADING char(1), GRADE int)

主码、外码请依据应用背景合理定义。ISLEADING 取值为‘Y’表示是，‘N’表示不是主角，也可能取空值，表示不太确定该演员在该电影中是否主角。GRADE 规定为 0~100 分之间或者为空值。

**电影院表【电影院编号，电影院名字，影院所在行政区，影院地址】**

THEATER (TID int, TNAME char(20), TAREA char(20), ADDRESS char(30))

主码为电影院编号，影院所在行政区取值如“洪山区”、“武昌区”等等。

**上映表【电影编号，影院编号，上映年份，上映月份】**

SHOW(FID int, TID int, PRICE int, YEAR int, MONTH int)

假定一部电影在一家影院只上映一次，主码、外码请依据应用背景合理定义。

## 2) 观察性实验

验证在建立外码时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

## 3) 数据准备

依据后续实验的要求，向上述表格中录入适当数量的实验数据，从而对相关的实验任务能够起到验证的作用。

## 2.2 数据更新

1) 分别用一条 sql 语句完成对电影表基本的增、删、改的操作；

2) 批处理操作

将演员表中的 90 后演员记录插入到一个新表 YOUNG\_ACTOR 中。

3) 数据导入导出

通过查阅 DBMS 资料学习数据导入导出功能，并将任务 2.1 所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

## 4) 观察性实验

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

## 5) 创建视图

创建一个有 80 后演员作主角的参演记录视图，其中的属性包括：演员编号、演员姓名、出生年份、作为主角参演的电影数量、这些电影的用户评分的最高分。

## 6) 触发器实验

编写一个触发器，用于实现对电影表的完整性控制规则：当增加一部电影时，若导演的姓名为周星驰，则电影类型自动设置为“喜剧”。

## 2.3 查询

请分别用一条 SQL 语句完成下列各个小题的查询需求：

1) 查询“战狼”这部电影在洪山区各家影院的 2017 年的上映情况，并按照上映的月份的降序排列；

2) 查询所有无参演演员信息的电影的基本信息，并且将结果按照电影类型的升序排列，相同类型的电影则按照用户评分的降序排列；

3) 查询所有直到 2017 年仍未上映的电影编号、电影名称、导演姓名；

4) 查询在每家电影院均上映过的电影编号；

5) 查询所有用户评分低于 80 分或者高于 89 分的电影编号、电影名称、导

演姓名及其用户评分，要求 where 子句中只能有一个条件表达式；

- 6) 查询每个导演所执导的全部影片的最低和最高用户评分；
- 7) 查询至少执导过 2 部电影的导演姓名、执导电影数量；
- 8) 查询至少 2 部电影的用户评分超过 80 分的导演及其执导过的影片数量、平均用户评分；
- 9) 查询至少执导过 2 部电影的导演姓名以及跟这些导演合作过的演员编号、姓名；
- 10) 查询每个演员担任主角的电影中的平均用户评分；
- 11) 查询用户评分超过 90 分的电影的最早上映年月；
- 12) 查询用户评分超过 90 分的电影的最早上映年月及其相应的上映影院编号；
- 13) 查询每个电影的上映总次数；
- 14) 查询执导过动作片，或者警匪片，或者枪战片的导演的姓名，要求 where 子句中只能有一个条件表达式；
- 15) 查询所有“战狼”系列的电影的编号、电影名称、上映电影院名称及其上映年月，结果按照电影名称的升序排列；
- 16) 查询在同一个年月上映 1 号和 2 号电影的影院编号；
- 17) 查询所有没参演过用户评分 85 分以下电影的演员的编号、姓名；
- 18) 查询参演过“吴宇森”执导过的所有电影的演员姓名；
- 19) 查询所有的演员的编号、姓名及其参演过的电影名称，要求即使该演员未参演过任何电影也要能够输出其编号、姓名；
- 20) 查询所有上映超过 3 次但没有用户评分的电影编号、名称。

#### 2.4 了解系统的查询性能分析功能（选做）

选择上述 2.3 任务中某些较为复杂的 SQL 语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

#### 2.5 DBMS 函数及存储过程和事务（选做）

- 1) 通过系统帮助文档学习系统关于时间、日期、字符串类型的函数，为电影表增加首映时间属性，然后查询下个月首映的电影信息。
- 2) 编写一个依据演员编号计算在其指定年份参演的电影数量的自定义的函数，并利用其查询 2017 年至少参演过 5 部电影的演员编号。
- 3) 尝试编写 DBMS 的存储过程，建立每家影院的上映电影总数的统计表，并通过存储过程更新该表。
- 4) 尝试在 DBMS 的交互式界面中验证事务机制的执行效果。

### 3 数据库应用系统设计

自行选择所擅长的 DBMS 软件以及数据库应用系统（客户端程序或者网

站)的程序开发工具,参考后面的题目例子,拟定一个自己感兴趣的数据库应用系统题目,完成该小型数据库应用系统的设计与实现工作。主要包括:需求调研与分析、总体设计、数据库设计、详细设计与实现、测试等环节的工作。

下列题目作为选题背景参考,也可依据这些题目拟定一个自己感兴趣的具有类似工作量和复杂程度的课题。

### 题目 3: 汽车租赁信息系统

采用 B/S 或 C/S 模式实现一个汽车租赁信息系统。完成用户、车辆、经手员工、租借情况、车辆损毁情况、交通违规罚款等信息的管理。

要求:

- 1) 实现不同权限的浏览和更新。
- 2) 能够根据车辆使用情况计算押金退还金额。
- 3) 能查询客户的租借历史记录,并进行信誉度评价,进行会员制和非会员制的客户管理。
- 4) 能够管理车辆报修信息;
- 5) 能够生成租借公司的日、月、季度、年财务报表。

## 4 撰写课程实践报告

在课设规定的时间内,撰写并完成课程实践报告。

实践报告由 5 章组成,依次对应下列内容:

### 1 课程任务概述

简要陈述介绍本实践课程的各项任务要求。

### 2 软件功能学习

阐述第 1 部分任务的完成过程。

### 3 SQL 练习

阐述第 2 部分的完成过程。

### 4 应用系统设计

阐述第 3 部分的完成过程。

### 5 课程总结

逐条概括、总结此次课程实践的主要工作,阐述此次课程实践的心得体会,展望此次课程实践的有待改进和完善的工作。

其中,第 4 章进一步分 7 个小节,依次是:

#### 4.1 系统设计目标

#### 4.2 需求分析

#### 4.3 总体设计



#### 4.4 数据库设计

#### 4.5 详细设计与实现

#### 4.6 系统测试

#### 4.7 系统设计与实现总结

### 5 提交书面报告和电子资料

以班为单位刻盘提交，每人一个压缩文件，文件名格式“班号\_姓名\_应用程序名.rar”，压缩文件内含 1 个文件（课程实践报告 word 版）和 3 个目录：1）第 2 部分源代码；2）第 3 部分应用程序源代码；3）第 3 部分应用程序可执行文件、数据库文件、以及程序使用说明 word 文档。

### 6 关于附录

实践报告要统一有附录，里面包含去除冗余之后第 2 部分、第 3 部分核心功能的源代码。

## 2 软件功能学习部分

### 2.1 任务要求

完成下列 1~2 题，并在实践报告中叙述过程，可适当辅以插图（控制在 A4 三页篇幅以内）

- 1) 练习 sqlserver 的两种完全备份方式：数据和日志文件的脱机备份、系统的备份功能。
- 2) 练习在新增的数据库上增加用户并配置权限的操作。

### 2.2 完成过程

#### 2.2.1 环境配置

操作系统：Windows 10 1803，数据库：MySQL 8.0，数据库可视化工具：SQLyog

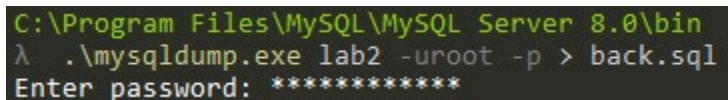
#### 2.2.2 数据备份

1) 数据与结构的逻辑备份 mysqldump

调用 mysqldump 程序来进行指定数据库中数据与结构向外部文本文件的保存过程。

执行命令：

`.\mysqldump.exe lab2 -uroot -p > back.sql`



```
C:\Program Files\MySQL\MySQL Server 8.0\bin
λ .\mysqldump.exe lab2 -uroot -p > back.sql
Enter password: *****
```

图 2-1 执行备份命令

```
-- MySQL dump 10.13 Distrib 8.0.11, for Win64 (x86_64)
--
-- Host: localhost    Database: lab2
--
-- Server version  8.0.11

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
SET NAMES utf8mb4 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `actin`
--

DROP TABLE IF EXISTS `actin`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `actin` (
  `ACTID` int(11) NOT NULL,
  `FID` int(11) NOT NULL,
  `ISLEADING` varchar(1) DEFAULT NULL,
  `GRADE` int(11) DEFAULT NULL,
  PRIMARY KEY (`ACTID`,`FID`),
  KEY `FID` (`FID`),
  CONSTRAINT `actin_ibfk_1` FOREIGN KEY (`FID`) REFERENCES `film` (`fid`),
  CONSTRAINT `actin_ibfk_2` FOREIGN KEY (`ACTID`) REFERENCES `actor` (`actid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

图 2-2 导出的 SQL 文件

## 2)物理备份 xtrabackup

通过 xtrabackup 命令来进行物理备份，纯粹是基于文件上的备份，不需要进行类似逻辑转换的 SQL 转换的过程，与 mysqlhotcopy 相比，xtrabackup 支持更多的数据库引擎。

执行命令：

grant reload,lock tables,replication client,create tablespace,super on \*.\* to

'backup'@'%' identified by '123456';

flush privileges;

innobackupex --user=backup --password=123456 --defaults-file=/tmp/my.cnf

/data/backup/hotbackup/full --no-timestamp

innobackupex 中配置为 datadir = /data/mysql/data

innodb\_data\_home\_dir = /data/mysql/data/data1

innodb\_data\_file\_path = ibdata1:10M:autoextend

innodb\_log\_group\_home\_dir = /data/mysql/data

innodb\_log\_files\_in\_group = 2

innodb\_log\_file\_size = 536870912

## 2.2.2 添加用户并配置权限

1)创建用户

```
CREATE USER gray@localhost IDENTIFIED BY '1569348'
```

2)用户授权

```
GRANT ALL ON lab2.* TO gray@localhost;
```

3)查看当前所有的用户

```
SELECT HOST,USER,authentication_string FROM mysql.user
```

host	user	authentication_string	
localhost	gray	\$A\$005\$38dRQjA}:6_Q&lQ5E1DeUfzYjbmU4shIXaXEMljDDnEXUeMie6...	70B
localhost	mysql.infoschema	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	41B
localhost	mysql.session	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	41B
localhost	mysql.sys	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	41B
localhost	root	*56AF250F544FA596BF20A511D9966A78CDBF44E6	41B

图 2-3 用户信息

## 2.3 任务总结

1.数据库在进行不同方式备份的时候还需要考虑储存的引擎是否支持该种备份方法，在 MySQL 中，默认使用的是 InnoDB，能支持热温冷多种备份方式。

2. MySQL 将很多数据库自身的数据保存在了 mysql 这个数据库中，包括用户信息等。

3.通过本次实验熟悉了数据库的基础操作和基础结构，在实验过程中主要困扰的问题发生在在使用数据库不同的备份方法的时候。MySQL 自带的几种数据备份方式，都是基于 mysqldump 延伸的工具（mysqlpump, mysqldumpslow 等），没办法完成物理备份来保存日志文件。通过安装和配置 xtrabackup 的一系列过程才完成了最后的物理备份操作，但是 xtrabackup 只支持 InnoDB 和 XtraDB，而不能支持 MyISAM，和 mysqlhotcopy 相反。

### 3 Sql 练习部分

#### 3.1 任务要求

##### 3.1.1 建表

##### 1) 创建下列跟电影相关的关系，包括主码和外码的说明

**电影表**【电影编号，电影名称，电影类型，导演姓名，电影时长（以分钟计），是否 3D，用户评分】

FILM(FID int, FNAME char(30), FTYPE char(10), DNAME char(30), length int, IS3D char(1), GRADE int)。

主码为电影编号，IS3D 取值为‘Y’表示是 3D 电影，‘N’表示不是，用户评分规定为 0~100 分之间或者为空值。

**演员表**【演员编号，演员姓名，性别，出生年份】

ACTOR(ACTID int, ANAME char(30), SEX char(2), BYEAR int)

主码为演员编号

**参演表**【演员编号，电影编号，是否主角，用户对该演员在该电影中的评分】

ACTIN(ACTID int, FID int, ISLEADING char(1), GRADE int)

主码、外码请依据应用背景合理定义。ISLEADING 取值为‘Y’表示是，‘N’表示不是主角，也可能取空值，表示不太确定该演员在该电影中是否主角。GRADE 规定为 0~100 分之间或者为空值。

**电影院表**【电影院编号，电影院名字，影院所在行政区，影院地址】

THEATER (TID int, TNAME char(20), TAREA char(20), ADDRESS char(30))

主码为电影院编号，影院所在行政区取值如“洪山区”、“武昌区”等等。

**上映表**【电影编号，影院编号，上映年份，上映月份】

SHOW(FID int, TID int, PRICE int, YEAR int, MONTH int)

假定一部电影在一家影院只上映一次，主码、外码请依据应用背景合理定义。

##### 2) 观察性实验

验证在建立外码时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

##### 3) 数据准备

依据后续实验的要求，向上述表格中录入适当数量的实验数据，从而对相关的实验任务能够起到验证的作用。

### 3.1.2 数据更新

- 1) 分别用一条 sql 语句完成对电影表基本的增、删、改的操作;
- 2) 批处理操作

将演员表中的 90 后演员记录插入到一个新表 YOUNG\_ACTOR 中。

- 3) 数据导入导出

通过查阅 DBMS 资料学习数据导入导出功能, 并将任务 2.1 所建表格的数据导出到操作系统文件, 然后再将这些文件的数据导入到相应空表。

- 4) 观察性实验

建立一个关系, 但是不设置主码, 然后向该关系中插入重复元组, 然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

- 5) 创建视图

创建一个有 80 后演员作主角的参演记录视图, 其中的属性包括: 演员编号、演员姓名、出生年份、作为主角参演的电影数量、这些电影的用户评分的最高分。

- 6) 触发器实验

编写一个触发器, 用于实现对电影表的完整性控制规则: 当增加一部电影时, 若导演的姓名为周星驰, 则电影类型自动设置为“喜剧”。

### 3.1.3 查询

请分别用一条 SQL 语句完成下列各个小题的查询需求:

- 1) 查询“战狼”这部电影在洪山区各家影院的 2017 年的上映情况, 并按照上映的月份的降序排列;
- 2) 查询所有无参演演员信息的电影的基本信息, 并且将结果按照电影类型的升序排列, 相同类型的电影则按照用户评分的降序排列;
- 3) 查询所有直到 2017 年仍未上映的电影编号、电影名称、导演姓名;
- 4) 查询在各家电影院均上映过的电影编号;
- 5) 查询所有用户评分低于 80 分或者高于 89 分的电影编号、电影名称、导演姓名及其用户评分, 要求 where 子句中只能有一个条件表达式;
- 6) 查询每个导演所执导的全部影片的最低和最高用户评分;
- 7) 查询至少执导过 2 部电影的导演姓名、执导电影数量;
- 8) 查询至少 2 部电影的用户评分超过 80 分的导演及其执导过的影片数量、平均用户评分;
- 9) 查询至少执导过 2 部电影的导演姓名以及跟这些导演合作过的演员编号、姓名;
- 10) 查询每个演员担任主角的电影中的平均用户评分;
- 11) 查询用户评分超过 90 分的电影的最早上映年月;

- 12) 查询用户评分超过 90 分的电影的最早上映年月及其相应的上映影院编号;
- 13) 查询每个电影的上映总次数;
- 14) 查询执导过动作片, 或者警匪片, 或者枪战片的导演的姓名, 要求 where 子句中只能有一个条件表达式;
- 15) 查询所有“战狼”系列的电影的编号、电影名称、上映电影院名称及其上映年月, 结果按照电影名称的升序排列;
- 16) 查询在同一个月上映 1 号和 2 号电影的影院编号;
- 17) 查询所有没参演过用户评分 85 分以下电影的演员的编号、姓名;
- 18) 查询参演过“吴宇森”执导过的所有电影的演员姓名;
- 19) 查询所有的演员的编号、姓名及其参演过的电影名称, 要求即使该演员未参演过任何电影也要能够输出其编号、姓名;
- 20) 查询所有上映超过 3 次但没有用户评分的电影编号、名称。

#### 3.1.4 了解系统的查询性能分析功能（选做）

选择上述 2.3 任务中某些较为复杂的 SQL 语句, 查看其执行之前系统给出的分析计划和实际的执行计划, 记录观察的结果, 并对其进行简单的分析。

#### 3.1.5 DBMS 函数及存储过程和事务（选做）

- 1) 通过系统帮助文档学习系统关于时间、日期、字符串类型的函数, 为电影表增加首映时间属性, 然后查询下个月首映的电影信息。
- 2) 编写一个依据演员编号计算在其指定年份参演的电影数量的自定义的函数, 并利用其查询 2017 年至少参演过 5 部电影的演员编号。
- 3) 尝试编写 DBMS 的存储过程, 建立每家影院的上映电影总数的统计表, 并通过存储过程更新该表。
- 4) 尝试在 DBMS 的交互式界面中验证事务机制的执行效果。

## 3.2 完成过程

### 3.2.1 创建表格

1) 创建表格

参演表:

```
CREATE TABLE `actin` (
  `ACTID` INT(11) NOT NULL,
  `FID` INT(11) NOT NULL,
  `ISLEADING` VARCHAR(1),
  `GRADE` INT(11),
  PRIMARY KEY (`ACTID`,`FID`),
  KEY `FID` (`FID`),
```

```

        CONSTRAINT `actin_ibfk_1` FOREIGN KEY (`FID`) REFERENCES `film`
(`fid`),
        CONSTRAINT `actin_ibfk_2` FOREIGN KEY (`ACTID`) REFERENCES
`actor` (`actid`));

```

演员表:

```

CREATE TABLE `actor` (
    `ACTID` INT(11) NOT NULL,
    `ANAME` VARCHAR(30),
    `SEX` VARCHAR(2),
    `BYEAR` INT(11),
    PRIMARY KEY (`ACTID`)
);

```

电影表:

```

CREATE TABLE `film` (
    `FID` INT(11) NOT NULL,
    `FNAME` VARCHAR(255),
    `FTYPE` VARCHAR(255),
    `DNAME` VARCHAR(255),
    `FLENGTH` INT(11),
    `IS3D` VARCHAR(255),
    `GRADE` INT(11),
    PRIMARY KEY (`FID`),
    CHECK((`GRADE`>0 AND `GRADE`<100) OR GRADE IS NULL),
    CHECK(`IS3D` = 'Y' OR `IS3D` = 'N')
);

```

上映表:

```

CREATE TABLE `showt` (
    `FID` INT(11) NOT NULL,
    `TID` INT(11) NOT NULL,
    `PRICEC` INT(11),
    `SYEAR` INT(11),
    `SMONTH` INT(11),
    PRIMARY KEY (`FID`,`TID`),
    KEY `TID` (`TID`),
    CONSTRAINT `showt_ibfk_1` FOREIGN KEY (`TID`) REFERENCES
`theater` (`tid`),
    CONSTRAINT `showt_ibfk_2` FOREIGN KEY (`FID`) REFERENCES

```



```
`film`(`fid`)  
);
```

电影院表:

```
CREATE TABLE `theater` (  
  `TID` INT(11) NOT NULL,  
  `TNAME` VARCHAR(20),  
  `TAREA` VARCHAR(20),  
  `ADDRESS` VARCHAR(30),  
  PRIMARY KEY (`TID`)  
);
```

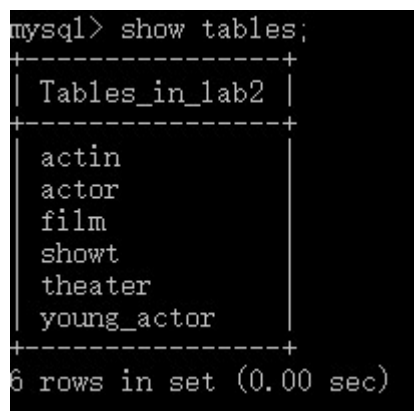


图 3.1 显示表格创建结果

引用列	引用数据库	引用表	引用列
`FID`	lab2	film	`fid`
`ACTID`	lab2	actor	`actid`
	lab2		

引用列	引用数据库	引用表	引用列
`TID`	lab2	theater	`tid`
`FID`	lab2	film	`fid`

图 3.2 定义外键结果

3.2.2 插入数据

```
INSERT INTO  
`film`(`FID`,`FNAME`,`FTYPE`,`DNAME`,`FLENGTH`,`IS3D`,`GRADE`)  
VALUES (1,'战狼','动作片','吴宇森',120,'Y',10),(2,'战狼 2','警匪片','导演  
1',12,'Y',91),(3,'战狼 3','枪战片 ','导演 1',50,'N',80),(4,'战狼 4','科幻片','吴宇森  
,79,'Y',NULL),(5,'星球大战','科幻','导演 2',NULL,NULL,89);
```

```
mysql> select * from film;
```

FID	FNAME	FTYPE	DNAME	FLENGTH	IS3D	GRADE
1	战狼	动作片	吴宇森	120	Y	10
2	战狼2	警匪片	导演1	12	Y	91
3	战狼3	枪战片	导演1	50	N	80
4	战狼4	科幻片	吴宇森	79	Y	NULL
5	星球大战	科幻	导演2	NULL	NULL	89

5 rows in set (0.02 sec)

图 3.3 电影表插入结果

INSERT INTO `actor`(`ACTID`,`ANAME`,`SEX`,`BYEAR`) VALUES (1,'男主角','男',1997),(2,'大魔王','男',1980),(3,'女主角','女',1996),(4,'路人甲','男',1983);

```
mysql> select * from actor;
```

ACTID	ANAME	SEX	BYEAR
1	男主角	男	1997
2	大魔王	男	1980
3	女主角	女	1996
4	路人甲	男	1983

4 rows in set (0.02 sec)

图 3.4 演员表插入结果

INSERT INTO `theater`(`TID`,`TNAME`,`TAREA`,`ADDRESS`) VALUES (1,'沁苑 415','洪山区','洪山区 1'),(2,'天河影院','江汉区','江汉区 1'),(3,'环球影院','武昌区','武昌区 1'),(4,'金逸影院','洪山区','洪山区 2');

```
mysql> select * from theater;
```

TID	TNAME	TAREA	ADDRESS
1	沁苑415	洪山区	洪山区1
2	天河影院	江汉区	江汉区1
3	环球影院	武昌区	武昌区1
4	金逸影院	洪山区	洪山区2

4 rows in set (0.01 sec)

图 3.5 电影院插入结果

INSERT INTO `showt`(`FID`,`TID`,`PRICEC`,`SYEAR`,`SMONTH`) VALUES (1,1,10,2017,1),(1,2,15,2017,11),(1,3,1,2018,11),(1,4,NULL,2017,2),(2,1,11,2017,10),(2,2,10,2017,11),(2,3,12,2016,12),(4,1,NULL,2018,NULL),(4,2,NULL,2018,NULL),(4,3,NULL,2018,NULL),(4,4,NULL,2000,1);

```
mysql> select * from showt;
```

FID	TID	PRICEC	SYEAR	SMONTH
1	1	10	2017	1
1	2	15	2017	11
1	3	1	2018	11
1	4	NULL	2017	2
2	1	11	2017	10
2	2	10	2017	11
2	3	12	2016	12
4	1	NULL	2018	NULL
4	2	NULL	2018	NULL
4	3	NULL	2018	NULL
4	4	NULL	2000	1

11 rows in set (0.01 sec)

图 3.6 上映表插入结果

```
INSERT INTO `actin`(`ACTID`,`FID`,`ISLEADING`,`GRADE`) VALUES
(1,1,'Y',99),(1,2,'Y',98),(2,1,'Y',10),(3,5,'Y',99);
```

```
mysql> select * from actin;
```

ACTID	FID	ISLEADING	GRADE
1	1	Y	99
1	2	Y	98
2	1	Y	10
3	5	Y	99

4 rows in set (0.01 sec)

图 3.7 参演表插入结果

### 3.2.3 数据更新

1)分别用一条 sql 语句完成对电影表基本的增、删、改的操作；  
增：

```
INSERT INTO film (FID,FNAME)
VALUES (8,'测试电影');
```

```
mysql> INSERT INTO film (FID,FNAME)
-> VALUES (8,'测试电影');
Query OK, 1 row affected (0.08 sec)

mysql> select * from film;
```

FID	FNAME	FTYPE	DNAME	FLENGTH	IS3D	GRADE
1	战狼	动作片	吴宇森	120	Y	10
2	战狼2	警匪片	导演1	12	Y	91
3	战狼3	枪战片	导演1	50	N	80
4	战狼4	科幻片	吴宇森	79	Y	NULL
5	星球大战	科幻	导演2	NULL	NULL	89
8	测试电影	NULL	NULL	NULL	NULL	NULL

```
6 rows in set (0.00 sec)
```

图 3.8 sql 插入

改:

UPDATE film

SET FNAME='修改电影名字'

WHERE FID=8;

```
mysql> UPDATE film
-> SET FNAME='修改电影名字'
-> WHERE FID=8;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from film;
```

FID	FNAME	FTYPE	DNAME	FLENGTH	IS3D	GRADE
1	战狼	动作片	吴宇森	120	Y	10
2	战狼2	警匪片	导演1	12	Y	91
3	战狼3	枪战片	导演1	50	N	80
4	战狼4	科幻片	吴宇森	79	Y	NULL
5	星球大战	科幻	导演2	NULL	NULL	89
8	修改电影名字	NULL	NULL	NULL	NULL	NULL

```
6 rows in set (0.00 sec)
```

图 3.9 sql 修改

删:

DELETE FROM film

WHERE FID=8;

```
mysql> DELETE FROM film
-> WHERE FID=8;
Query OK, 1 row affected (0.10 sec)

mysql> select * from film;
```

FID	FNAME	FTYPE	DNAME	FLENGTH	IS3D	GRADE
1	战狼	动作片	吴宇森	120	Y	10
2	战狼2	警匪片	导演1	12	Y	91
3	战狼3	枪战片	导演1	50	N	80
4	战狼4	科幻片	吴宇森	79	Y	NULL
5	星球大战	科幻	导演2	NULL	NULL	89

```
5 rows in set (0.00 sec)
```

图 3.10 sql 删除

2)将演员表中的 90 后演员记录插入到一个新表 YOUNG\_ACTOR 中)

```
CREATE TABLE YOUNG_ACTOR
SELECT * FROM actor
WHERE BYEAR > 1989 AND BYEAR < 2000
```

```
mysql> CREATE TABLE YOUNG_ACTOR
-> SELECT * FROM actor
-> WHERE BYEAR > 1989 AND BYEAR < 2000
-> ;
Query OK, 2 rows affected (0.07 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> select * from young_actor;
```

ACTID	ANAME	SEX	BYEAR
1	男主角	男	1997
3	女主角	女	1996

```
2 rows in set (0.00 sec)
```

图 3.11 90 后演员记录插入

3)数据导入导出:通过查阅 DBMS 资料学习数据导入导出功能，并将任务 2.1 所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

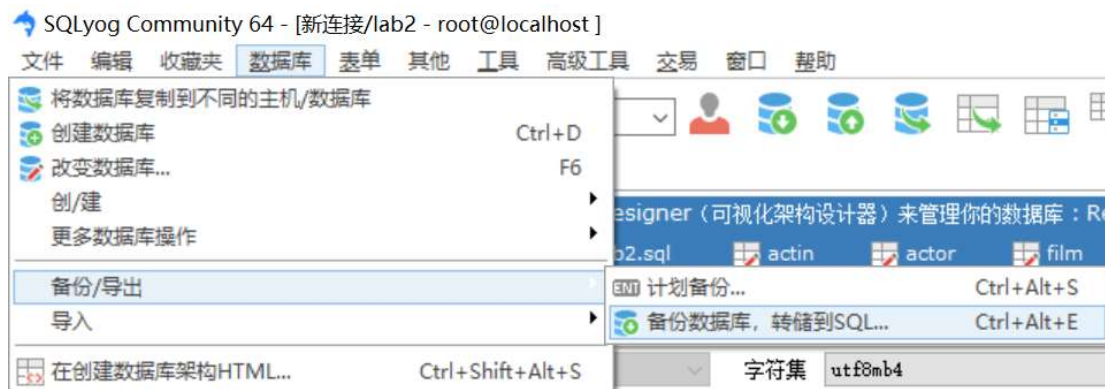


图 3.12 SQLyog 界面的导入/导出

```
.\mysqldump =uroot -p lab2 film > f:\film.b
```

```
DROP TABLE IF EXISTS `film`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
  SET character_set_client = utf8mb4 ;
CREATE TABLE `film` (
  `FID` int(11) NOT NULL
  `FNAME` varchar(255) DEFAULT NULL,
  `FTYPE` varchar(255) DEFAULT NULL,
  `DNAME` varchar(255) DEFAULT NULL,
  `FLENGTH` int(11) DEFAULT NULL,
  `IS3D` varchar(255) DEFAULT NULL,
  `GRADE` int(11) DEFAULT NULL,
  PRIMARY KEY (`FID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

图 3.13 导出数据

.\mysql -uroot -p lab2 < f:\film.b 则能导入数据到现有的数据库中。

4)观察性实验：建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

a	b
1	1
1	1
1	1
1	1

图 3.14 重复元素插入结果

```
UPDATE test SET a=2 WHERE a=1;
```

a	b
2	1
2	1
2	1
2	1

图 3.15 重复元组修改

DELETE FROM test WHERE a=2;

a	b
(NULL)	(NULL)

图 3.16 重复元组删除

5)创建视图：创建一个有 80 后演员作主角的参演记录视图，其中的属性包括：演员编号、演员姓名、出生年份、作为主角参演的电影数量、这些电影的用户评分的最高分。

```
CREATE VIEW view_test AS
SELECT actor.ACTID, actor.ANAME, actor.BYEAR,
COUNT(*),MAX(film.GRADE)
FROM actor, actin, film
WHERE actor.ACTID = actin.ACTID AND film.FID = actin.FID AND
actin.ISLEADING = 'Y' AND actor.BYEAR>1979 AND actor.BYEAR<1990
GROUP BY actor.ACTID
```

```
mysql> CREATE VIEW view_test AS
-> SELECT actor.ACTID, actor.ANAME, actor.BYEAR, COUNT(*),MAX(film.GRADE)
-> FROM actor, actin, film
-> WHERE actor.ACTID = actin.ACTID AND film.FID = actin.FID AND actin.ISLEADIN
G = 'Y' AND actor.BYEAR>1979 AND actor.BYEAR<1990
-> GROUP BY actor.ACTID;
Query OK, 0 rows affected (0.10 sec)

mysql> select * from view_test;
+-----+-----+-----+-----+-----+
| ACTID | ANAME | BYEAR | COUNT(*) | MAX(film.GRADE) |
+-----+-----+-----+-----+-----+
| 2 | 大魔王 | 1980 | 1 | 10 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

图 3.17 视图创建结果

6)触发器实验：编写一个触发器，用于实现对电影表的完整性控制规则：当增加一部电影时，若导演的姓名为周星驰，则电影类型自动设置为“喜剧”。

DELIMITER \$\$

```
CREATE
TRIGGER tri1 BEFORE INSERT
```

```

ON film
FOR EACH ROW BEGIN
IF(new.DNAME = '周星驰') THEN
SET new.FTYPE = '喜剧片';
END IF;
END$$

```

DELIMITER ;

```

mysql> DELIMITER $$
mysql>
mysql> CREATE
-> TRIGGER tri1 BEFORE INSERT
-> ON film
-> FOR EACH ROW BEGIN
-> IF(new.DNAME = '周星驰') THEN
-> SET new.FTYPE = '喜剧片';
-> END IF;
-> END$$
Query OK, 0 rows affected (0.09 sec)

mysql>
mysql> DELIMITER ;
mysql> insert into `FILM`(`FID`,`DNAME`) VALUES(13,'周星驰');
Query OK, 1 row affected (0.01 sec)

mysql> select * from film;
+----+-----+-----+-----+-----+-----+-----+
| FID | FNAME | FTYPE | DNAME | FLENGTH | IS3D | GRADE |
+----+-----+-----+-----+-----+-----+-----+
| 1   | 战狼  | 动作片 | 吴宇森 | 120     | Y    | 10    |
| 2   | 战狼2 | 警匪片 | 导演1  | 12      | Y    | 91    |
| 3   | 战狼3 | 枪战片 | 导演1  | 50      | N    | 80    |
| 4   | 战狼4 | 科幻片 | 吴宇森 | 79      | Y    | NULL  |
| 5   | 星球大战 | 科幻片 | 导演2  | NULL    | NULL | 89    |
| 13  | NULL  | 喜剧片 | 周星驰 | NULL    | NULL | NULL  |
+----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

图 3.18 触发器结果

## 2.2.4 数据查询

1)查询“战狼”这部电影在洪山区各家影院的 2017 年的上映情况，并按照上映的月份的降序排列；

```
SELECT showt.SYEAR, showt.SMONTH,theater.TNAME
```

```
FROM (film INNER JOIN showt ON film.FID=showt.FID) INNER JOIN theater ON
showt.TID=theater.TID
```



WHERE film.FNAME = '战狼' AND theater.TAREA = '洪山区' AND showt.SYEAR = 2017

ORDER BY showt.SMONTH;

<input type="checkbox"/>	YEAR	SMONTH	TNAME
<input type="checkbox"/>	2017	1	沁苑415
<input type="checkbox"/>	2017	2	金逸影院

图 3.19 查询 1 结果

2)查询所有无参演演员信息的电影的基本信息，并且将结果按照电影类型的升序排列，相同类型的电影则按照用户评分的降序排列；

```
SELECT * FROM film
WHERE FID NOT IN (
SELECT film.FID
FROM film ,actin
WHERE film.FID = actin.FID)
ORDER BY FTYPE ASC, GRADE DESC
```

<input type="checkbox"/>	FID	FNAME	FTYPE	DNAME	FLNGTH	IS3D	GRADE
<input type="checkbox"/>	13	(NULL)	喜剧片	周星驰	(NULL)	(NULL)	(NULL)
<input type="checkbox"/>	3	战狼3	枪战片	导演1	50	N	80
<input type="checkbox"/>	4	战狼4	科幻片	吴宇森	79	Y	(NULL)

图 3.20 查询 2 结果

3)查询所有直到 2017 年仍未上映的电影编号、电影名称、导演姓名；

```
SELECT FID,FNAME,DNAME FROM film
WHERE FID NOT IN (
SELECT showt.FID
FROM showt,film
WHERE showt.FID = film.FID AND showt.SYEAR <= 2017)
```

<input type="checkbox"/>	FID	FNAME	DNAME
<input type="checkbox"/>	3	战狼3	导演1
<input type="checkbox"/>	4	战狼4	吴宇森
<input type="checkbox"/>	5	星球大战	导演2
<input type="checkbox"/>	13	(NULL)	周星驰

图 3.21 查询 3 结果

4)查询在各家电影院均上映过的电影编号；

```
SELECT FID FROM(
```

```
SELECT FID , COUNT(TID) AS b FROM showt
GROUP BY FID) AS a
WHERE b IN (SELECT COUNT(*) FROM theater);
```

<input type="checkbox"/>	FID
<input type="checkbox"/>	1
<input type="checkbox"/>	4

图 3.22 查询 4 结果

5)查询所有用户评分低于 80 分或者高于 89 分的电影编号、电影名称、导演姓名及其用户评分，要求 where 子句中只能有一个条件表达式；

```
SELECT FID, FNAME, DNAME, GRADE
FROM FILM
WHERE GRADE NOT BETWEEN 80 AND 89
```

<input type="checkbox"/>	FID	FNAME	DNAME	GRADE
<input type="checkbox"/>	2	战狼2	导演1	91
<input type="checkbox"/>	1	战狼	吴宇森	10

图 3.23 查询 5 结果

6)查询每个导演所执导的全部影片的最低和最高用户评分；

```
SELECT DNAME, MAX(GRADE),MIN(GRADE)
FROM film
GROUP BY DNAME;
```

<input type="checkbox"/>	DNAME	MAX (GRADE)	MIN (GRADE)
<input type="checkbox"/>	吴宇森	10	10
<input type="checkbox"/>	导演1	91	80
<input type="checkbox"/>	导演2	89	89
<input type="checkbox"/>	周星驰	(NULL)	(NULL)

图 3.24 查询 6 结果

7)查询至少执导过 2 部电影的导演姓名、执导电影数量；

```
SELECT DNAME,COUNT(DNAME)
FROM film
GROUP BY DNAME HAVING COUNT(DNAME) >= 2;
```

<input type="checkbox"/>	DNAME	COUNT (DNAME)
<input type="checkbox"/>	吴宇森	2
<input type="checkbox"/>	导演1	2

图 3.25 查询 7 结果

8)查询至少 2 部电影的用户评分超过 80 分的导演及其执导过的影片数量、平均用户评分；

```
SELECT DNAME,COUNT(DNAME),AVG(GRADE)
FROM film
WHERE DNAME IN (
SELECT DNAME FROM (SELECT DNAME FROM film WHERE GRADE >= 80)
AS a
GROUP BY DNAME HAVING COUNT(DNAME) >= 2
) GROUP BY DNAME;;
```

<input type="checkbox"/>	DNAME	COUNT (DNAME)	AVG (GRADE)
<input type="checkbox"/>	导演1	2	85.5000

图 3.26 查询 8 结果

9)查询至少执导过 2 部电影的导演姓名以及跟这些导演合作过的演员编号、姓名；

```
SELECT film.DNAME, actor.ACTID, actor.ANAME
FROM actin, actor, film
WHERE actin.ACTID = actor.ACTID AND actin.FID = film.FID AND
film.DNAME IN(
SELECT DNAME
FROM film
GROUP BY DNAME HAVING COUNT(DNAME) >= 2);
```

<input type="checkbox"/>	DNAME	ACTID	ANAME
<input type="checkbox"/>	吴宇森	1	男主角
<input type="checkbox"/>	吴宇森	2	大魔王
<input type="checkbox"/>	导演1	1	男主角

图 3.27 查询 9 结果

10)查询每个演员担任主角的电影中的平均用户评分；

用户评分：用户对于演员的评分

```
SELECT ACTID, FID, AVG(actin.GRADE)
FROM actin
WHERE ISLEADING = 'Y'
GROUP BY ACTID;
```

<input type="checkbox"/>	ACTID	FID	AVG(actin.GRADE)
<input type="checkbox"/>	1	1	98.5000
<input type="checkbox"/>	2	1	10.0000
<input type="checkbox"/>	3	5	99.0000

图 3.28 查询 10 结果

11)查询用户评分超过 90 分的电影的最早上映年月；

```
SELECT FNAME, MINYEAR, MIN(SMONTH)
FROM (SELECT FILM.FID, FNAME, MIN(SHOWT.SYEAR) AS MINYEAR
FROM FILM, SHOWT
WHERE FILM.FID=SHOWT.FID AND GRADE > 80
GROUP BY FNAME) AS a, showt
WHERE a.FID = SHOWT.FID
GROUP BY SHOWT.FID
```

<input type="checkbox"/>	FNAME	SYEAR	SMONTH
<input type="checkbox"/>	战狼2	2016	12

图 3.29 查询 11 结果

12)查询用户评分超过 90 分的电影的最早上映年月及其相应的上映影院编号；

```
SELECT FNAME, MINYEAR, MIN(SMONTH), SHOWT.TID
FROM (SELECT FILM.FID, FNAME, MIN(SHOWT.SYEAR) AS MINYEAR
FROM FILM, SHOWT
WHERE FILM.FID=SHOWT.FID AND GRADE > 80
GROUP BY FNAME) AS a, showt
WHERE a.FID = SHOWT.FID
GROUP BY SHOWT.FID
```

<input type="checkbox"/>	FNAME	TID	SYEAR	SMONTH
<input type="checkbox"/>	战狼2	3	2016	12

图 3.30 查询 12 结果

13)查询每个电影的上映总次数；

```
SELECT FID,COUNT(FID)
FROM showt
GROUP BY FID;
```

	FID	COUNT(FID)
<input type="checkbox"/>	1	4
<input type="checkbox"/>	2	3
<input type="checkbox"/>	4	4

图 3.31 查询 13 结果

14)查询执导过动作片，或者警匪片，或者枪战片的导演的姓名，要求 where 子句中只能有一个条件表达式；

```
SELECT DISTINCT DNAME
FROM film
WHERE FTYPE IN ('动作片','警匪片','枪战片');
```

DNAME
吴宇森
导演1

图 3.32 查询 14 结果

15)查询所有“战狼”系列的电影的编号、电影名称、上映电影院名称及其上映年月，结果按照电影名称的升序排列；

```
SELECT film.FID, film.FNAME, theater.TNAME, showt.SYEAR, showt.SMONTH
FROM film,showt,theater
WHERE film.FID = showt.FID AND showt.TID = theater.TID AND LOCATE('战狼',film.FNAME) <> 0
ORDER BY film.FNAME ASC;
```

FID	FNAME	TNAME	SYEAR	SMONTH
1	战狼	沁苑415	2017	1
1	战狼	天河影院	2017	11
1	战狼	环球影院	2018	11
1	战狼	金逸影院	2017	2
2	战狼2	沁苑415	2017	10
2	战狼2	天河影院	2017	11
2	战狼2	环球影院	2016	12
4	战狼4	沁苑415	2018	(NULL)
4	战狼4	天河影院	2018	(NULL)
4	战狼4	环球影院	2018	(NULL)
4	战狼4	金逸影院	2000	1

图 3.33 查询 15 结果

16)查询在同一个年月上映 1 号和 2 号电影的影院编号；

```
SELECT a.TID
FROM (SELECT TID,SYEAR,SMONTH FROM showt
WHERE FID = 1) AS a,(SELECT TID,SYEAR,SMONTH FROM showt
WHERE FID = 2) AS b
WHERE a.SYEAR = b.SYEAR AND a.SMONTH = b.SMONTH;
```

TID
2

图 3.34 查询 10 结果

17)查询所有没参演过用户评分 85 分以下电影的演员的编号、姓名；

```
SELECT actor.ACTID, actor.ANAME
FROM actin,actor
WHERE actin.ACTID = actor.ACTID AND actin.ACTID NOT IN (
SELECT ACTID
FROM actin,film
WHERE actin.FID = film.FID AND film.GRADE < 85);
```

ACTID	ANAME
3	女主角

图 3.35 查询 10 结果

18)查询参演过“吴宇森”执导过的所有电影的演员姓名；

```
SELECT actor.ANAME
FROM actor,actin,film
WHERE actor.ACTID = actin.ACTID AND actin.FID = film.FID AND
film.DNAME = '吴宇森'
GROUP BY(actor.ANAME)
```

ANAME
男主角
大魔王

图 3.36 查询 10 结果

19)查询所有的演员的编号、姓名及其参演过的电影名称，要求即使该演员未参演过任何电影也要能够输出其编号、姓名；

```
SELECT actor.ACTID,actor.ANAME,film.FNAME
FROM actor
LEFT JOIN actin
ON actor.ACTID = actin.ACTID
LEFT JOIN film
ON film.FID = actin.FID;
```

ACTID	ANAME	FNAME
1	男主角	战狼
1	男主角	战狼2
2	大魔王	战狼
3	女主角	星球大战
4	路人甲	(NULL)

图 3.37 查询 19 结果

20)查询所有上映超过 3 次但没有用户评分的电影编号、名称。

```
SELECT FID,FNAME
FROM film
WHERE GRADE IS NULL AND FID IN (
SELECT FID
FROM showt
GROUP BY FID HAVING COUNT(FID)>3);
```

FID	FNAME
4	战狼4

图 3.38 查询 20 结果

### 3.2.5 了解系统的查询性能分析功能（选做）

目的: 选择上述 3.2.4 任务中某些较为复杂的 SQL 语句，查看其执行之前系统给出的分析计划和实际的执行计划，记录观察的结果，并对其进行简单的分析。

使用 explain 查询第 20 题中的 SQL 语句执行:

```
EXPLAIN SELECT FID,FNAME
FROM film
WHERE GRADE IS NULL AND FID IN (
SELECT FID
```

FROM showt  
GROUP BY FID HAVING COUNT(FID)>3);

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	film	(NULL)	OK ALL	(NULL)	(NULL)	(NULL)	(NULL)	6	16.67	Using where
2	SUBQUERY	showt	(NULL)	OK index	PRIMARY,TID	PRIMARY	8	(NULL)	11	100.00	Using index

图 3.39 分析语句执行结果

table :表明, type :连接类型, possible\_keys :可能被使用的索引, key :正在被使用的索引, key\_len :索引, ref :索引的被使用列, rows :返回请求数据的行, Extra :备注信息。

### 3.2.6 DBMS 函数及存储过程和事务（选做）

1) 通过系统帮助文档学习系统关于时间、日期、字符串类型的函数, 为电影表增加首映时间属性, 然后查询下个月首映的电影信息。

```
ALTER TABLE film ADD premiere DATE;
UPDATE FILM SET premiere=DATE_FORMAT("2018-7-7","%y-%m-%d")
WHERE FID=1;
SELECT * FROM FILM WHERE MONTH(premiere)-MONTH(NOW())=1;
```

FID	FNAME	FTYPE	DNAME	FLLENGTH	IS3D	GRADE	premiere
1	战狼	动作片	吴宇森	120	Y	10	2018-07-07

图 3.40 下月首映电影

## 3.3 任务总结

1. 创建触发器时, 需要重定义分隔符, 防止数据库认为 SQL 语句已经输入完毕。
2. MySQL 中的约束没有强制起作用的效力, 在插入或更新入约束外的非法数据时仍会正常操作, 需要通过定义一个触发器的方法来起到相同的效用。
3. SQL 中的自带数据类型丰富, 也封装了许多的复杂内部实现函数。
4. 本次实验中主要是增强了自己对于 SQL 脚本语法的熟练程度, 虽然不同数据库对于 SQL 的高级特性有不同的支持程度, 但是在 CURD 操作上还是大致相同的。
5. 获得相同结果的 SQL 查询脚本的查询过程和性能会有很大的差别, 在较大的数据基数上, 这样的时间空间差距会进一步被扩大。



## 4 数据库应用系统设计

### 4.1 系统设计目标

自行选择所擅长的 DBMS 软件以及数据库应用系统（客户端程序或者网站）的程序开发工具，参考后面的题目例子，拟定一个自己感兴趣的数据库应用系统题目，完成该小型数据库应用系统的设计与实现工作。主要内容包括：需求调研与分析、总体设计、数据库设计、详细设计与实现、测试等环节的工作。

选择题目：汽车租赁信息系统

### 4.2 需求分析

#### 4.2.1 任务文档要求

采用 C/S 模式实现一个汽车租赁信息系统。完成用户、车辆、经手员工、租借情况、车辆损毁情况、交通违规罚款等信息的管理。实现以下功能：

- 1) 实现不同权限的浏览和更新。
- 2) 能够根据车辆使用情况计算押金退还金额。
- 3) 能查询客户的租借历史记录，并进行信誉度评价，进行会员制和非会员制的客户管理。
- 4) 能够管理车辆报修信息；
- 5) 能够生成租借公司的日、月、季度、年财务报表。
- 6) 能够在客户端中对数据库的数据进行备份。

#### 4.2.2 需求分析内容

概述：租车信息管理系统是一个面向用户和汽车租赁公司双方的一个跨平台的信息查询管理系统。

功能需求：

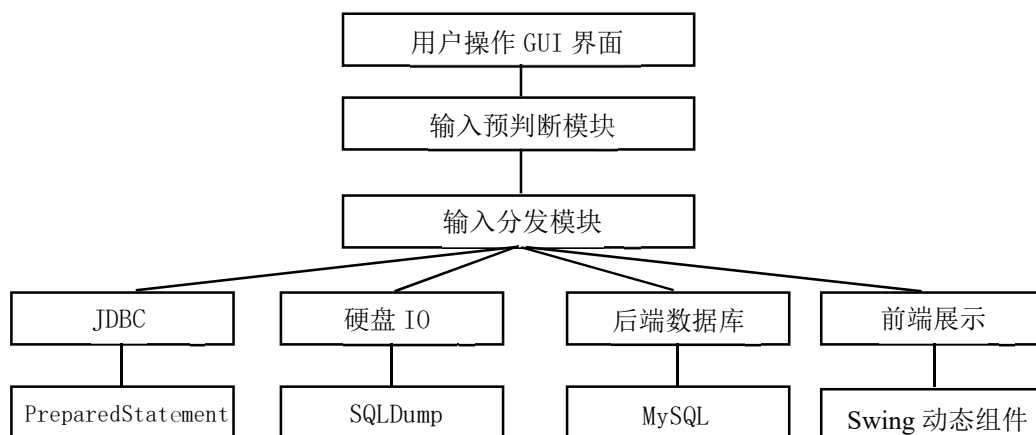
1. 作为用户方，能够通过管理系统查询自己的租借记录和交通违规，公司目前的车辆等信息。
2. 作为公司方，能够分级进行修改、更新管理系统中的包括用户、车辆、租借记录、管理员在内的各种信息。
3. 不同级别的信息管理系统账户有不同的修改和浏览权限（在 GUI 界面上实现阻止访问）。
4. 信息管理系统具有一定的统计功能，能为公司进行财报等信息的基础整合统计，并可以导出统计图表文件。
5. 使用者无需在 SQL 级别上进行操作，均可通过信息系统的 GUI 界面满足需求。

6. 信息管理系统通过预编译和词法分析的方式防止 SQL 注入攻击，保证基础的安全性。
7. 管理员可以对数据库的数据进行备份。

性能需求：多用户在不同的平台上操作时，面对并发的数据吞吐情况也需要能够保证操作流畅性。

## 4.3 总体设计

### 4.3.1 系统体系结构



### 4.3.2 总体功能模块划分

系统的各项功能模块如表 4.1 所示。

功能编号	功能名称	功能描述	解决需求
1	操作界面	基于 Java Swing 的 GUI 库为用户提供其权限范围内的各项控制	多权限用户的访问控制需求
2	输入预判断	判断用户在 GUI 界面中的操作的合理性。	防止异常操作，异常值输入，维护信息系统内部信息的正常，预防 SQL 注入攻击的安全性需求
3	输入分发	将输入的数据预处理后分发到各个处理模块中进行进一步处理。	衔接层
4	JDBC	使 Java 客户端程序能与后端的 MySQL 链接进行操作。	满足前后端链接的驱动需求。
5	硬盘 IO	导入导出数据库结构与数据。	数据的迁移以及备份需求
6	后端数据库	使用 MySQL 进行对数据的管理。	保证了数据存储的性能高效，满足性能需求。

7	前端展示	使用动态的图表来展示数据结果和财报等统计信息。	满足数据管理统计等增值功能需求。
---	------	-------------------------	------------------

表 4.1 系统功能模块

## 4.4 数据库设计

图 4.1 展示了主要管理数据的 E-R 图模型。数据库中的主要数据由四个实体组成，车辆、员工、顾客、日志。每个实体都有各自的一系列属性和对应的主码（id）。其中最中心的实体为日志，每个日志都展示了汽车租赁公司的业务流水情况，因此和车辆、员工、顾客都有了连接关系。（信息管理系统的用户表由于信息的独立性，故不在 E-R 图中进行展示。）

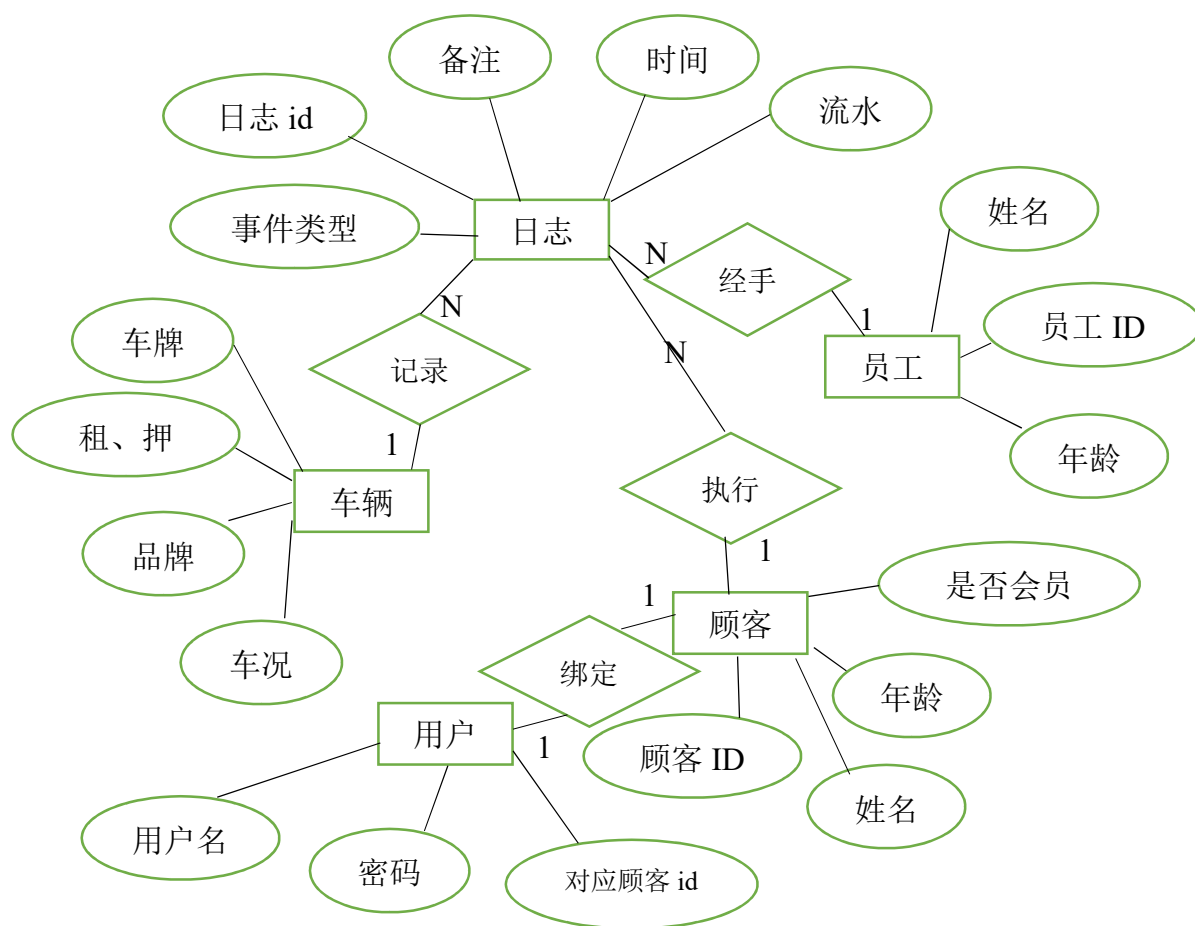


图 4.1 核心数据 E-R 图模型

表 4.2-4.6 为数据库中所存储的关系模型和具体数据类型。

属性名	作用	数据类型	长度	键关系	非空	备注
license	车牌	varchar	255	主键	√	

brand	品牌	varchar	11		√	
cost	租金	int	11		√	
status	车况	int	11		√	0-5 代表车况从坏到好
pledge	押金	int	11		√	

表 4.2 车辆信息表

属性名	作用	数据类型	长度	键关系	非空	备注
id	顾客 ID	int	11	主键	√	
name	顾客姓名	varchar	255		√	
age	顾客年龄	int	255			
member	是否会员	varchar	31		√	默认为 N

表 4.3 顾客信息表

属性名	作用	数据类型	长度	键关系	非空	备注
id	员工 ID	int	11	主键	√	
name	员工姓名	varchar	255		√	
age	员工年龄	int	11			

表 4.4 员工信息表

属性名	作用	数据类型	长度	键关系	非空	备注
name	用户账户	varchar	255	主键	√	
password	用户密码	varchar	255		√	
author	用户权限等级	int	11		√	权限从 1-3，分别对应超级管理员，管理员，普通用户，管理员不能修改用户表，普通用户只能浏览有关自己的信息。
customerid	绑定顾客 id	int	11	外键于顾客表 id		若为顾客账户，需要绑定一个顾客 id

表 4.5 用户信息表

属性名	作用	数据类型	长度	键关系	非空	备注
infoid	事件 ID	int	11	主键	√	
license	车牌	varchar	255	外键于车辆表 id	√	
customerid	顾客 ID	int	11	外键于顾客表 id		
stuffid	经手员工 ID	int	11	外键于员工表 id	√	

time	时间	int	11		√	格式 20180526
event	事件类型	int	11		√	类型 1-4，分别为损坏、罚款、借车、还车。使用触发器进行约束。
moychange	这次事件带来的流水	int	11			默认为 0
detailevent	事件备注	varchar	255			事件的详细情况

表 4.6 日志信息表

表 4.7 为不同权限的用户的可访问能力，通过权限的限制来保证用户的操作不越界。

	超级管理员 (级别: 1)	管理员 (级别: 2)	普通用户 (级别: 3)
浏览自身相关信息	√	√	√
浏览全局信息	√	√	
浏览管理系统用户	√		
修改自身信息	√	√	√
修改租车信息	√	√	
修改系统用户信息	√		

表 4.7 用户访问权限表

## 4.5 数据流图 and 业务流程图

数据流图如图 4.3 所示，展示了数据在不同功能模块子系统间的传递逻辑。

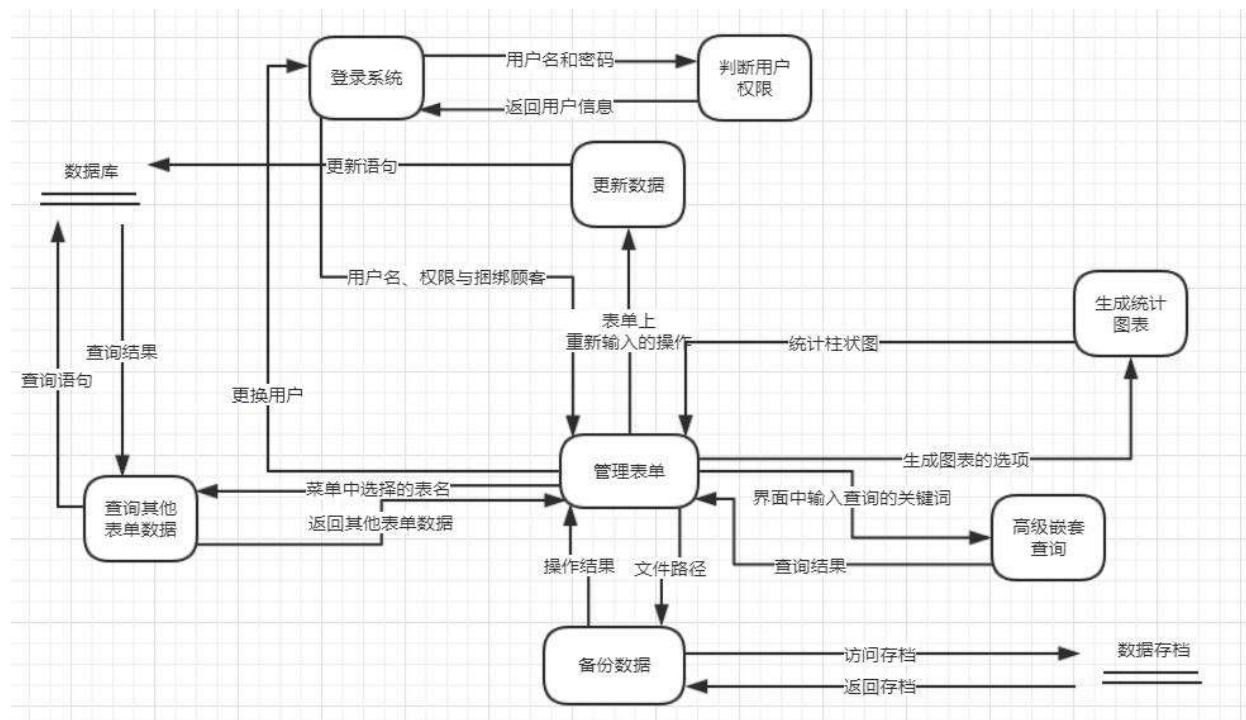


图 4.2 数据流图

用户在信息管理系统中进行的操作响应流程图如图 4.3 所示。

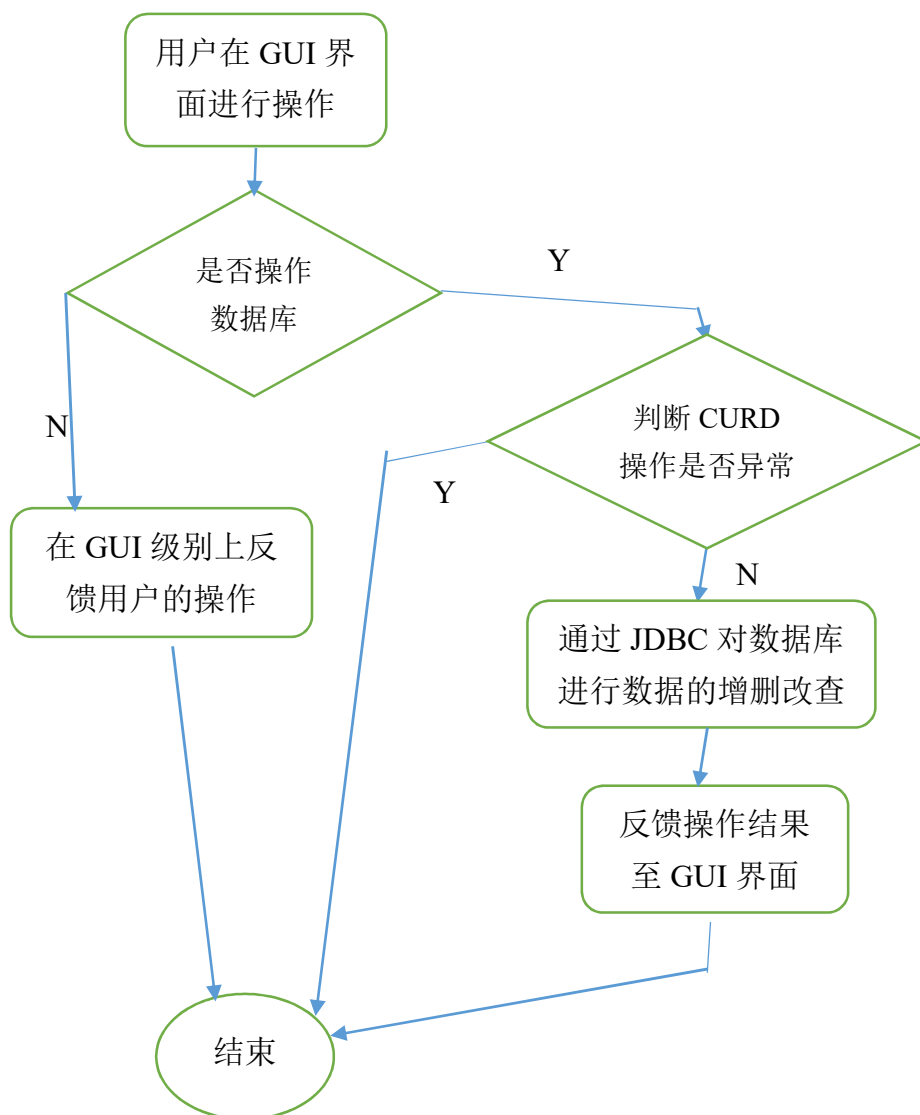


图 4.3 操作响应流程图

对操作系统执行 SQL 脚本的流程图如图 4.4 所示。

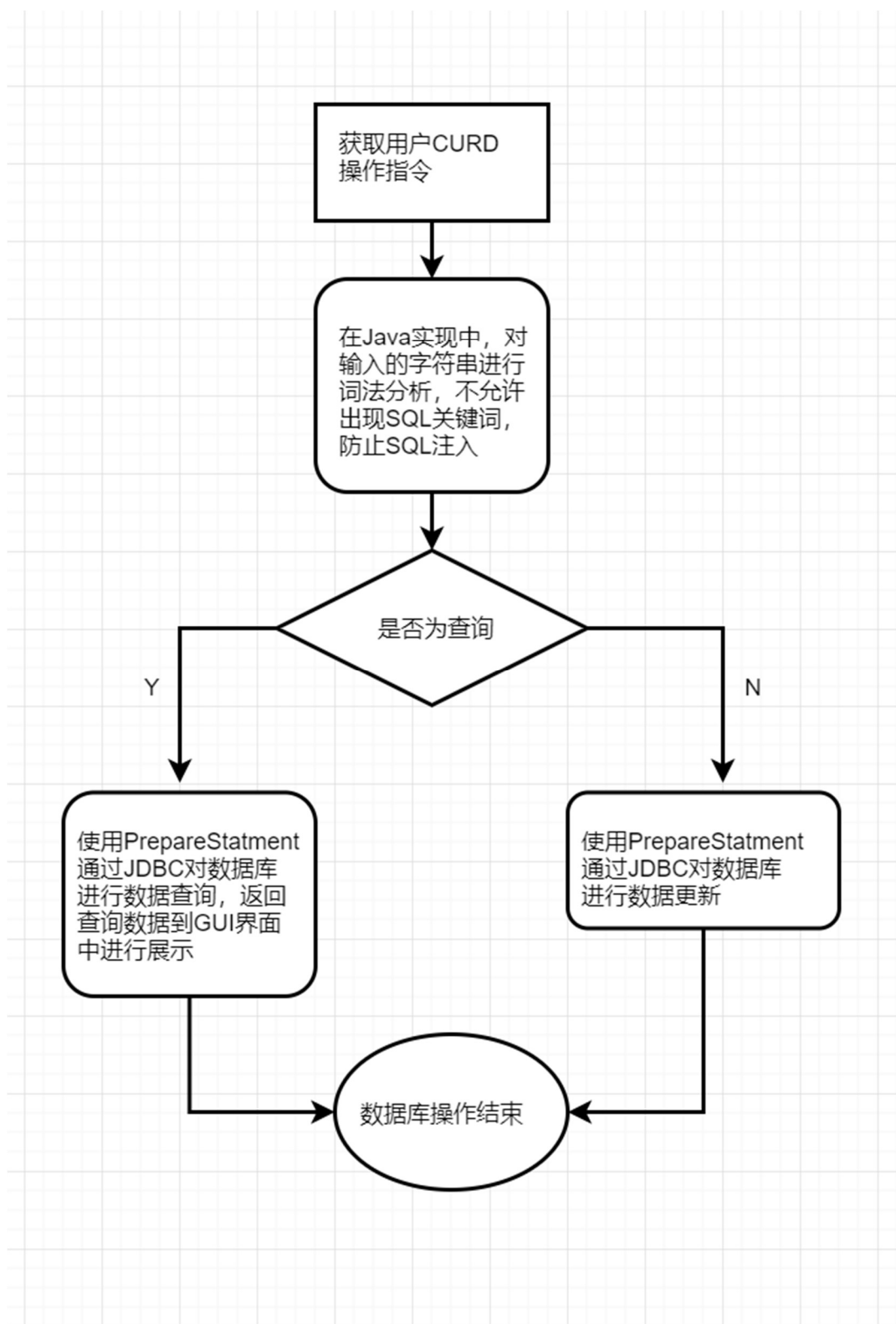


图 4.4 对数据库操作流程



## 4.6 详细设计与实现

### 4.6.1 运行环境

系统前端界面基于 Java 标准库中的 Swing 组件进行构造，基于 BeautyEye 开源框架进行优化视图，基于 JFreeChart 进行统计报表展示，后端通过 JDBC 驱动链接 MySQL 数据库，通过 Java.Sql 标准库对数据库进行操作。由于 JVM 的跨平台特性，本程序能够运行在各支持 JVM 的操作系统上。

### 4.6.2 前端主要组件

1. 登录界面：通过 JDialog 上布局成对标签与文本框，在用户点击确定登录后，将输入的用户名和密码送入数据库查询模块中进行查询，正确则进入系统主界面。

对应核心代码：

```
private JFrame initLogin() {
    DataBase = DataBase.getInstance();//单例模式，获得一个数据库控制的
    实例

    frame = new JFrame();//初始化父容器
    if (DataBase.initConnect()) { //初始化数据库连接

        dialogLogin = new JDialog();
        try {
            dialogLogin.setContentPane(new
BackgrouPanel("res/loginBackground.jpg"));
        } catch (IOException e) {
            e.printStackTrace();
        }
        ...
        初始化 Dialog 布局等配置与其包括 TextFiled 在内的子组件
        ...
        return frame;
    } else {
        JOptionPane.showMessageDialog(frame, "数据库连接失败");
        return null;
    }
}
```

2. 系统主界面：系统主界面主要提供表单的展示，以及其他功能的入口。

在主 JFrame 上通过 BorderLayout 分置上下两部分。

上部通过嵌套 JPanel 放置一系列的搜索组件（成对标签与输入框或 Combobox），当用户点击搜索按钮后，生成带有指定 WHERE 子句的 SELECT SQL 语句，获得数据后更新更新表单。

下部放置 JTable 组件，用来展示数据库数据，JTable 根据当前进入不同的表单展示不同的数据，根据不同的用户指定不同的修改与查看权限。用户可以对其有权限修改的表格直接进行修改，当用户尝试修改表单内数据后，检查输入数据是否合法，若合法则生成 SQL 语句对数据库进行更新操作。对于除事件表外的表单，都直接展示原始数据，而因为事件表链接了多个外表，数据的展示需要进行筛选，展示有可阅读性的数据。JTable 上指定一个右键菜单来进行删除行和添加新行，添加新行时，通过打开一个新的 JDialog 来让用户输入新行的数据，同样进行语法检查操作后，生成 SQL 语句来操作数据库。

核心实现函数组：

//初始化主界面的标题、菜单等各项配置。

public void initMainFrame()

//初始化右键菜单，实现删除行与添加行的进入位置（权限为 3 的普通用户无法打开）。

private void createPopupMenu()

//初始化一个添加行对话框，每个新元组数据进行合法性分析后，获得 DataBase 单实例进行添加数据。

private void setAddRowDialog()

//刷新表格 Panel，将传入的两个 Vector 参数作为数据源，根据当前权限，对是否可修改或查看进行权限管理，并实现右键菜单弹出渲染逻辑。

private void setTablePanel(Vector<Vector<String>> vectors, Vector<String> columns)

//根据不同的表，绘制不同的成对 Label 与 TextField 或 Combobox，内部通过获取搜索栏的新约束条件获得新的结果并重绘表格 Panel。传入参数 Panel 为整个 Search 逻辑的父容器。

private void setSearchPanel(JPanelOpen jPanelSearch)

//检查数据是否合法，传入两字符串参数为属性名与值。本函数主要用于检测用户定义约束以及关键词检测，防止 SQL 注入的功能。

```
public boolean checkDataLegal(String name, String value)
```

3. 统计图表：通过主界面的生成报表选项进入，在财务报表的界面中，用户通过两个 Combobox 来选择横轴（查询项目）与纵轴（时间单位），确认后通过 JFreeChart 绘制柱状图来展示统计数据。基于 JFreeChart 组件，用户可以缩放图表，展示指定区间，导出图表为图片或 PDF 文件或直接调用打印接口。在信誉度查询的界面中，信誉值计算的方法为系统通过记录用户相关的日志事件进行加权计值，然后使用 Sigmoid 函数非线性缩放到-10~10 的范围内。

对应核心代码：

```
private void showChart() {
    JDialog chartDialog = new JDialog(frame);
    JPanel contChartPane = new JPanel(new BorderLayout());
    chartDialog.setContentPane(contChartPane);

    JPanel argPanel = new JPanel(new FlowLayout());
    ...
    初始化并添加 JLabelOpen, JCombobox, JButton 等子组件以及对应的
    二级容器
    ...
    jConfirmBut.addActionListener(e -> {
        drawChart(((String) jComboxMode.getSelectedItem()),
            ((String) jComboxFunc.getSelectedItem()),
            chartPanel);//绘制逻辑
        chartDialog.pack();
    });
    ...
    初始化 chartDialog、chartDialog 添加关系与位置等配置
    ...
}
//绘制表格逻辑
private void drawChart(String mode, String func, JPanel panel) {
    panel.removeAll();
    String horizon;//纵轴标签
    if (func.equals("流水")) {
        horizon = "元";
    } else {
        horizon = "次数";
    }
}
```

```

    }
    JFreeChart chart = ChartFactory.createBarChart3D(func, mode,
horizon, getChartData(func, mode), PlotOrientation.VERTICAL, false, false, false);//
从 ChartFactory 工厂类中拿所需的 3D 柱状图实例
    ...
    初始化字体等配置
    ...

    panel.add(chartP);

}

```

4. 备份：通过主界面的备份选项进入，通过 JFileChooser 获得用户指定的路径，再通过 Java Runtime 调用外部程序 MySQLDump.exe 来对数据库的结构与数据备份到指定路径的 SQL 脚本文件中。

对应核心代码：

//弹出 JFileChooser 监听器

```

ActionListener fileListener = new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        JFileChooser jFileChooser = new JFileChooser();
        File file;
        String path = null; //指定文件路径
        if (command.equals("导入")) {
            jFileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
            jFileChooser.showOpenDialog(frame);
            file = jFileChooser.getSelectedFile();
            if (file.isFile()) {
                path = file.getAbsolutePath();
            }
        } else {
            jFileChooser.showSaveDialog(frame); //展示出
            file = jFileChooser.getSelectedFile();
            String typeInName = jFileChooser.getName(file);
            if (typeInName == null || typeInName.equals("")) return; //防止无
效空路径

            path = jFileChooser.getCurrentDirectory() + "\\" + typeInName;
        }
    }
}

```

```

        if (path != null) {
            try {
                String bat;
                if (command.equals("导出")) {
                    bat = "C:\\Program Files\\MySQL\\MySQL Server
8.0\\bin\\mysqldump.exe lab3 -uroot " + " -pXIANG1569348 -r" + path + " --skip-
lock-tables";
                }
                Process process = Runtime.getRuntime().exec(bat);//调用
Runtime 来执行 bat 命令
                int com = process.waitFor();//等待执行
                if (com == 0) {
                    noticeMsg("操作成功");
                } else {
                    noticeMsg("操作失败");
                }
            } catch (IOException e1) {
                e1.printStackTrace();
            } catch (InterruptedException e1) {
                e1.printStackTrace();
            }
        }
    }
};

```

#### 4.6.3 数据库操作

对于数据库的增删改查操作均基于 JDBC，Java.sql 包内标准库的 Statement 或 PreparedStatement 来完成对 SQL 语句的执行。

1. 获得数据：根据不同的表名选择 FROM 子句，不同的约束条件生成 WHERE 子句，最后生成一条完整的 SQL 语句，最后通过 Statement 执行该 SQL 语句后，返回多行数据，再回到界面上进行更新表单的操作。
2. 更新数据：根据不同的表名选择 FROM 子句，不同的列名和新值生成 SET 子句，不同的主键值来生成 WHERE 子句，最后通过 Statement 执行该 SQL 语句。
3. 删除行：根据不同的表名选择 FROM 子句，不同的主键值生成 WHERE 子句，最后通过 Statement 执行该 SQL 语句。

4. 添加行：根据不同的表名选择 FROM 子句，不同的新值列表和列名列表来生成 VALUES 子句，最后通过 Statement 执行该 SQL 语句。

5. 获得财务统计数据：报表根据不同的查询项目和时间单位来生成不同的 SELECT 语句子项，返回多行数据。JFreeChart 通过返回的数据来绘制图表。

6. 获得信誉值统计数据：在日志表中通过聚集函数来生成查询不同用户发生不同日志事件的总次数。

## 4.7 系统测试

测试环境：

操作系统：Windows 10 1803，数据库：MySQL 8.0，JRE：Java(TM) SE Runtime Environment (build 1.8.0\_161-b12)，JVM：Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)，数据库可视化工具：SQLyog

### 4.7.1 查询数据测试

通过菜单栏的选择指定车辆表，如图 4.5，主界面刷新正确展示出车辆表的搜索框和车辆表可浏览的全部信息。



图 4.5

如图 4.6 所示系统生成的对应查询语句，符合查询需求。

```
SELECT * FROM car
```

图 4.6

如图 4.7 所示，在搜索框中指定条件后点击搜索按钮后，系统展示出了符合

指定条件的车辆数据。

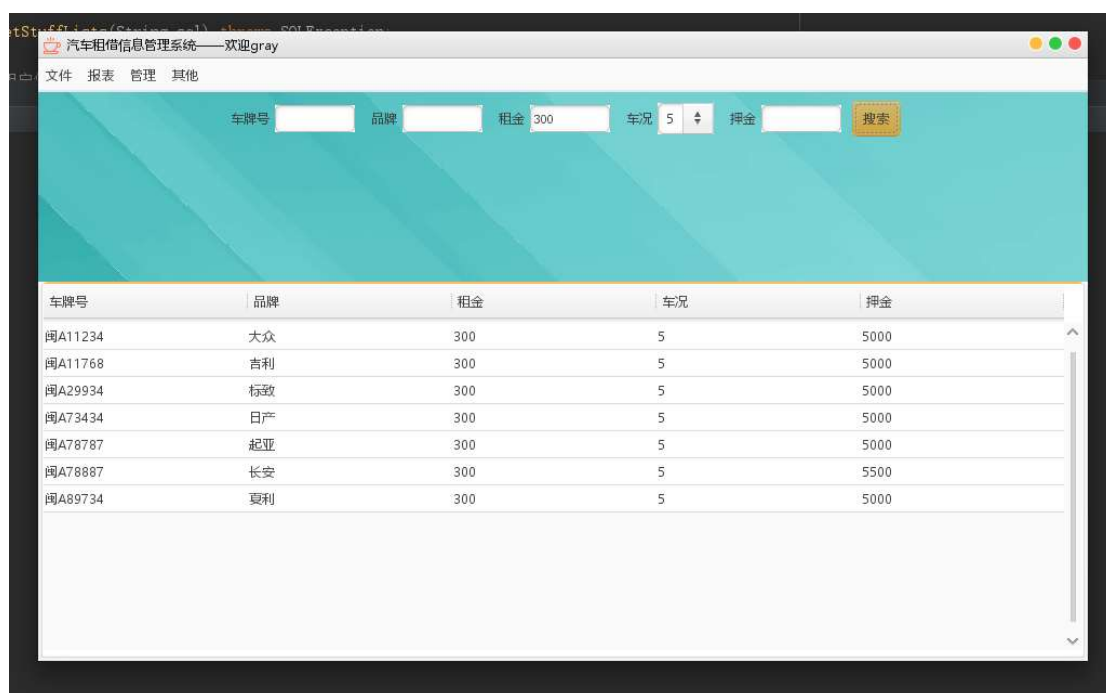


图 4.7

如图 4.8 所示，系统生成了符合搜索条件的 SQL 查询语句。

```
SELECT * FROM car WHERE cost = '300' and status = '5'
```

图 4.8

### 4.7.2 添加数据测试

如图 4.9 所示，用户输入新数据进行添加后，表单中出现新的正确数据。



图 4.9

如图 4.10 所示，系统执行了正确的添加行 SQL 语句。

```
INSERT INTO car (brand,license,pledge,cost,status) VALUES ('兰博基尼','鄂A98999','200000','2000','5')
```

图 4.10

### 4.7.3 修改数据测试

如图 4.11 所示，用户可以通过点击表单数据，直接进行修改。

闽A12344	大众	250	4	4000
闽A22227	马自达	300	4	5000
闽A29934	标致407	300	5	5000
闽A73434	日产	300	5	5000
闽A78787	起亚	300	5	5000

图 4.11

如图 4.12 所示，系统执行了正确的更新数据 SQL 语句。

```
com.mysql.cj.jdbc.ClientPreparedStatement: UPDATE car SET brand = '标致407' where license = '闽A29934'
```

图 4.12

#### 4.7.4 删除数据测试

如图 4.13 所示，用户通过右键菜单可以指定删除任意行。

闽A12344	大众	250	4	4000
闽A22227	马自达	300	4	5000
闽A29934	标致407	300	5	5000
闽A73434	日产	300	5	5000
闽A78787	起亚	300	5	5000
闽A78887	长安	300	5	5500

图 4.13

如图 4.14 所示，系统执行了正确的删除数据 SQL 语句

```
com.mysql.cj.jdbc.ClientPreparedStatement: DELETE FROM car WHERE license = '闽A29934'
```

图 4.14

#### 4.7.5 生成报表测试

如图 4.15 所示，用户可以选择纵轴和横轴后生成所需的图表，且可以通过右键菜单导出图表为 PDF 或 PNG 等格式。





图 4.15

如图 4.16、4.17 所示，用户可以截取指定纵轴范围进行展示。

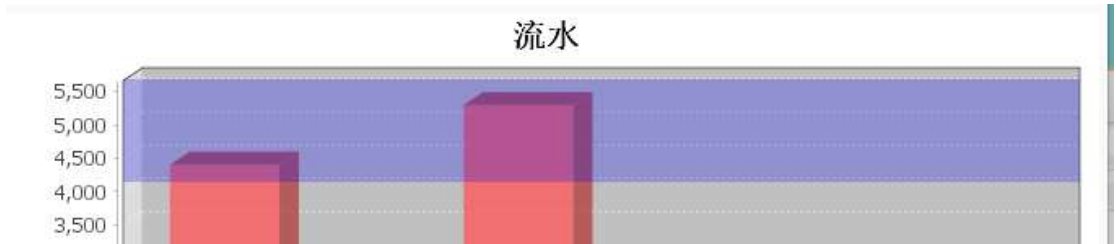


图 4.16



图 4.17

如图 4.18 所示，系统执行了正确的选取数据 SQL 语句。

```
SELECT TIME, SUM(moychange) FROM info GROUP BY TIME
```

图 4.18

#### 4.7.6 备份数据测试

如图 4.19 所示，用户可以通过菜单栏的备份导出入口将数据库的结构与数据转化为 SQL 脚本备份到外部。

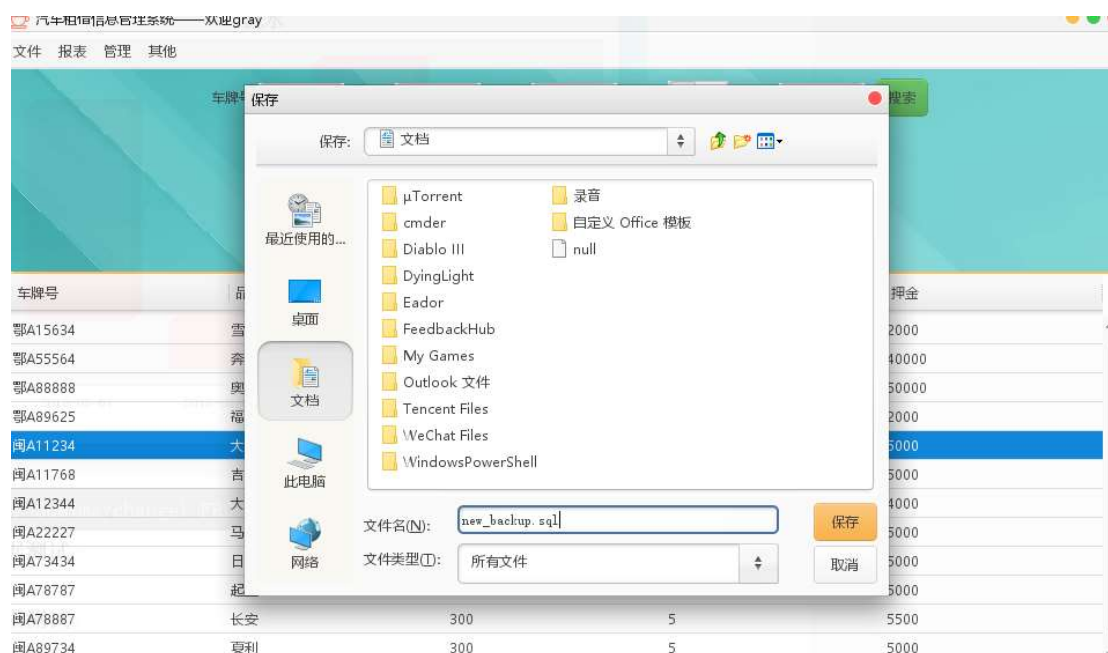


图 4.19

如图 4.20 所示，系统通过调用 mysqldump 生成了备份文件。

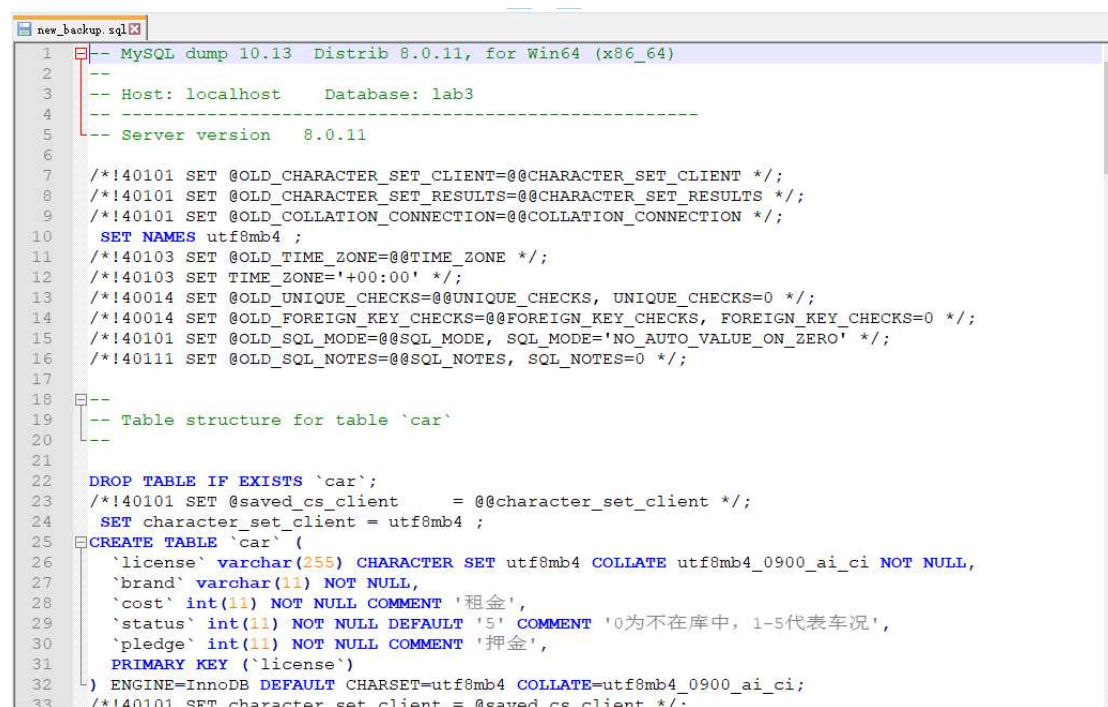


图 4.20

### 4.7.7 用户权限测试

如图 4.21 所示，系统通过维护用户表来管理用户权限。

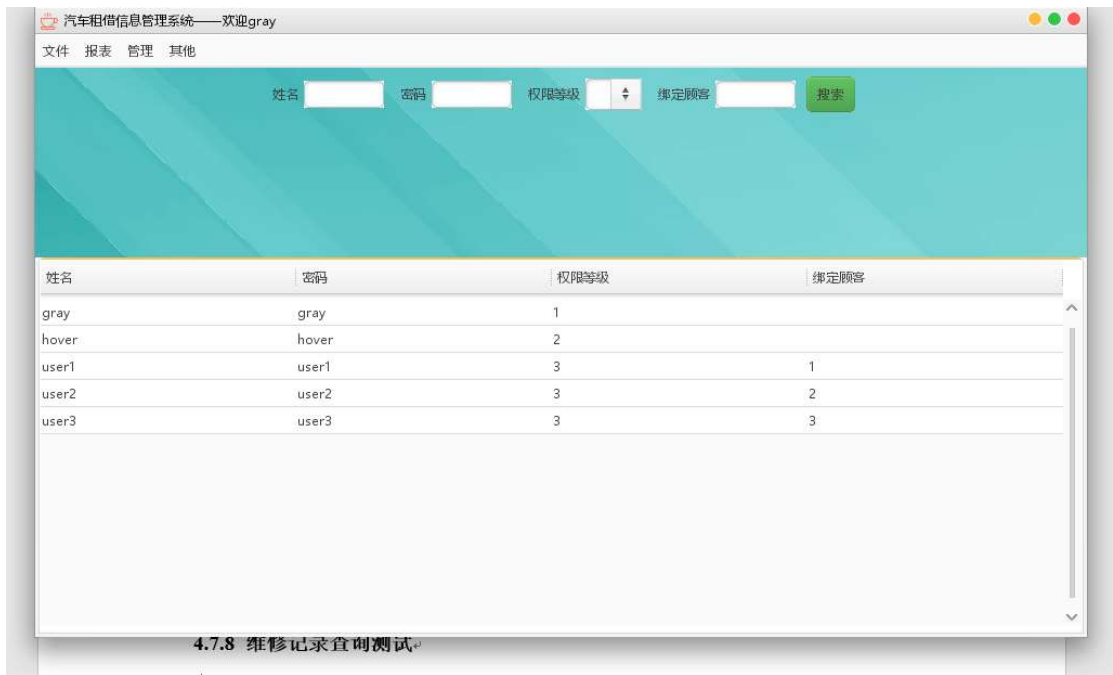


图 4.21

如图 4.22 所示，权限为 2 的普通用户无法查看高级管理员的用户信息，也无法修改其他用户的权限等级与绑定顾客。



图 4.22

如图 4.23 所示，权限为 3 的普通用户无法查看员工表、顾客表，高权限用户的用户信息，事件表中只能查询与自己相关的信息，与此同时无法修改除了自己用户名和密码外的任何数据。



图 4.23

如图 4.24 所示，系统正确生成了低权限用户的查询语句。

```
SELECT info.inford , info.moychange, car.license, customer.id,customer.name, info.event, info.detailevent, info.time, stuff.id,stuff.name
FROM info,car,stuff,customer
WHERE info.license = car.license AND info.stuffid = stuff.id AND customer.id = info.customerid AND customer.id = 1
```

图 4.24

4.7.8 维修记录查询测试

如图 4.25 所示，通过搜索栏中指定维修，即可在事件表中查询维修记录。



图 4.25

4.7.9 租借记录查询测试

如图 4.26、4.27 所示，通过搜索栏中指定租借时间，即可在事件表中查询租借记录数据。



图 4.26



图 4.27

#### 4.7.10 顾客会员查询测试

如图 4.28 所示，通过搜索栏中指定是否会员，即可在顾客表中查询是否为会员。



图 4.28

#### 4.7.11 押金租金记录查询测试

如图 4.29、4.30 所示，系统通过事件表中的流水记录来记录押金租金流入流出情况，当用户租车时流入租金与押金，还车时流出押金。

id	流水
1	2200
3	2200
6	1000
7	1000
9	5300
10	-200
12	-2000

图 4.29

租金	车况	押金
200	5	2000
4000	5	40000
5000	5	50000
200	4	2000
300	5	5000
300	5	5000
250	4	4000

图 4.30

**4.7.12 顾客信誉度查询测试**

如图 4.31 所示，通过菜单栏中打开信誉度统计，即可查看信誉度的统计。  
（信誉度的记录来自不同加权后的用户的日常事件统计，当用户完成租借时提高信誉度，罚款或损坏时降低信誉度）



id	姓名	信誉值
1	张三	-4.5
2	李四	1.0
3	刘五	1.0
5	王七	2.5

图 4.31

## 4.8 系统设计与实现总结

本次课程设计的代码量相对过往几个学期的学期实验来说较多，任务量还是比较大。在本次实验中，我们通过从底层到前端整体地实现了一个数据库管理系统，将数据库系统概论理论课中所学习到的数据库设计思路应用到实际的作品中。

通过初期对数据库功能、数据字典的设计，加深了对于关系模式、E-R图、范式、数据流图等概念的理解。

通过前后端的交互，锻炼了对于 SQL 脚本编写的能力以及对于 SQL 语法的熟悉。在前端界面的设计中，绘制 GUI 以及 GUI 的动态更新刷新等功能上，需要大量的代码进行维护，在这其中也提高了我们对于 Design Pattern 的理解，掌握 Object Oriented Programming 对于工程代码设计管理复用等方面的影响，更是由编写代码的过程中理解到前期数据库设计对于后期编写工程以及使用逻辑优化上的必要性。



实验中主要遇到的难题在于整体 GUI 如何更好地面向用户进行设计，由于比较缺乏针对 UI 的前期设计，在编写代码的时候，主要是想到一步写一步，造成了后期回顾代码的时候经常需要进行重构或甚至是重做逻辑。但好在最后还是完成了一个易用度较为完善的数据库管理系统。

## 附录:

### 代码托管:

完整项目代码资源与版本迭代记录托管于 Github 上:

<https://github.com/GrayXu/HUST-DB-Experiments>

### 源代码:

#### **MainGUI.java:**

```
public class MainGui {

    JPanel jMainPanel;
    JFrame frame;
    private JTable jTable;//当前界面的 Table
    DefaultTableModel tableModel;

    private String userName;
    private JTextField textDialogName;
    private JPasswordField textDialogPsw;
    private JDialog dialogLogin;

    private static int authority;
    private DataBase dataBase;
    private static int SCREEN_WIDTH;
    private static int SCREEN_HEIGHT;

    private String PANEL_MODE = ""; //users, stuff, car ....
    private int delete_row_id = -1;
    private Vector<String> columns;
    private JScrollPane scrollPane;

    public static void main(String[] args) {

        try {

            org.jb2011.lnf.beautyeye.BeautyEyeLNFHelper.launchBeautyEyeLNF();
            BeautyEyeLNFHelper.frameBorderStyle =
            BeautyEyeLNFHelper.FrameBorderStyle.translucencyAppleLike;
```

```

org.jb2011.lnf.beautyeye.BeautyEyeLNFHelper.launchBeautyEyeLNF();
    } catch (Exception e) {
        e.printStackTrace();
    }
    UIManager.put("RootPane.setupButtonVisible", false);
    setFontForBeautyEye();

    Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
    SCREEN_WIDTH = dim.width;
    SCREEN_HEIGHT = dim.height;

    JFrame getFrame = new MainGui().RunBabyRun();

}

/**
 * 修复字体发虚
 */
private static void setFontForBeautyEye() {
    String[] DEFAULT_FONT = new String[] {
        "Table.font"
        , "TableHeader.font"
        , "CheckBox.font"
        , "Tree.font"
        , "Viewport.font"
        , "ProgressBar.font"
        , "RadioButtonMenuItem.font"
        , "ToolBar.font"
        , "ColorChooser.font"
        , "ToggleButton.font"
        , "Panel.font"
        , "TextArea.font"
        , "Menu.font"
        , "TableHeader.font"
        , "OptionPane.font"
        , "MenuBar.font"
        , "Button.font"
        , "Label.font"
        , "PasswordField.font"
        , "ScrollPane.font"
    }
}

```

```

        , "MenuItem.font"
        , "ToolTip.font"
        , "List.font"
        , "EditorPane.font"
        , "Table.font"
        , "TabbedPane.font"
        , "RadioButton.font"
        , "CheckBoxMenuItem.font"
        , "TextPane.font"
        , "PopupMenu.font"
        , "TitledBorder.font"
        , "ComboBox.font"
    };

    for (int i = 0; i < DEFAULT_FONT.length; i++) {
        UIManager.put(DEFAULT_FONT[i], new Font("微软雅黑",
Font.PLAIN, 12));
    }
}

public JFrame RunBabyRun() {
    dataBase = DataBase.getInstance();
    frame = new JFrame();//顺便初始化一下父容器

    if (dataBase.initConnect()) {

        dialogLogin = new JDialog();
        try {
            dialogLogin.setContentPane(new
BackgrouPanel("res/loginBackground.jpg"));
        } catch (IOException e) {
            e.printStackTrace();
        }
        dialogLogin.setTitle("汽车租赁信息系统");

        dialogLogin.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        JLabel labelName = new JLabel("用户名:");
        JLabel labelPsw = new JLabel("密码 :");
    }
}

```

```

labelPsw.setForeground(Color.white);
labelName.setForeground(Color.white);

textDialogName = new JTextField(17);
textDialogPsw = new JPasswordField(10);
JButton butLogin = new JButton("登录");

JPanelOpen namePanel = new JPanelOpen();
namePanel.add(labelName, BorderLayout.WEST);
namePanel.add(textDialogName, BorderLayout.EAST);

JPanelOpen pswPanel = new JPanelOpen();
pswPanel.add(labelPsw, BorderLayout.WEST);
pswPanel.add(textDialogPsw, BorderLayout.EAST);

dialogLogin.getContentPane().setLayout(new BorderLayout());
dialogLogin.getContentPane().add(namePanel,
BorderLayout.NORTH);
dialogLogin.getContentPane().add(pswPanel,
BorderLayout.CENTER);
dialogLogin.getContentPane().add(butLogin, BorderLayout.SOUTH);
butLogin.addActionListener(otherListener);
dialogLogin.setSize(new Dimension(270, 200));
dialogLogin.setResizable(false);
setCenter(dialogLogin);
return frame;

} else {
JOptionPane.showMessageDialog(frame, "数据库连接失败");
return null;
}

}

/**
 * 菜单栏中“其他”的监听器
 */
ActionListener otherListener = (ActionEvent e) -> {
String strClick = e.getActionCommand();
System.out.println(strClick);

```

```

        if (strClick.equals("登录")) {
            //check and set authority
            String nameInput = textDialogName.getText();
            String pswInput = textDialogPsw.getText();
            System.out.println("name:" + nameInput + "\npsw:" + pswInput);
            userName = nameInput;
            try {
                authority = dataBase.checkUser(nameInput, pswInput);
                if (authority != -1) {
                    dialogLogin.setVisible(false);
                    initMainFrame();

                } else {
                    noticeMsg("用户名或密码错误");
                }
            } catch (SQLException e1) {
                e1.printStackTrace();
                noticeMsg("数据库连接失败");
            }
        } else if (strClick.equals("切换账户")) {
            frame.getContentPane().removeAll();
            frame.setVisible(false);
            dialogLogin.setVisible(true);
            PANEL_MODE = "";
        } else if (strClick.equals("说明")) {
            noticeMsg("本信息管理系统基于 Java Swing 进行界面开发，
MySQL 后台支持\n\t——Power by Gray");
        }
    };

    /**
     * 初始化主框架
     */
    public void initMainFrame() {
        try {
            jMainPanel = new BackgrouPanel("res/mainBack2.jpg");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```

frame.setContentPane(jMainPanel);
frame.pack();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setTitle("汽车租赁信息管理系统——欢迎" + userName);
initMenu();
createPopupMenu();
frame.setSize(1025, 635);
setCenter(frame);
frame.setVisible(true);

JTextFiledOpen field = new JTextFiledOpen();
field.setFont(new Font("宋体", Font.BOLD, 30));
field.setText("汽车租赁信息系统 数据库实验 2018 春季");
field.setHorizontalAlignment(JTextField.CENTER);
field.setBorder(null);
field.setEditable(false);

frame.getContentPane().setLayout(new BorderLayout());
frame.getContentPane().add(field, BorderLayout.CENTER);
jMainPanel.updateUI();
}

/**
 * 初始化菜单
 */
private void initMenu() {
    JMenuBar mb = new JMenuBar();
    JMenu menuOther = new JMenu("其他");
    ArrayList<JMenuItem> itemsOther = new ArrayList<>();
    itemsOther.add(new JMenuItem("切换账户"));
    itemsOther.add(new JMenuItem("说明"));
    for (JMenuItem i :
        itemsOther) {
        menuOther.add(i);
        i.addActionListener(otherListener);
    }

    JMenu menuOp = new JMenu("文件");

```

```

ArrayList<JMenuItem> itemsOp = new ArrayList<>();
itemsOp.add(new JMenuItem("导出"));
//    itemsOp.add(new JMenuItem("导入"));
for (JMenuItem i :
        itemsOp) {
    menuOp.add(i);
    i.addActionListener(fileListener);
}

JMenu menuReport = new JMenu("统计");
JMenuItem itemChart = new JMenuItem("统计报表");
JMenuItem itemCredit = new JMenuItem("用户信誉度");
menuReport.add(itemChart);
menuReport.add(itemCredit);
itemChart.addActionListener(e -> showChart());
itemCredit.addActionListener(e -> showCredit());

JMenu menuManage = new JMenu("管理");
ArrayList<JMenuItem> itemManage = new ArrayList<>();
itemManage.add(new JMenuItem("车辆")); //普通用户不能修改车辆表,
we define this logic in updateData function
if (authority != 3) {
    itemManage.add(new JMenuItem("顾客"));
    itemManage.add(new JMenuItem("员工"));
}
//以下两张表普通用户只能看到与他相关的
itemManage.add(new JMenuItem("用户"));
itemManage.add(new JMenuItem("事件"));

for (JMenuItem i :
        itemManage) {
    menuManage.add(i);
    i.addActionListener(changeTableListener);
}
if (authority != 3) {
    mb.add(menuOp);
    mb.add(menuReport);
}

```



```

        mb.add(menuManage);
        mb.add(menuOther);
        frame.setJMenuBar(mb);

    }

    /**
     * 文件选择框
     */
    ActionListener fileListener = new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand();
            JFileChooser jFileChooser = new JFileChooser();
            File file;
            String path = null;
            if (command.equals("导入")) {
                jFileChooser.setSelectionMode(JFileChooser.FILES_ONLY);
                jFileChooser.showOpenDialog(frame);
                file = jFileChooser.getSelectedFile();
                if (file.isFile()) {
                    path = file.getAbsolutePath();
                }
            } else {
                jFileChooser.showSaveDialog(frame);
                file = jFileChooser.getSelectedFile();
                String typeInName = jFileChooser.getName(file);
                if (typeInName == null || typeInName.equals("")) return;
                path = jFileChooser.getCurrentDirectory() + "\\ " + typeInName;
            }

            if (path != null) {
                try {
                    String bat;
                    if (command.equals("导出")) {
                        bat = "C:\\Program Files\\MySQL\\MySQL Server
8.0\\bin\\mysqldump.exe lab3 -uroot " + " -pXIANG1569348 -r" + path + " --skip-
lock-tables";
                    } else {
                        bat = "C:\\Program Files\\MySQL\\MySQL Server

```

```

8.0\\bin\\mysql.exe lab3 -uroot -pXIANG1569348 -e source " + path;
    }
    System.out.println(bat);
    Process process = Runtime.getRuntime().exec(bat);//save
    int com = process.waitFor();
    if (com == 0) {
        noticeMsg("操作成功");
    } else {
        noticeMsg("操作失败");
    }
}

} catch (IOException e1) {
    e1.printStackTrace();
} catch (InterruptedException e1) {
    e1.printStackTrace();
}
}

}

};

/**
 * 展示图表
 */
private void showChart() {
    JDialog chartDialog = new JDialog(frame);
    JPanel contChartPane = new JPanel(new BorderLayout());
    chartDialog.setContentPane(contChartPane);
    chartDialog.setTitle("统计图表");

    JPanel argPanel = new JPanel(new FlowLayout());

    JLabelOpen jLabelF = new JLabelOpen("纵轴: ");
    JLabelOpen jLabelM = new JLabelOpen("横轴: ");

    JComboBox<String> jComboxFunc = new JComboBox<>(new String[]{"
    借车", "还车", "损坏维修", "罚款", "流水"});

```

```
JComboBox<String> jComboxMode = new JComboBox<>(new String[] {"
年", "月", "日"});
```

```
JPanelOpen jPOF = new JPanelOpen();
JPanelOpen jPOM = new JPanelOpen();
```

```
jPOF.add(jLabelF);
jPOF.add(jComboxFunc);
jPOM.add(jLabelM);
jPOM.add(jComboxMode);
```

```
JButton jConfirmBut = new JButton("确定");
jConfirmBut.setUI(new
BEButtonUI().setNormalColor(BEButtonUI.NormalColor.green));
```

```
argPanel.add(jPOF);
argPanel.add(jPOM);
argPanel.add(jConfirmBut);
```

```
JPanelOpen chartPanel = new JPanelOpen();
```

```
jConfirmBut.addActionListener(e -> {
    drawChart(((String) jComboxMode.getSelectedItem()),
              ((String) jComboxFunc.getSelectedItem()),
              chartPanel);
    chartDialog.pack();
});
```

```
contChartPane.add(argPanel, BorderLayout.CENTER);
contChartPane.add(chartPanel, BorderLayout.SOUTH);
```

```
chartDialog.setSize(new Dimension(600, 500));
chartDialog.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
setCenter(chartDialog);
```

```
}
```

```
//绘制图表界面
```

```

private void drawChart(String mode, String func, JPanel panel) {
    panel.removeAll();
    String horizon;
    if (func.equals("流水")) {
        horizon = "元";
    } else {
        horizon = "次数";
    }
    JFreeChart chart = ChartFactory.createBarChart3D(func, mode, horizon,
getChartData(func, mode), PlotOrientation.VERTICAL, false, false, false);

    //设置字体
    CategoryPlot plot = chart.getCategoryPlot();//获取图表区域对象
    CategoryAxis domainAxis = plot.getDomainAxis();           //水平底部
列表
    domainAxis.setLabelFont(new Font("黑体", Font.BOLD, 14));           //
水平底部标题
    domainAxis.setTickLabelFont(new Font("宋体", Font.BOLD, 12)); //垂
直标题
    ValueAxis rangeAxis = plot.getRangeAxis();//获取柱状
    rangeAxis.setLabelFont(new Font("黑体", Font.BOLD, 15));
    chart.getTitle().setFont(new Font("宋体", Font.BOLD, 20));//设置标题字
体
    ChartPanel chartP = new ChartPanel(chart);
    chartP.setOpaque(false);

    panel.add(chartP);

}

//为图表赋值
private CategoryDataset getChartData(String func, String mode) {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    try {
        ArrayList<ArrayList<String>> dataLists =
DataBase.getInstance().getAllChartData(mode, func);
        for (ArrayList<String> list : dataLists) {
            int length = list.size();

```

```

        double value = Double.parseDouble(list.get(length - 1));
        String columnKey;
        if (length == 3) {
            columnKey = list.get(0) + list.get(1);
        } else {
            columnKey = list.get(0);
        }
        dataset.addValue(value, "gray", columnKey);
    }

    } catch (SQLException e) {
        e.printStackTrace();
        noticeMsg("数据库获取数据出错");
    }

    return dataset;
}

/**
 * 窗口放置桌面中央
 *
 * @param c
 */
public void setCenter(Component c) {
//    ((Window) c).pack();
    c.setLocation((SCREEN_WIDTH - c.getWidth()) / 2, (SCREEN_HEIGHT
- c.getHeight()) / 2);
    c.setVisible(true);
}

private void noticeMsg(String in) { //make code elegant
    JOptionPane.showMessageDialog(frame, in);
}

/**
 * initialize search panel
 */
private void setSearchPanel(JPanelOpen jPanelSearch) {
    jPanelSearch.setLayout(new FlowLayout());

```

```

        if (PANEL_MODE.equals("车辆")) {
            setBlankInSearchPanel(jPanelSearch, DataNameUtils.carColumns);
        } else if (PANEL_MODE.equals("顾客")) {
            setBlankInSearchPanel(jPanelSearch,
DataNameUtils.customerColumns);
        }
        if (PANEL_MODE.equals("用户")) {
            setBlankInSearchPanel(jPanelSearch, DataNameUtils.usersColumns);
        }
        if (PANEL_MODE.equals("事件")) {
            setBlankInSearchPanel(jPanelSearch, DataNameUtils.infoColumns);
        }
        if (PANEL_MODE.equals("员工")) {
            setBlankInSearchPanel(jPanelSearch, DataNameUtils.stuffColumns);
        }
        JButton jButton = new JButton("搜索");
        jButton.addActionListener(e -> {
            HashMap<String, String> dataMap = getDataMap(jPanelSearch);
            if (checkSearchData(dataMap)) {
                try {
                    //在这里利用搜索栏中新的约束条件获得新的获取结果
                    Vector<Vector<String>> vectors = null;
                    if (PANEL_MODE.equals("车辆")) {
                        vectors = DataBase.getInstance().getCarLists(dataMap);
                    } else if (PANEL_MODE.equals("顾客")) {
                        vectors =
DataBase.getInstance().getCustomerLists(dataMap);
                    } else if (PANEL_MODE.equals("用户")) {
                        vectors =
DataBase.getInstance().getUserLists(dataMap, authority, userName);
                    } else if (PANEL_MODE.equals("员工")) {
                        vectors =
DataBase.getInstance().getStuffLists(dataMap);
                    } else if (PANEL_MODE.equals("事件")) {
                        vectors = DataBase.getInstance().getInfoLists(dataMap,
authority, userName);
                    }

                    setTablePanel(vectors, columns);
                }
            }
        });
    }
}

```

```

        jMainPanel.add(scrollPane, BorderLayout.SOUTH);
        jMainPanel.updateUI();
    } catch (SQLException e1) {
        e1.printStackTrace();
        noticeMsg("搜索数据不合法或数据库发生错误");
    }
}

});
jButton.setUI(new
BEButtonUI().setNormalColor(BEButtonUI.NormalColor.green));
jPanelSearch.add(jButton);
}

private void setBlankInSearchPanel(JPanelOpen jPanelSearch, String names[]) {
    Set<String> comboBoxSet = comboBoxMap.keySet();

    for (int i = 0; i < names.length; i++) {
        JPanelOpen jPanelO = new JPanelOpen();
        JLabelOpen jLO = new JLabelOpen();
        String key = names[i];
        if (key.equals("顾客") || key.equals("经手员工")) continue;//dirty code

        jLO.setText(key);

        if (comboBoxSet.contains(names[i])) {
            JComboBox<String> stringJComboBox =
getComboBox(names[i]);
            jPanelO.add(jLO);
            jPanelO.add(stringJComboBox);
        } else {
            JTextField jTextField = new JTextField();
            jTextField.setColumns(11);
            jPanelO.add(jLO);
            jPanelO.add(jTextField);
        }

        jPanelSearch.add(jPanelO);
    }
}
}

```

```

private HashMap<String, String[]> comboxMap = new HashMap<String,
String[]>() {{
    put("车况", new String[] {"", "1", "2", "3", "4", "5"});
    put("是否会员", new String[] {"", "Y", "N"});
    put("权限等级", new String[] {"", "1", "2", "3"});
    put("事件", new String[] {"", "损坏维修", "罚款", "借车", "还车"});
}};

private JComboBox<String> getComboBox(String name) {
    JComboBox<String> jComboBox = null;
    String[] contentStrings = comboxMap.get(name);
    if (contentStrings != null) {
        jComboBox = new JComboBox<>(contentStrings);
    }
    return jComboBox;
}

/**
 * 展示信誉度统计
 */
private void showCredit(){
    HashMap<String, Float> dataMap = null;
    HashMap<String, String> nameMap = null;

    try {
        dataMap = DataBase.getInstance().getCredit();
        nameMap = DataBase.getInstance().getMapId2Name();
    } catch (SQLException e) {
        noticeMsg("数据库查询信誉度时发生错误");
        e.printStackTrace();
    }
    if (dataMap == null || nameMap == null) return;

    JDialog dialog = new JDialog(frame);
    dialog.setContentPane(new JPanel());
    dialog.getContentPane().setLayout(new BorderLayout());

```



```

        Set<String> keys = dataMap.keySet();
        Vector<String> columns = new Vector<String>(Arrays.asList("id","姓名","
信誉值"));
        Vector<Vector<String>> datas = new Vector<>();
        for (String key: keys) {
            datas.add(new
Vector<>(Arrays.asList(key,String.valueOf(nameMap.get(key)),String.valueOf(data
Map.get(key)))));
        }

        DefaultTableModel model = new DefaultTableModel(datas,columns);
        JTable creditTable = new JTable(){
            @Override
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        };

        creditTable.getTableHeader().setReorderingAllowed(false);
        creditTable.setModel(model);

        JScrollPane jScrollPane = new JScrollPane(creditTable);
        dialog.getContentPane().add(jScrollPane, BorderLayout.CENTER);
        dialog.pack();
        setCenter(dialog);
        dialog.setTitle("信誉度统计");
        dialog.setVisible(true);
        dialog.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }

    /**
     * 加载全新的表格
     * after called, scroll panel would be reset, you should add scroll panel to
content panel again.
     *
     * @param vectors
     * @param columns
     */
    private void setTablePanel(Vector<Vector<String>> vectors, Vector<String>
columns) {

```

```

        if (scrollPane != null) {
            jMainPanel.remove(scrollPane);
            scrollPane = null;
        }

        tableModel = new DefaultTableModel(vectors, columns) {
            @Override
            public void setValueAt(Object aValue, int row, int column) {
                if (updateData(aValue, row, column)) {
                    super.setValueAt(aValue, row, column); // 在这里做修改值
                }
            }
        };

        /**
         * 修改权限的体现
         */
        if ((PANEL_MODE.equals("车辆") || PANEL_MODE.equals("事件")) &
authority == 3) { // 顾客不能修改车辆表和事件表
            jTable = new JTable() {
                @Override
                public boolean isCellEditable(int row, int column) {
                    return false;
                }
            };
        } else if (PANEL_MODE.equals("用户") & authority != 1) { // 除了超级管
理员，不能修改权限和绑定顾客
            jTable = new JTable() {
                @Override
                public boolean isCellEditable(int row, int column) {
                    return column != 2 && column != 3;
                }
            };
        } else if (PANEL_MODE.equals("事件") && authority != 3) { // 修改事件
表的时候在顾客和经手员工的地方，只能修改对应 id
            jTable = new JTable() {
                @Override
                public boolean isCellEditable(int row, int column) {

```

```

        return column != 4 && column != 9 && column != 0;
    }
};

} else if (columns.get(0).equals("id")) { //id 不能被修改
    jTable = new JTable() {
        @Override
        public boolean isCellEditable(int row, int column) {
            return column != 0;
        }
    };

} else {
    jTable = new JTable() {
        @Override
        public boolean isCellEditable(int row, int column) {
            return true;
        }
    };
}

jTable.getTableHeader().setReorderingAllowed(false);
jTable.setModel(tableModel);
jTable.addMouseListener(new java.awt.event.MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        mouseRightButtonClick(e, jTable);
    }
});

scrollPane = new JScrollPane(jTable);
scrollPane.setPreferredSize(new Dimension(1000, 400));

scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
}

/**
 * 检查单项数据是否格式合法
 *
 * @param name

```

```

* @param value
* @return
*/
public boolean checkDataLegal(String name, String value) {
    //check is legal or not
    if (value.contains("") || value.contains(" ") || value.contains("=")) {
        noticeMsg("新数据中含有非法字段(\"\", \" \", \"=\");");
        return false;
    }
    if (name.equals("事件")) {
        if (DataNameUtils.eventName.indexOf(value) == -1) {
            noticeMsg("事件只有四种类型：借车、还车、损坏维修、罚
款");
            return false;//不合法事件代码
        }
    }
    if (name.equals("时间")) {
        if (value.length() != 10 || value.charAt(4) != '-' || value.charAt(4) != '-' ||
Pattern.compile("[^\\d-]").matcher(value).find()) {
            noticeMsg("时间格式错误");
            return false;
        }
    }
    if (name.equals("车牌号")) {
        if (value.length() != 7) {
            noticeMsg("车牌号格式错误");
            return false;
        }
    }
    if (name.equals("车况")) {
        if (Pattern.compile("[^\\d]+").matcher(value).find()) { //保证全数字，
防止后面强转出错
            noticeMsg("只能输入数字");
            return false;
        }
        if (Integer.valueOf(value) < 1 || Integer.valueOf(value) > 5) {
            noticeMsg("只能输入 1-5 的数字");
            return false;
        }
    }
}

```

```

        }
    }
    if (name.equals("是否会员")) {
        if (!value.equals("Y") && !value.equals("N")) {
            noticeMsg("只能输入 Y 或 N");
            return false;
        }
    }
    if (name.equals("权限等级")) {
        if (Pattern.compile("[^\\d]+").matcher(value).find()) { //保证全数字,
防止后面强转出错
            noticeMsg("只能输入数字");
            return false;
        }
        int valueInt = Integer.valueOf(value);
        if (valueInt < 1 || valueInt > 3) {
            noticeMsg("只能输入 1-3 的数字");
            return false;
        }
        if (valueInt != 3 && authority != 1) {
            noticeMsg("无权限操作");
            return false;
        }
    }

    return true;
}

```

//判断是否为鼠标的 BUTTON3 按钮，BUTTON3 为鼠标右键

private void mouseRightButtonClick(MouseEvent evt, JTable jTable) {

```

    if (evt.getButton() == java.awt.event.MouseEvent.BUTTON3) {
        //通过点击位置找到点击为表格中的行
        int focusedRowIndex = jTable.rowAtPoint(evt.getPoint());
        if (focusedRowIndex == -1) {
            return;
        }
        delete_row_id = focusedRowIndex;
        //将表格所选项设为当前右键点击的行
    }
}

```

```

        jTable.setRowSelectionInterval(focusedRowIndex,
focusedRowIndex);
        //弹出菜单
        if (jPopupMenu != null) {
            jPopupMenu.show(jTable, evt.getX(), evt.getY());
        }
    }
}

/**
 * 修改表格界面的监听器
 */
ActionListener changeTableListener = e -> {
    String strClick = e.getActionCommand();
    System.out.println(strClick);
    if (!PANEL_MODE.equals(strClick)) { //界面如已加载过则不继续加载
        PANEL_MODE = strClick;
        frame.getContentPane().removeAll(); //移除原有的东西

        Vector<Vector<String>> vectors = null; //数据 vector

        try {
            if (strClick.equals("顾客")) {
                columns = new
Vector<>(Arrays.asList(DataNameUtils.customerColumns));
                vectors = DataBase.getInstance().getCustomerLists();
            } else if (strClick.equals("车辆")) {
                columns = new
Vector<>(Arrays.asList(DataNameUtils.carColumns));
                vectors = DataBase.getInstance().getCarLists();
            } else if (strClick.equals("员工")) {
                columns = new
Vector<>(Arrays.asList(DataNameUtils.stuffColumns));
                vectors = DataBase.getInstance().getStuffLists();
            } else if (strClick.equals("用户")) {
                columns = new
Vector<>(Arrays.asList(DataNameUtils.usersColumns));
                vectors = DataBase.getInstance().getUserLists(authority,

```

```

userName);
        } else if (strClick.equals("事件")) {
            columns = new
Vector<>(Arrays.asList(DataNameUtils.infoColumns));
            vectors = DataBase.getInstance().getInfoLists(authority,
userName);
        }
    } catch (SQLException e1) {
        e1.printStackTrace();
        noticeMsg("数据库发生错误");
    }

    if (vectors != null && columns != null) {
        setTablePanel(vectors, columns);
        //重新渲染整个界面
        JPanelOpen jPanelSearch = new JPanelOpen();
        setSearchPanel(jPanelSearch);
        jMainPanel.add(jPanelSearch, BorderLayout.CENTER);
        jMainPanel.add(scrollPane, BorderLayout.SOUTH);
        jMainPanel.updateUI();
    }

}

};

private JPopupMenu jPopupMenu;

//右键菜单
private void createPopupMenu() {
    if (authority == 3) return;
    jPopupMenu = new JPopupMenu();

    JMenuItem delMenuItem = new JMenuItem();
    delMenuItem.setText("删除本行");
    delMenuItem.addActionListener(evt -> {
        String key = (String) jTable.getValueAt(delete_row_id, 0);
        try {
            DataBase.getInstance().deleteRow(PANEL_MODE, key);
            tableModel.removeRow(delete_row_id);
        } catch (SQLException e) {

```

```

        e.printStackTrace();
        noticeMsg("删除失败");
    }
});

JMenuItem addMenuItem = new JMenuItem();
addMenuItem.setText("添加新行");
addMenuItem.addActionListener(evt -> {
    setAddRowDialog();
});

jPopupMenu.add(delMenuItem);
jPopupMenu.add(addMenuItem);
}

JDialog dialogAddRow;

private void setAddRowDialog() {
    //create a dialog
    dialogAddRow = new JDialog(frame);
    dialogAddRow.setTitle("添加");
    dialogAddRow.setLayout(new FlowLayout());
    JPanel contentPanel = new JPanel();
    dialogAddRow.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

    String[] columnNames =
    DataNameUtils.getColumnNamesByMode(PANEL_MODE);

    Set<String> comboxSet = comboxMap.keySet();

    for (String s : columnNames) {
        if (!s.equals("id") && !s.equals("顾客") && !s.equals("经手员工")) {
            JPanel jPanel = new JPanel();
            JLabel jLabel = new JLabel(s);
            jPanel.add(jLabel, BorderLayout.WEST);
            if (comboxSet.contains(s)) {
                jPanel.add(getComboBox(s), BorderLayout.EAST);
            } else {
                JTextField jTextField = new JTextField(10);
                jPanel.add(jTextField, BorderLayout.EAST);
            }
        }
    }
}

```



```

        }
        contentPanel.add(jPanel);
    }
}
JButton jButton = new JButton("确认");

jButton.addActionListener(e -> {
    //set this hashmap
    HashMap<String, String> map = getDataMap((JPanel)
dialogAddRow.getContentPane());
    if (checkNewData(map)) {
        try {
            DataBase.getInstance().addRow(PANEL_MODE, map);
            tableModel.addRow(map2vector(map));
            dialogAddRow.dispose();
        } catch (SQLException e1) {
            e1.printStackTrace();
            noticeMsg("新数据不合法或数据库发生错误");
        }
    }
});

contentPanel.add(jButton);
dialogAddRow.setContentPane(contentPanel);
dialogAddRow.pack();
setCenter(dialogAddRow);
}

private boolean checkNewData(HashMap<String, String> map) {
    Set<String> set = map.keySet();
    Iterator<String> i = set.iterator();
    while (i.hasNext()) {
        String key = i.next();
        String value = map.get(key);
        if (!checkDataLegal(key, value)) { //一个个检查
            return false;
        }
    }
    return true;
}

```

```

    }

    private boolean checkSearchData(HashMap<String, String> map) {
        Set<String> set = map.keySet();
        Iterator<String> i = set.iterator();
        while (i.hasNext()) {
            String key = i.next();
            String value = map.get(key);
            if (value.equals("")) continue;
            if (!checkDataLegal(key, value)) { //一个个检查
                return false;
            }
        }
        return true;
    }

    /**
     * 更新表格数据
     */
    private boolean updateData(Object aValue, int row, int column) {

        String[] columnNames =
        DataNameUtils.getColumnNamesByMode(PANEL_MODE);
        try {
            //          if (checkDataLegal(columnNames[column], aValue.toString(),
            (String) jTable.getValueAt(row, 0))) {
                if (checkDataLegal(columnNames[column], aValue.toString())) {
                    DataBase.getInstance().updateData(PANEL_MODE,
                    columnNames[column], aValue.toString(), (String) jTable.getValueAt(row, 0));
                    return true;
                } else {
                    return false;
                }
            }

        } catch (SQLException e) {
            noticeMsg("新数据格式不合法");
            e.printStackTrace();
            return false;
        }
    }
}

```

```

/**
 * 获得用户填入的新行的数据
 * 传入一个 JPanel 父容器
 * 传出的字符串可能有带空
 *
 * @return
 */
private HashMap<String, String> getDataMap(JPanel jPanelIn) {
    HashMap<String, String> map = new HashMap<>();
    Component[] components = jPanelIn.getComponents();
    for (int i = 0; i < components.length - 1; i++) { //不算 button
        JPanel jPanelGet = (JPanel) components[i];
        String key = ((JLabel) jPanelGet.getComponents()[0]).getText();
        if (key.equals("顾客") || key.equals("经手员工")) {
            continue;
        }

        String value = null;
        try {
            value = ((JTextField) jPanelGet.getComponents()[1]).getText();
        } catch (Exception e) {
            value = (String) ((JComboBox)
jPanelGet.getComponents()[1]).getSelectedItem();
        }
        map.put(key, value);
        System.out.println("key: " + key + "; value: " + value);
    }
    return map;
}

/**
 * hashmap 转 vector, 方便 addRow
 *
 * @param map
 * @return
 */
private Vector<String> map2vector(HashMap<String, String> map) {

```

```

        Vector<String> vector = new Vector<>();

        String[] columnNames =
        DataNameUtils.getColumnNamesByMode(PANEL_MODE);
        for (String s : columnNames) {
            String v = map.get(s);
            if (v != null) {
                vector.add(v);
            } else {
                vector.add("");
            }
        }

        return vector;
    }
}

```

#### **DataBase.java:**

```

public class DataBase implements DataBaseIF {

    public static final String URL =
    "jdbc:mysql://localhost:3306/lab3?serverTimezone=UTC&useSSL=false";
    public static final String USER = "root";
    public static final String PASSWORD = "XIANG1569348";

    private Connection connection;

    /**
     * singleton
     */
    private static class InnerHelper {
        private final static DataBase dataBase = new DataBase();
    }

    private DataBase() {
    }

    public static DataBase getInstance() {

```

```

        return InnerHelper.dataBase;
    }

    /**
     * @return status of connection
     */

    public boolean initConnect() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, USER,
PASSWORD);
            return true;
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    /**
     * return users' authority
     *
     * @param name 输入的用户名
     * @param psw 输入的密码
     * @return authority, -1->no found, 1->root user, 2->administrator, 3->normal
user
     * @throws SQLException
     */

    @Override
    public int checkUser(String name, String psw) throws SQLException {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM users");//
全表搜索
        while (resultSet.next()) {
            if (resultSet.getString("name").equals(name)) {
                if (resultSet.getString("password").equals(psw)) {
                    return resultSet.getInt("author");
                }
            }
        }
    }

```

```

        }
    }
    statement.close();
    return -1;
}

//从约束里拿 where 子句
private String getWhereClause(HashMap<String, String> map, boolean
haveWhere) {
    StringBuilder sql = new StringBuilder();
    boolean isFirst = true;
    if (haveWhere) {
        isFirst = false;
    }
    Set<String> keySet = map.keySet();

    for (String key : keySet) {
        String value = map.get(key);
        if (value.equals("")) continue;
        if (isFirst) {
            sql.append("WHERE ");
            isFirst = false;
        } else {
            sql.append("and ");
        }

        if (DataNameUtils.name2name(key).equals("event"))
            value =
String.valueOf(DataNameUtils.eventName.indexOf(value) + 1);
        sql.append(DataNameUtils.name2name(key) + " = '" + value + "'");
    }
    return String.valueOf(sql);
}

//压一个二维 list 出来
private Vector<Vector<String>> pullVectors(ResultSet resultSet, int length)
throws SQLException {
    Vector<Vector<String>> vectors = new Vector<>();
    while (resultSet.next()) {
        Vector<String> tempVec = new Vector<>();

```

```

        for (int i = 1; i <= length; i++) { //从 1 开始
            tempVec.add(resultSet.getString(i));
        }
        vectors.add(tempVec);
    }
    return vectors;
}

/**
 * getLists 为从数据库中获得对应的数据，只能支持普通查询（车辆，顾客，）
 * three override methods for each one
 *
 * @return 获得的数据项
 * @throws SQLException 语句错误或者受到约束
 */
@Override
public Vector<Vector<String>> getCarLists() throws SQLException {
    String sql = "SELECT * FROM car ";
    return getCarLists(sql);
}

@Override
public Vector<Vector<String>> getCarLists(HashMap<String, String> map)
throws SQLException {
    String sql = "SELECT * FROM car ";
    sql = sql + getWhereClause(map, false);
    return getCarLists(sql);
}

@Override
public Vector<Vector<String>> getCarLists(String sql) throws SQLException {
    Statement statement = connection.createStatement();
    System.out.println(sql);
    ResultSet resultSet = statement.executeQuery(sql);
    Vector<Vector<String>> vectors = pullVectors(resultSet,
DataNameUtils.carColumns.length);
    statement.close();
    return vectors;
}

```

```

@Override
public Vector<Vector<String>> getCustomerLists() throws SQLException {
    String sql = "SELECT * FROM customer ";
    return getCustomerLists(sql);
}

@Override
public Vector<Vector<String>> getCustomerLists(HashMap<String, String>
map) throws SQLException {
    String sql = "SELECT * FROM customer ";
    sql = sql + getWhereClause(map, false);
    return getCustomerLists(sql);
}

@Override
public Vector<Vector<String>> getCustomerLists(String sql) throws
SQLException {
    Statement statement = connection.createStatement();
    System.out.println(sql);
    ResultSet resultSet = statement.executeQuery(sql);
    Vector<Vector<String>> vectors = pullVectors(resultSet,
DataNameUtils.customerColumns.length);
    statement.close();
    return vectors;
}

@Override
public Vector<Vector<String>> getStuffLists() throws SQLException {
    String sql = "SELECT * FROM stuff ";
    return getStuffLists(sql);
}

@Override
public Vector<Vector<String>> getStuffLists(HashMap<String, String> map)
throws SQLException {
    String sql = "SELECT * FROM stuff ";
    sql = sql + getWhereClause(map, false);
    return getStuffLists(sql);
}

```



```

@Override
public Vector<Vector<String>> getStuffLists(String sql) throws SQLException
{
    Statement statement = connection.createStatement();
    System.out.println(sql);
    ResultSet resultSet = statement.executeQuery(sql);
    Vector<Vector<String>> vectors = pullVectors(resultSet,
DataNameUtils.stuffColumns.length);
    statement.close();
    return vectors;
}

//权限 3 用户只能查看自己的用户信息，和修改用户名和密码
@Override
public Vector<Vector<String>> getUserLists(int authority, String userName)
throws SQLException {
    String sql = null;
    if (authority == 1) {
        sql = "SELECT * FROM users ";
    } else if (authority == 2) {
        sql = "SELECT * FROM users WHERE author < 1 ";
    } else if (authority == 3) {
        sql = "SELECT * FROM users WHERE name = '" + userName + "' ";
    }
    return getUserLists(sql);
}

@Override
public Vector<Vector<String>> getUserLists(HashMap<String, String> map, int
authority, String userName) throws SQLException {
    String sql = null;
    if (authority == 1) {
        sql = "SELECT * FROM users ";
        sql = sql + getWhereClause(map, false);
    } else if (authority == 2) {
        sql = sql + getWhereClause(map, true);
        sql = "SELECT * FROM users WHERE author < 1 ";
    } else if (authority == 3) {

```

```

        sql = "SELECT * FROM users WHERE name = '" + userName + "' ";
        sql = sql + getWhereClause(map, true);
    }
    return getUserLists(sql);
}

@Override
public Vector<Vector<String>> getUserLists(String sql) throws SQLException
{
    Statement statement = connection.createStatement();
    System.out.println(sql);
    ResultSet resultSet = statement.executeQuery(sql);
    Vector<Vector<String>> vectors = pullVectors(resultSet,
DataNameUtils.usersColumns.length);
    statement.close();
    return vectors;
}

@Override
public Vector<Vector<String>> getInfoLists(String sql) throws SQLException {
    Statement statement = connection.createStatement();
    System.out.println(sql);
    ResultSet resultSet = statement.executeQuery(sql);

    Vector<Vector<String>> vectors = new Vector<>();
    while (resultSet.next()) {
        Vector<String> tempVec = new Vector<>();
        for (int i = 1; i <= DataNameUtils.infoColumns.length; i++) {从 1 开
始
            if (i == 6) {
                switch (resultSet.getInt(i)) {eventid to event
                    case 1:
                        tempVec.add("损坏维修");
                        break;
                    case 2:
                        tempVec.add("罚款");
                        break;
                    case 3:
                        tempVec.add("借车");
                        break;

```

```

        case 4:
            tempVec.add("还车");
            break;
        }
    } else {
        tempVec.add(resultSet.getString(i));
    }
}
vectors.add(tempVec);
}

statement.close();
return vectors;
}

@Override
public Vector<Vector<String>> getInfoLists(HashMap<String, String> map, int
authority, String userName) throws SQLException {
    String sql = "SELECT info.infoId , info.moychange, car.license,
customer.id,customer.name, info.event, info.detailevent, info.time,
stuff.id,stuff.name\n" +
        "FROM info,car,stuff,customer\n" +
        "WHERE info.license = car.license AND info.stuffid = stuff.id
AND customer.id = info.customerid ";
    sql = sql + getWhereClause(map, true);
    return getInfoLists(sql);
}

@Override
public Vector<Vector<String>> getInfoLists(int authority, String userName)
throws SQLException {
    String sql = "SELECT info.infoId , info.moychange, car.license,
customer.id,customer.name, info.event, info.detailevent, info.time,
stuff.id,stuff.name\n" +
        "FROM info,car,stuff,customer\n" +
        "WHERE info.license = car.license AND info.stuffid = stuff.id
AND customer.id = info.customerid ";
    if (authority == 3) {
        sql = sql + "AND customer.id = " + getIDbyUserName(userName) + "
";
    }
}

```

```

    }
    return getInfoLists(sql);
}

@Override
public String getIDbyUserName(String name) throws SQLException {
    String sql = "SELECT customerid FROM users WHERE NAME = '" +
name + "'";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sql);
    Vector<String> vector = new Vector<>();
    while (resultSet.next()) {
        vector.add(resultSet.getString(1));
    }
    statement.close();
    return vector.get(0);
}

/**
 * 在数据库中级联删除一行
 *
 * @param tableMode 表格模式的标志字符串
 * @param primaryKey 主键的值
 * @throws SQLException 语句错误或者受到约束
 */

@Override
public void deleteRow(String tableMode, String primaryKey) throws
SQLException {
    String tableName = DataNameUtils.tableMode2Name(tableMode);

    String primaryKeyName =
DataNameUtils.primaryKeyMap.get(tableName);
    if (primaryKeyName == null) primaryKeyName = "id";
    String sql = "DELETE FROM " + tableName + " WHERE " +
primaryKeyName + " = ?";
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setString(1, primaryKey);
    System.out.println(preparedStatement.toString());
    preparedStatement.execute();
}

```

```

    }

    /**
     * 更新数据
     *
     * @param tableMode 表格模式的标志字符串
     * @param name      更新属性的列名
     * @param value      更新后的值
     * @param primaryKey 主键的值
     * @throws SQLException 语句错误或者受到约束
     */

    @Override
    public void updateData(String tableMode, String name, String value, String
primaryKey) throws SQLException {
        String tableName = DataNameUtils.tableMode2Name(tableMode);

        if (name.equals("事件")) value =
String.valueOf(DataNameUtils.eventName.indexOf(value) + 1);

        //拿到主键的名字
        String primaryKeyName =
DataNameUtils.primaryKeyMap.get(tableName);
        if (primaryKeyName == null) primaryKeyName = "id";

        String sql = "UPDATE " + tableName + " SET " +
DataNameUtils.name2name(name) + " = ? where " + primaryKeyName + " = ?";
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        if (value.equals("")) {
            preparedStatement.setNull(1, Types.CHAR); //this types.* is useless...
        } else {
            preparedStatement.setString(1, value);
        }
        preparedStatement.setString(2, primaryKey);
        System.out.println(preparedStatement.toString());
        preparedStatement.execute();
    }

```

```

/**
 * 添加行
 *
 * @param tableMode 表格模式的标志字符串
 * @param data      一张哈希表存新数据
 * @throws SQLException 语句错误或者受到约束
 */

@Override
public void addRow(String tableMode, HashMap<String, String> data) throws
SQLException {
    Statement statement = connection.createStatement();
    String tableName = DataNameUtils.tableMode2Name(tableMode);
    if (tableName != null) {
        StringBuilder columns = new StringBuilder("");
        StringBuilder values = new StringBuilder("");
        Set<String> keySet = data.keySet();
        Iterator<String> iterator = keySet.iterator();

        boolean isFirst = true;
        while (iterator.hasNext()) {
            String s = iterator.next();
            System.out.println(s);
            String value = data.get(s);
            if (DataNameUtils.eventName.contains(value))
                value =
String.valueOf(DataNameUtils.eventName.indexOf(value) + 1);
            if (value != null && !value.equals("")) {
                if (isFirst) { // 第一个不加逗号
                    isFirst = false;
                    columns.append(DataNameUtils.name2name(s));
                    values.append(""" + value + """);
                } else {
                    columns.append(", " + DataNameUtils.name2name(s));
                    values.append(", "" + value + """);
                }
            }
        }

        String sql = "INSERT INTO " + tableName + " (" + columns.toString()

```

```

+ ") VALUES (" + values.toString() + "));
        System.out.println(sql);
        statement.execute(sql);
        statement.close();
    }
}

/**
 * 获得所有图表数据
 * 1->year 2->month 3->day
 * func:借车 还车 损坏维修 罚款 流水
 *
 * @return
 * @throws SQLException
 */
@Override
public ArrayList<ArrayList<String>> getAllChartData(String mode, String
func) throws SQLException {

    String selectItem;
    String whereClause = "";
    if (func.equals("流水")) {
        selectItem = "SUM(moychange)";
    } else {
        selectItem = "COUNT(TIME)"; // 计算次数
        int indexid = DataNameUtils.eventName.indexOf(func) + 1;
        whereClause = " WHERE event = " + String.valueOf(indexid);
    }

    String sql = null;
    if (mode.equals("年")) {
        sql = "SELECT YEAR(TIME), " + selectItem + " FROM info " +
whereClause + " GROUP BY YEAR(TIME)";
    } else if (mode.equals("月")) {
        sql = "SELECT YEAR(TIME),MONTH(TIME), " + selectItem + "
FROM info " + whereClause + " GROUP BY YEAR(TIME),MONTH(TIME)";
    } else if (mode.equals("日")) {
        sql = "SELECT TIME, " + selectItem + " FROM info " + whereClause
+ " GROUP BY TIME";
    }
}

```

```

    }

    System.out.println(sql);
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sql);

    ArrayList<ArrayList<String>> lists = new ArrayList<>();
    while (resultSet.next()) {
        ArrayList<String> listTemp = new ArrayList<>();
        listTemp.add(resultSet.getString(1));
        listTemp.add(resultSet.getString(2));
        if (mode.equals("月")) listTemp.add(resultSet.getString(3));
        lists.add(listTemp);
    }
    statement.close();
    return lists;
}

public HashMap<String, Float> getCredit() throws SQLException {
    HashMap<String, Float> hashMap = new HashMap<>();

    for (int i = 1; i <= 4; i++) {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(getCreditClause(i));
        while (resultSet.next()) {
            String key = resultSet.getString(1);
            int times = resultSet.getInt(2);
            Float oldValue = hashMap.get(key);
            if (oldValue == null) oldValue = 0f;
            switch (i) {
                case 1:
                    hashMap.put(key, oldValue - 3 * times);
                    break;
                case 2:
                    hashMap.put(key, oldValue - 5 * times);
                    break;
                case 3:
                    hashMap.put(key, oldValue + 1 * times);
                    break;
                case 4:

```



```

        hashMap.put(key, (float) (oldValue + 1.5 * times));
        break;
    }
}
statement.close();
}

return hashMap;
}

private String getCreditClause(int eventid) {
    return "SELECT customerid,COUNT(customerid) \n" +
        "FROM info\n" +
        "WHERE EVENT = " + eventid + "\n" +
        "GROUP BY customerid";
}

public HashMap<String, String> getMapId2Name() throws SQLException {
    HashMap<String, String> hashMap = new HashMap<>();
    Statement statement = connection.createStatement();
    String sql = "SELECT id, NAME\n" +
        "FROM customer";
    ResultSet resultSet = statement.executeQuery(sql);
    while (resultSet.next()) {
        hashMap.put(resultSet.getString(1), resultSet.getString(2));
    }
    return hashMap;
}
}

```

### **DataNameUtils.java:**

```

package Support;

import org.jetbrains.annotations.Nullable;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;

```

```

/**
 * 本类用来管理获得数据库中的列名、属性以及在 GUI 中的名字
 * 在修改数据库后记得查看本类进行检查。
 * 注意不能持有外部引用，防止内存泄露。
 */
public class DataNameUtils {

    public static String[] customerColumns = new String[]{"id", "姓名", "年龄", "是否会员"};

    public static String[] carColumns = new String[]{"车牌号", "品牌", "租金", "车况", "押金"};

    public static String[] stuffColumns = new String[]{"id", "姓名", "年龄"};

    public static String[] usersColumns = new String[]{"姓名", "密码", "权限等级", "绑定顾客"};

    public static String[] infoColumns = new String[]{"id", "流水", "车牌号", "顾客id", "顾客", "事件", "备注", "时间", "经手员工 id", "经手员工"};

    public static HashMap<String, String> primaryKeyMap = new HashMap<String, String>() {{
        put("users", "name");
        put("car", "license");
        put("info", "infoid");
    }};

    public static String[] getColumnNamesByMode(String mode) {
        String[] columnNames = new String[0];
        if (mode.equals("用户")) {
            columnNames = DataNameUtils.usersColumns;
        } else if (mode.equals("顾客")) {
            columnNames = DataNameUtils.customerColumns;
        } else if (mode.equals("员工")) {
            columnNames = DataNameUtils.stuffColumns;
        } else if (mode.equals("车辆")) {
            columnNames = DataNameUtils.carColumns;
        } else if (mode.equals("事件")) {
            columnNames = DataNameUtils.infoColumns; //这里应该用更好的办法
        }
    }
}

```

```

    }
    return columnNames;
}

private static HashMap<String, String> name2nameMap = new
HashMap<String, String>() {{
    put("姓名", "name");
    put("品牌", "brand");
    put("车牌号", "license");
    put("租金", "cost");
    put("车况", "status");
    put("年龄", "age");
    put("是否会员", "member");
    put("密码", "password");
    put("权限等级", "author");
    put("押金", "pledge");
    put("绑定顾客", "customerid");
    put("顾客 id", "customerid");
    put("经手员工 id", "stuffid");
    put("事件", "event");
    put("流水", "moychange");
    put("备注", "detailevent");
    put("时间", "time");
    put("id", "id");
}};

@Nullable
public static String name2name(String strIn) {
    return name2nameMap.get(strIn);
}

public static ArrayList<String> eventName = new ArrayList<>(Arrays.asList("
损坏维修", "罚款", "借车", "还车"));

public static String swtichEventId(String strIn) {
    return String.valueOf(eventName.indexOf(strIn));
}

```

```

        private static HashMap<String, String> tableMode2NameMap = new
HashMap<String, String>() {{
            put("车辆", "car");
            put("顾客", "customer");
            put("事件", "info");
            put("员工", "stuff");
            put("用户", "users");
        }};

        public static String tableMode2Name(String tableMode) {
            return tableMode2NameMap.get(tableMode);
        }
    }

```