

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： |  |
| 姓 名： | 王占成 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1707 |
| 学 号： | U201714786 |
| 指导教师： | 吴海 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

20 年 月 日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

目 录

[1 课程任务概述 1](#_Toc482021027)

[2实验任务一（标题改为具体任务名称） 2](#_Toc482021028)

[2.1 任务要求 2](#_Toc482021029)

[2.2 完成过程 3](#_Toc482021030)

[2.3任务总结 3](#_Toc482021031)

[3实验任务二（标题改为具体任务名称） 3](#_Toc482021032)

[3.1 任务要求 3](#_Toc482021033)

[3.2 完成过程 3](#_Toc482021034)

[3.3任务总结 3](#_Toc482021035)

[4 综合实践任务 3](#_Toc482021036)

[4.1系统设计目标 3](#_Toc482021037)

[4.2 需求分析 3](#_Toc482021038)

[4.3 总体设计 4](#_Toc482021039)

[4.4 数据库设计 4](#_Toc482021040)

[4.5 详细设计与实现 4](#_Toc482021041)

[4.6 系统测试 4](#_Toc482021042)

[4.7 系统设计与实现总结 4](#_Toc482021043)

[4 课程总结 4](#_Toc482021044)

[附录 4](#_Toc482021045)

# 1 课程任务概述

实验主要分成三部分：软件功能学习部分、SQL联系部分和综合实践部分。

1. 软件功能学习部分：实验指定使用SQL server数据库，该部分主要是熟悉该数据库的操作，包括创建用户，设置权限，备份数据和恢复数据等。
2. SQL练习部分：用于熟悉sql语句的使用，包括表的创建、更新以及查询语句，所有的查询都基于乘客与火车的关系，共16条查询语句，涵盖了大部分sql语法。
3. 综合实践部分：提供若干个题目供选择，如工资管理系统、超市收银系统等，每个系统都需要根据实际情况设计并创建若干个表，选用一种编程语言实现一个客户端程序/网站，要求数据库关系设计合理，客户端要有良好的人机交互界面。

# 软件功能学习部分

## 2.1 任务要求

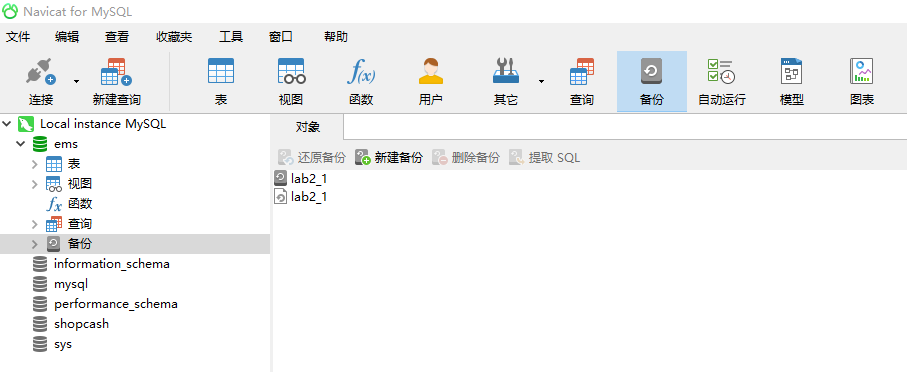
1. 练习SQL Server或其他某个主流关系数据库管理系统软件的备份方式，要求要有通过数据库的软件功能进行的备份和通过文件形式的脱机备份。
2. 练习在新增的数据库上增加用户并配置权限的操作。

## 2.2 完成过程

### 2.2.1 脱机备份

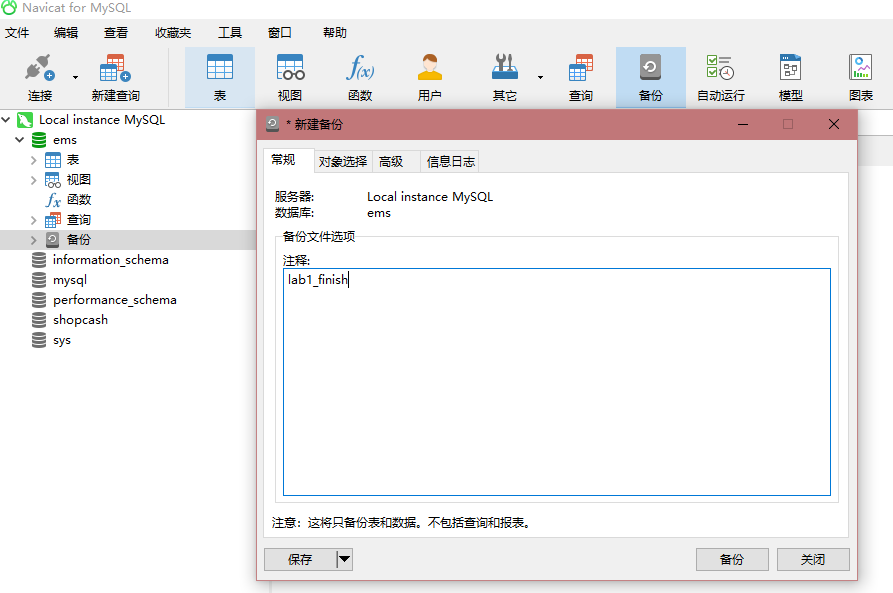
### 2.2.2系统备份

1. 选中数据库ems，通过双击备份打开备份选项卡，如下：



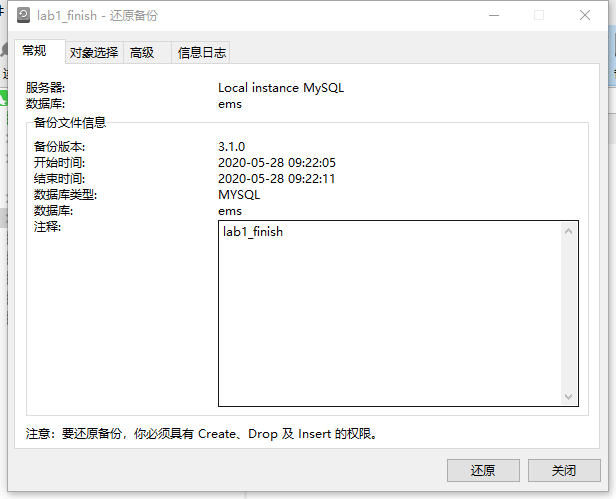
数据库系统备份

1. 备份到如下文件，并且将备份后的文件名重命名为lab1\_finish如下：



数据库备份位置

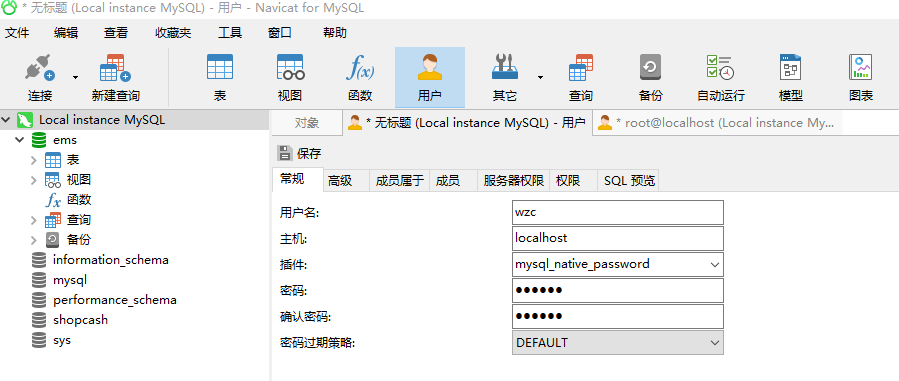
1. 选中备份文件lab1\_finish，点击还原，即可将数据库还原到之前的状态，如下：



数据库还原

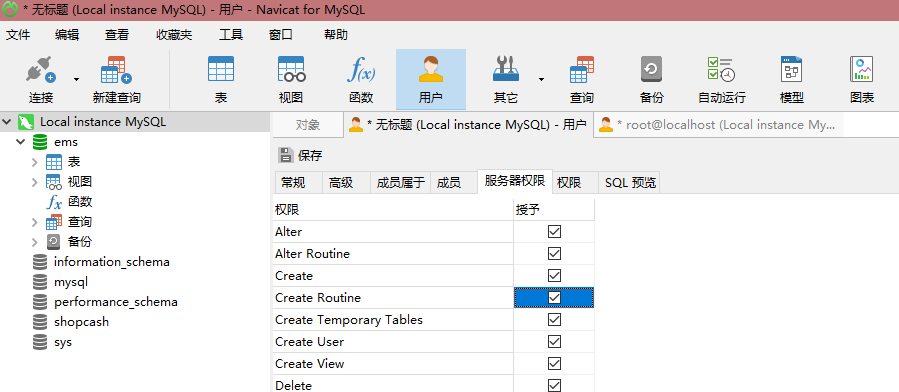
### 2.2.3新增用户

1. 选中用户，点击创建用户即可创建一个新的用户，如下：



新增用户

1. 给用户wzc授予所有权限，授权操作如下：



用户授权

1. 登录数据库，如下：



登录数据库

## 2.3任务总结

（总结实验中遇到的主要问题及所采用的解决方案）

# 3实验任务二（标题改为具体任务名称）

## 3.1 任务要求

1. 建表：在DBMS中创建指定的关系表，包括主码和外码的说明，写出指定关系的建表SQL语句。
2. 数据更新：使用SQL语句对各个表的增删改查、数据导入导出、编写触发器。
3. 查询：使用SQL语句完成相应的小题。

## 3.2 完成过程

### 3.2.1建表

1. 创建数据库ems，如下图3 . 1所示：

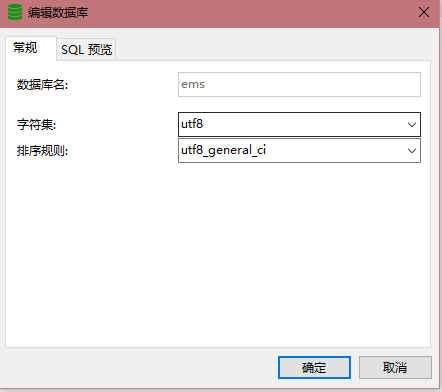


图3 . 1创建数据库

1. 使用数据库ems，并且创建所有的关系表，建表SQL语句如下：

SET SCHEMA ems;

create table Station(

SID int,

SName char(20),

CityName char(30),

primary key (SID)

);

create table Train(

TID int,

SDate date,

TName char(20),

SStationID int,

AStationID int,

STime time,

ATime time,

primary key (TID),

unique (SDate,TName),

foreign key (SStationID) references Station(SID),

foreign key (AStationID) references Station(SID)

);

create table TrainPass(

TID int,

SNo smallint,

SID int,

STime time,

ATime time,

primary key (TID,SNo),

foreign key (TID) references Train(TID),

foreign key (SID) references Station(SID)

);

create table Passenger(

PCardID char(18),

PName char(20),

Sex bit,

Age smallint,

primary key (PCardID)

);

create table TakeTrainRecord(

RID int,

PCardID char(18),

TID int,

SStationID int,

AStationID int,

CarrigeID smallint,

SeatRow smallint,

SeatNo char(1),

SStatus int,

primary key (RID),

foreign key (PCardID) references Passenger(PCardID),

foreign key (TID) references Train(TID),

foreign key (SStationID) references Station(SID),

foreign key (AStationID) references Station(SID),

constraint c1 check (SeatNo in ('A','B','C','E','F')),

constraint c2 check (SStatus in (0,1,2))

);

create table DiagnoseRecord(

DID int,

PCardID char(18),

DDay date,

DStatus smallint,

FDay date,

primary key (DID),

foreign key (PCardID) references Passenger(PCardID),

constraint c3 check (Dstatus in (1,2,3))

);

CREATE TABLE TrainContatcor(

CDate date,

CCardID char(18),

DStatus smallint,

PCardID char(18),

primary key (CDate,CCardID,DStatus,PCardID),

foreign key (CCardID) references Passenger(PCardID),

foreign key (PCardID) references Passenger(PCardID));

1. 观察性实验
2. 用户在创建一个用例表的时候要参考Passenger表中的主键，如下图3 . 2 nothis并不是passenger表的主码，运行SQL语句时出错：

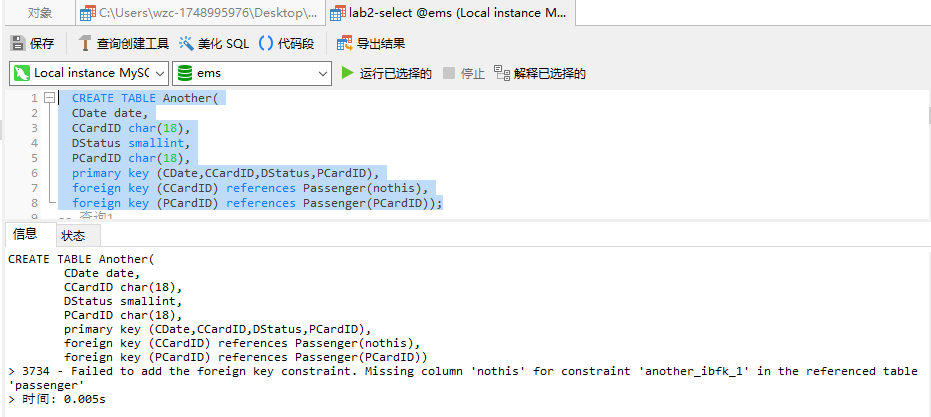


图3 . 2 参照非主码

1. 缺少参照，运行SQL语句仍然出错，如下图3 . 3所示：



图3 . 3缺少参照

1. 数据准备

这里数据的生成使用SQL语句和excel语句随机生成的，由于生成的数据量比较庞大，在这里只展示一部分。

1. taketrainrecord

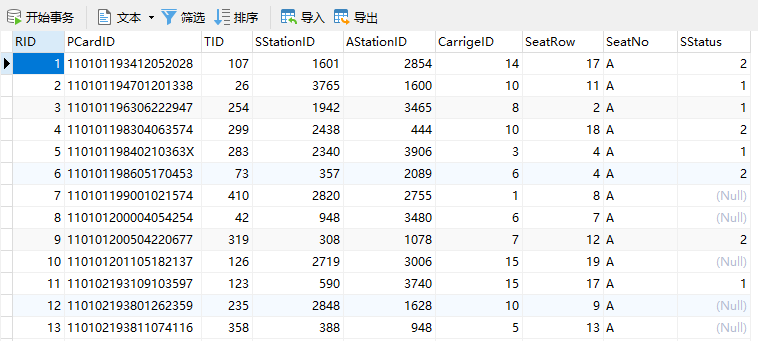


图3 . 4 taketrainrecord数据

1. diagnoserecord

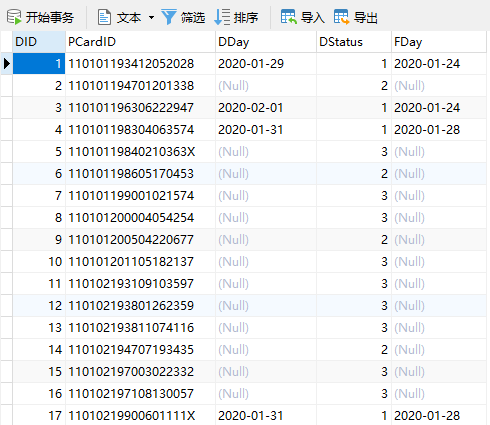


图3 . 5 diagnoserecord数据

1. traincontatcor

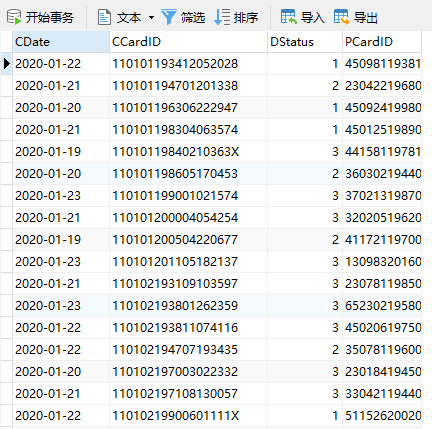


图3 . 6 traincontatcor数据

### 3.2.2数据更新

1. 乘车记录表taketrainrecord增删改
2. 增

INSERT INTO taketrainrecord VALUES

(1001,'130628194909274510',1,1,1,1,1,'F',1)

查看taketrainrecord表：



taketrainrecord增

1. 改

UPDATE taketrainrecord SET SeatNo = 'A' WHERE RID = 1001;

查看taketrainrecord表：

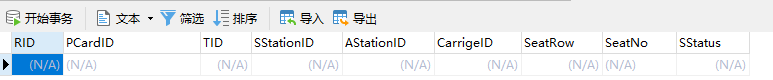


taketrainrecord改

1. 删

DELETE FROM taketrainrecord WHERE RID = 1001;

查看taketrainrecord表：



taketrainrecord删

1. 批处理操作
2. 代码如下：

create table WH\_TakeTrainRecord(

RID int,

PCardID char(18),

TID int,

SStationID int,

AStationID int,

CarrigeID smallint,

SeatRow smallint,

SeatNo char(1),

SStatus int,

primary key (RID),

foreign key (PCardID) references Passenger(PCardID),

foreign key (TID) references Train(TID),

foreign key (SStationID) references Station(SID),

foreign key (AStationID) references Station(SID),

constraint c4 check (SeatNo in ('A','B','C','E','F')),

constraint c5 check (SStatus in (0,1,2))

);

INSERT INTO WH\_TakeTrainRecord

SELECT \*

FROM taketrainrecord

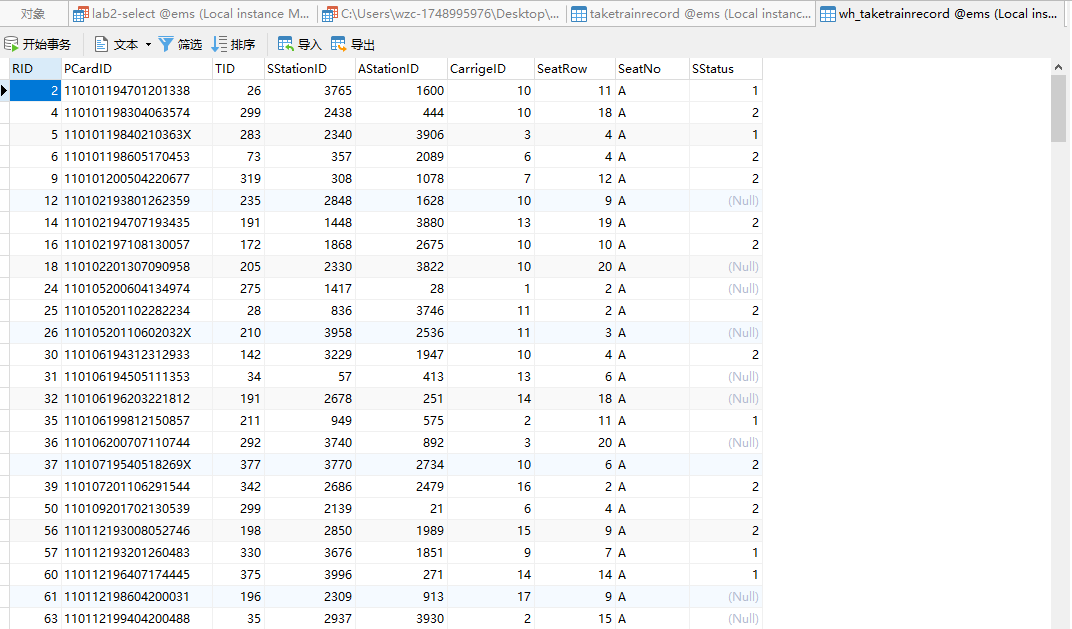
WHERE TID IN

(SELECT TID

FROM train,station

WHERE train.SStationID = station.SID AND station.SName = '武汉')

1. 操作后的WH\_TakeTrainRecord表如下所示：



批处理后的WH\_TakeTrainRecord表

1. 数据导入导出
2. 观察性实验
3. 代码如下：

CREATE TABLE TEST(

data1 int,

data2 text

);

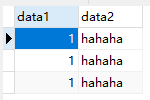
insert into observe values

(1, 'hahaha'),

(1, 'hahaha'),

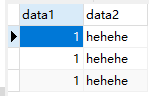
(1, 'hahaha')

1. 更新后的表如下所示：



没有主键的表

1. 对重复元组进行修改或者删除报错，结果如下所示：



重复元组修改成功



重复元组删除成功

1. 创建视图
2. 代码如下：

CREATE VIEW SickView(PCardID,PName,Age,TID,SDate,CarrigeID,SeatRow,SeatNo)

AS

SELECT passenger.PCardID,passenger.PName,passenger.Age,taketrainrecord.TID,SDate,CarrigeID,SeatRow,SeatNo

FROM taketrainrecord,diagnoserecord,passenger,train

WHERE diagnoserecord.PCardID = taketrainrecord.PCardID AND diagnoserecord.PCardID = passenger.PCardID

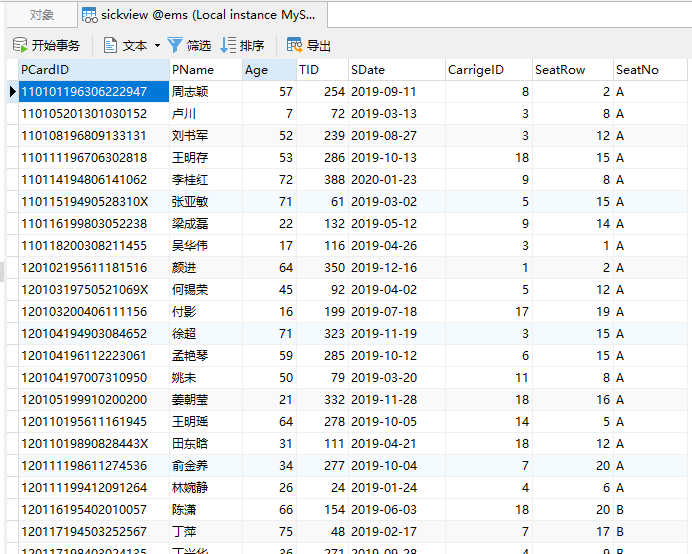
AND taketrainrecord.TID = train.TID

AND diagnoserecord.DStatus = 1

AND taketrainrecord.SStatus = 1

ORDER BY passenger.PCardID,SDate DESC

1. 创建后的sickview视图如下：



sickview视图

1. 触发器实验
2. 代码如下：

CREATE TRIGGER trigger\_1

BEFORE UPDATE ON diagnoserecord

FOR EACH ROW

BEGIN

IF(new.DStatus = 1) AND (old.DStatus <> 1) THEN

INSERT INTO traincontatcor SELECT \* FROM

(SELECT tr1.SDate,t1.PCardID,1 as DS,new.PCardID as PC

FROM taketrainrecord t1,train tr1,

(SELECT t2.TID,t2.CarrigeID,t2.SeatRow

AS T,C,S

FROM taketrainrecord t2,train tr2

WHERE t2.TID = tr2.TID

AND DATEDIFF(new.FDay,tr2.SDate) <= 14

AND t2.PCardID = new.PCardID

AND t2.SStatus = 1) as tmp

WHERE t1.TID = tr1.TID

AND tmp.T = t1.TID

AND tmp.C = t1.CarrigeID

AND tmp.S-t1.SeatRow <= 1

AND tmp.S-t1.SeatRow >= -1

AND t1.SStatus = 1

AND t1.PCardID <> new.PCardID) as tmp2;

UPDATE traincontatcor SET DStatus=1 WHERE PCardID=new.PCardID;

END IF;

END;

CREATE TRIGGER trigger\_2

BEFORE INSERT ON diagnoserecord

FOR EACH ROW

BEGIN

IF(new.DStatus = 1) THEN

INSERT INTO traincontatcor SELECT \* FROM

(SELECT tr1.SDate,t1.PCardID,1 as DS,new.PCardID as PC

FROM taketrainrecord t1,train tr1,

(SELECT t2.TID,t2.CarrigeID,t2.SeatRow

AS T,C,S

FROM taketrainrecord t2,train tr2

WHERE t2.TID = tr2.TID

AND DATEDIFF(new.FDay,tr2.SDate) <= 14

AND t2.PCardID = new.PCardID

AND t2.SStatus = 1) as tmp

WHERE t1.TID = tr1.TID

AND tmp.T = t1.TID

AND tmp.C = t1.CarrigeID

AND tmp.S-t1.SeatRow <= 1

AND tmp.S-t1.SeatRow >= -1

AND t1.SStatus = 1

AND t1.PCardID <> new.PCardID) as tmp2;

UPDATE traincontatcor SET DStatus=1 WHERE PCardID=new.PCardID;

END IF;

END;

1. 测试代码如下：

插入后触发器触发输出提示

### 3.2.3数据查询

1. 查询确诊者“张三”的在发病前14天内的乘车记录。
2. 代码：

SELECT taketrainrecord.RID,taketrainrecord.PCardID,taketrainrecord.TID,taketrainrecord.SStationID,taketrainrecord.AStationID,CarrigeID,SeatRow,SeatNo,SStatus

FROM taketrainrecord,diagnoserecord,train,passenger

WHERE taketrainrecord.PCardID = diagnoserecord.PCardID

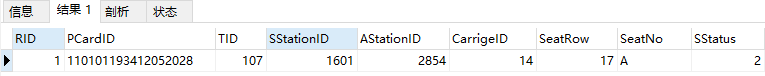
AND taketrainrecord.PCardID = passenger.PCardID

AND taketrainrecord.TID = train.TID

AND passenger.PName = '李玉兰'

AND DATEDIFF(train.SDate,diagnoserecord.FDay) <= 14;

1. 查询结果：



1. 查询所有从城市“武汉”出发的乘客乘列车所到达的城市名。
2. 代码：

SELECT s2.CityName

FROM taketrainrecord,station s1,station s2

WHERE s1.SID = taketrainrecord.SStationID

AND s2.SID = taketrainrecord.AStationID

AND s1.CityName = '武汉'

1. 查询结果：



1. 计算每位新冠患者从发病到确诊的时间间隔（天数）及患者身份信息，并将结果按照发病时间天数的降序排列。
2. 代码：

SELECT DATEDIFF(diagnoserecord.DDay,diagnoserecord.FDay),passenger.PCardID,PName,Age,Sex

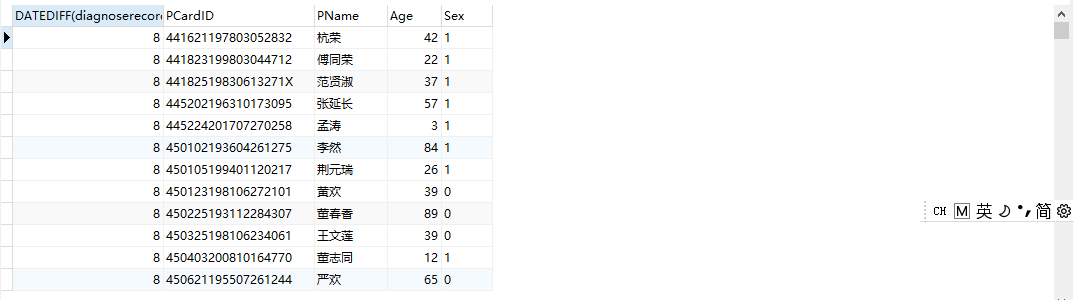
FROM diagnoserecord,passenger

WHERE diagnoserecord.PCardID = passenger.PCardID

AND diagnoserecord.DStatus = 1

ORDER BY DATEDIFF(diagnoserecord.DDay,diagnoserecord.FDay) DESC

1. 查询结果：



1. 查询“2020-01-22”从“武汉”发出的所有列车，由于随机数据的原因，实际查询时这里日期改为“2019-01-22”
2. 代码：

SELECT \*

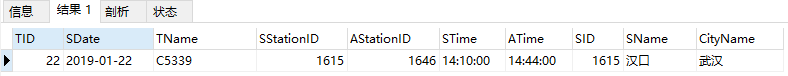
FROM train,station

WHERE train.SStationID = station.SID

AND station.CityName = '武汉'

AND train.SDate = '2020-01-22'

1. 查询结果：



1. 查询“2020-01-22”途经“武汉”的所有列车。
2. 代码：

SELECT \*

FROM train,station,trainpass

WHERE train.TID = trainpass.TID

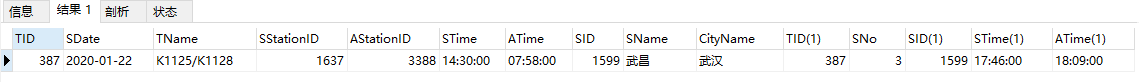
AND trainpass.SID = station.SID

AND train.SDate = '2020-01-22'

AND station.CityName = '武汉'

AND trainpass.SNo <> 1

1. 查询结果：



1. 查询“2020-01-22”从武汉离开的所有乘客的身份证号、所到达的城市、到达日期，由于随机数据的原因，实际查询时这里日期改为“2019-10-15”。
2. 代码：

SELECT PCardID,s2.CityName,SDate

FROM taketrainrecord,station s1,station s2,train

WHERE s1.SID = taketrainrecord.SStationID

AND s2.SID = taketrainrecord.AStationID

AND taketrainrecord.TID = train.TID

AND s1.CityName = '武汉'

AND train.SDate = '2020-01-22'

1. 查询结果



1. 统计“2020-01-22” 从武汉离开的所有乘客所到达的城市及达到各个城市的武汉人员数，由于随机数据的原因，实际查询时这里日期改为“2019-10-15”。
2. 代码：

SELECT s2.CityName,COUNT(\*)

FROM taketrainrecord,station s1,station s2,train

WHERE s1.SID = taketrainrecord.SStationID

AND s2.SID = taketrainrecord.AStationID

AND taketrainrecord.TID = train.TID

AND s1.CityName = '武汉'

AND train.SDate = '2020-01-22'

GROUP BY s2.CityName

1. 查询结果：



1. 查询2020年1月到达武汉的所有人员，由于随机数据的原因，实际查询时这里日期改为2019年1月。
2. 代码：

SELECT PCardID

FROM taketrainrecord,station,train

WHERE taketrainrecord.AStationID = station.SID

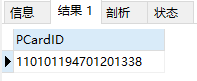
AND taketrainrecord.TID = train.TID

AND station.CityName = '武汉'

AND train.SDate >= '2020-01-01'

AND train.SDate <= '2020-01-31'

1. 查询结果：



1. 查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）。
2. 代码：

SELECT DISTINCT passenger.PCardID,passenger.PName,passenger.Age,passenger.Sex,train.TName

FROM train,station,trainpass,taketrainrecord,passenger

WHERE taketrainrecord.TID = train.TID

AND taketrainrecord.PCardID = passenger.PCardID

AND train.TID = trainpass.TID

AND trainpass.SID = station.SID

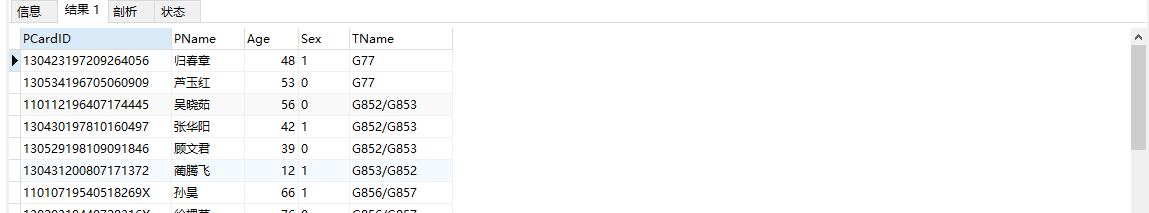
AND train.SDate >= '2020-01-01'

AND train.SDate <= '2020-01-31'

AND station.CityName = '武汉'

AND taketrainrecord.PCardID NOT LIKE '420%'

1. 查询结果：



1. 统计“2020-01-22”乘坐过‘G007’号列车的新冠患者在火车上的密切接触乘客人数（每位新冠患者的同车厢人员都算同车密切接触），由于随机数据的原因，实际查询时这里可把日期和火车号限制去掉。
2. 代码：

SELECT taketrainrecord.PCardID,COUNT(\*)

FROM train,taketrainrecord

WHERE train.TID = taketrainrecord.TID

AND (taketrainrecord.TID,taketrainrecord.CarrigeID) IN

(SELECT DISTINCT sickview.TID,sickview.CarrigeID

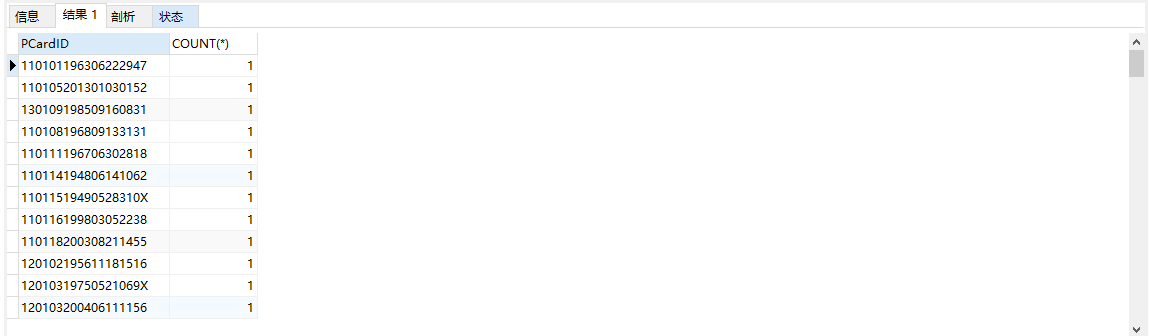
FROM sickview)

AND train.TName = '%G007%'

AND train.SDate = '2020-01-22'

GROUP BY taketrainrecord.PCardID

1. 查询结果：



1. 查询一趟列车的一节车厢中有3人及以上乘客被确认患上新冠的列车名、出发日期，车厢号，由于随机数据的原因，在实际查询时将3人改为1人。
2. 代码：

SELECT TName,s1.SDate,s1.CarrigeID

FROM sickview s1,train

WHERE 3<=(SELECT COUNT(\*)

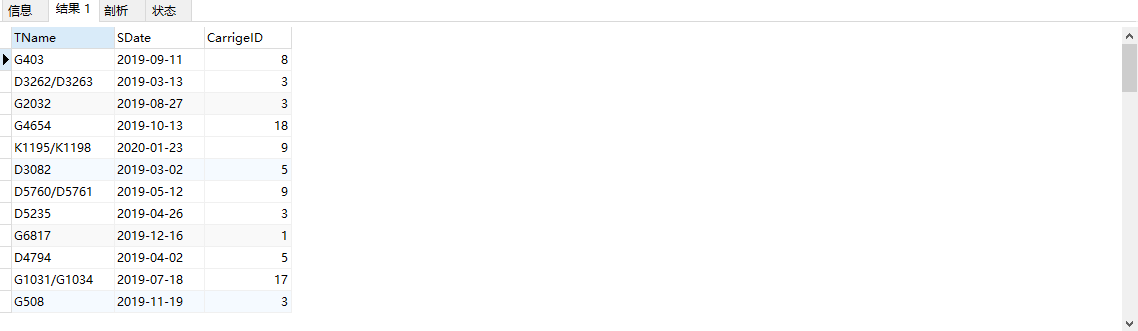
FROM sickview s2

WHERE s1.TID = s2.TID

AND s1.CarrigeID = s2.CarrigeID)

AND s1.TID = train.TID

1. 查询结果：



1. 查询没有感染任何周边乘客的新冠乘客的身份证号、姓名、乘车日期。
2. 代码：

SELECT\*

FROM sickview

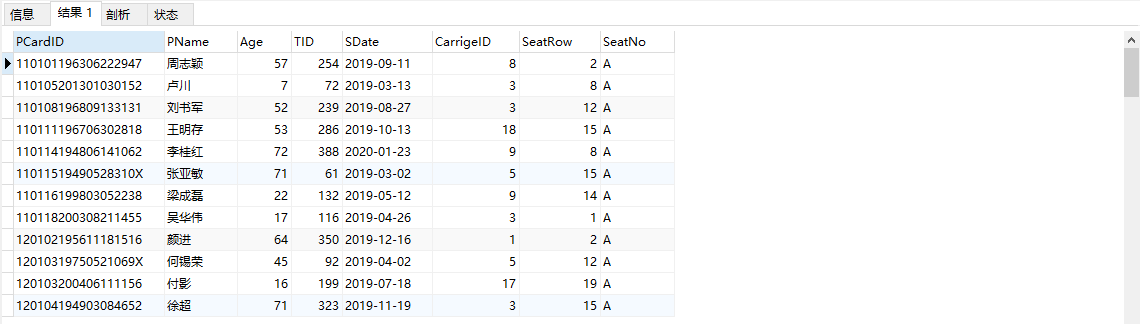
WHERE sickview.PCardID NOT IN

(SELECT traincontatcor.PCardID

FROM traincontatcor

WHERE traincontatcor.DStatus = 1)

1. 查询结果：



1. 查询到达 “北京”、或“上海”，或“广州”（即终点站）的列车名，要求where子句中除了连接条件只能有一个条件表达式。
2. 代码：

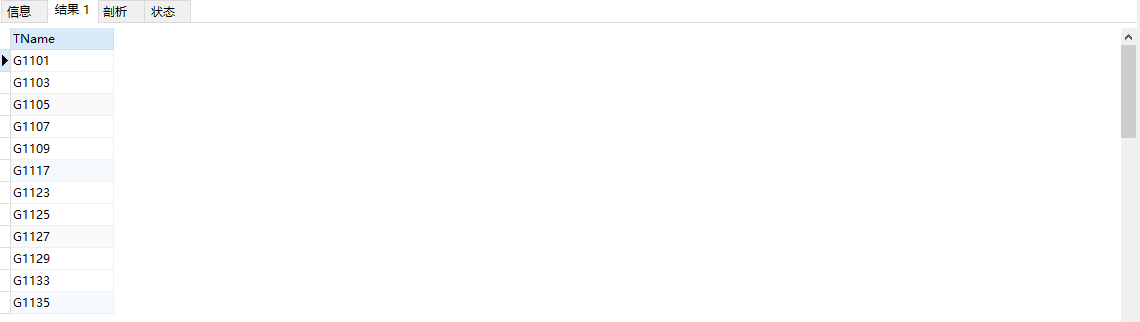
SELECT train.TName

FROM train,station

WHERE train.AStationID = station.SID

AND station.CityName IN ('北京','上海','广州')

1. 查询结果：



1. 查询“2020-01-22”从“武汉站”出发，然后当天换乘另一趟车的乘客身份证号和首乘车次号，结果按照首乘车次号降序排列，同车次则按照乘客身份证号升序排列，由于随机数据的原因，在实际查询时将日期更换为“2019-10-15”
2. 代码：

SELECT take\_1.PCardID,train\_1.TName

FROM taketrainrecord take\_1,train train\_1,station s1,trainpass pass\_1

WHERE take\_1.TID = pass\_1.TID

AND pass\_1.SID = s1.SID

AND s1.SName = '武汉'

AND train\_1.SDate = '2020-01-22'

AND EXISTS(

SELECT \*

FROM taketrainrecord take\_2,train train\_2,trainpass pass\_2

WHERE take\_2.TID = train\_2.TID

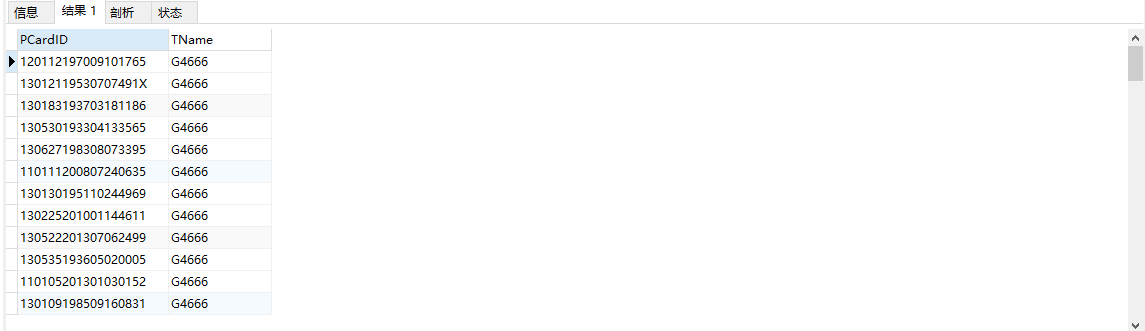
AND take\_2.SStationID = pass\_2.SID

AND train\_2.SDate = '2020-01-22'

AND pass\_2.STime > pass\_1.STime

)

1. 查询结果：



1. 查询所有新冠患者的身份证号，姓名及其2020年以来所乘坐过的列车名、发车日期，要求即使该患者未乘坐过任何列车也要列出来。
2. 代码：

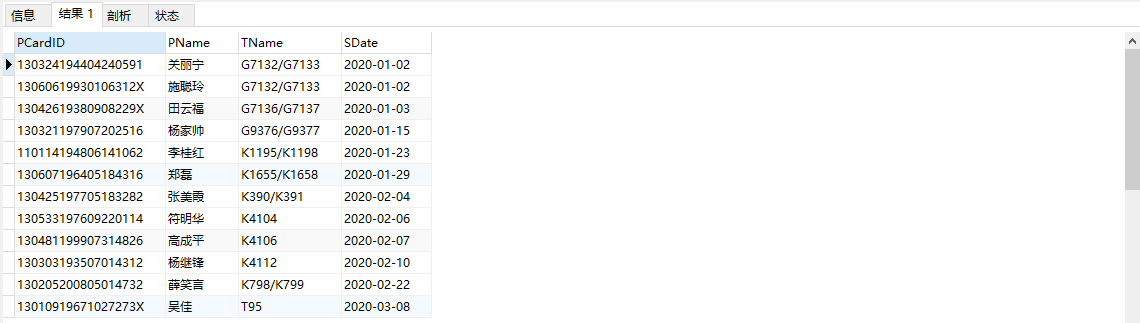
SELECT sickview.PCardID,sickview.PName,TName,sickview.SDate

FROM sickview,train

WHERE sickview.TID = train.TID

AND sickview.SDate LIKE '2020%'

1. 查询结果：



1. 查询所有发病日期相同而且确诊日期相同的病患统计信息，包括：发病日期、确诊日期和患者人数，结果按照发病日期降序排列的前提下再按照确诊日期降序排列。
2. 代码：

SELECT d1.FDay,d1.DDay,COUNT(\*)

FROM diagnoserecord d1

WHERE d1.FDay IN

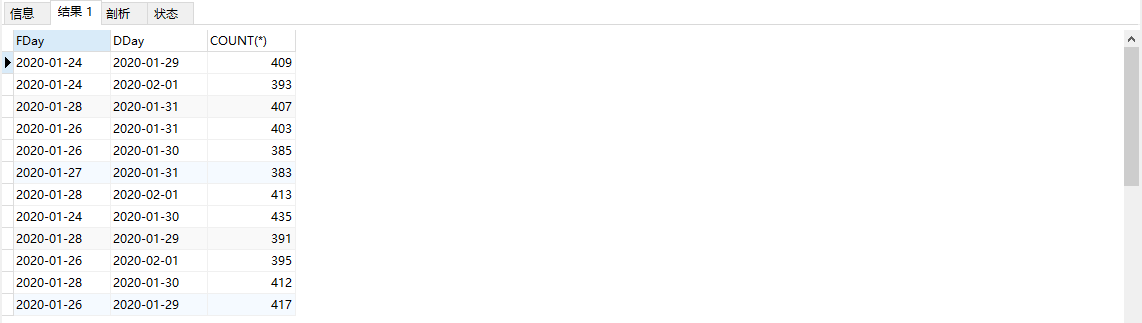
(SELECT d2.FDay

FROM diagnoserecord d2

WHERE d1.DDay = d2.DDay)

GROUP BY d1.FDay,d1.DDay

1. 查询结果：



## 3.3任务总结

1. 学会了如何使用Mysql创建数据库、数据表和一些基本的操作。
2. 学会了如何创建触发器来保证数据表之间的约束完整性；
3. 基本掌握了在数据库中检索数据的方法。

# 4 综合实践任务

## 4.1系统设计目标

随着经济的发展，人们的生活越来越好。日常用品越来越多，超市的规模越来越大，超市收银员的工作量也越来越大。为了适应超市规模的不断变化，减轻收银员的工作负担，根据超市的实际需要设计了一个具有开放体系结构、易扩充的、易维护的、具有良好人机交互界面的超市收银系统，为超市的决策者和管理者提供超市的各种数据信息、方便的查询和高效便捷的管理。该超市收银系统包括前台操作和后台数据库，后台数据库包括销售记录、消费记录、会员信息、收银员信息、商品信息，前台操作是收银、办卡。

## 4.2 需求分析

1. 假设一家小型超市的收银台（前台）要完成下列日常工作：

* 收银：收银员输入顾客的会员卡卡号（若有卡）、所购商品的货号等信息，系统根据这些信息获取相应的价格信息并计算应收取的总金额。完成收银后，记录交易信息，修改有关种类商品的剩余量以及该持卡顾客的消费情况。
* 发卡：顾客可交纳一定的费用（如50元）办理一张会员卡，以后在该商场购物可凭卡享受9折优惠。如果一个未持卡顾客一次购物满1000元，可为其免费发放一张会员卡，每张卡的优惠期为一年，一年内消费达到一定金额的可继续享受下一年的优惠。
* 款项盘存：收银员下班或交接班前对本收银台中本班次收取的款额进行盘存，明确责任。

此外，还应提供下列后台功能：

* 商品信息的录入、修改、删除和查询等。
* 收银员身份及口令管理。

设计一个C/S模式的系统实现上述功能。

1. 系统数据流图如图4 . 1所示：

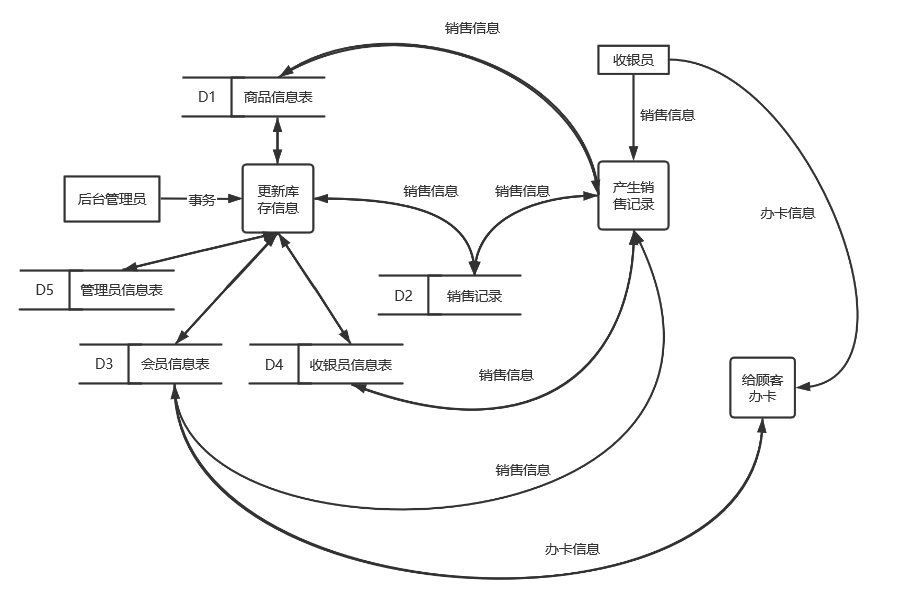


图4 . 1 超市收银系统数据流图

1. 超市收银系统数据字典如图4 . 2所示：

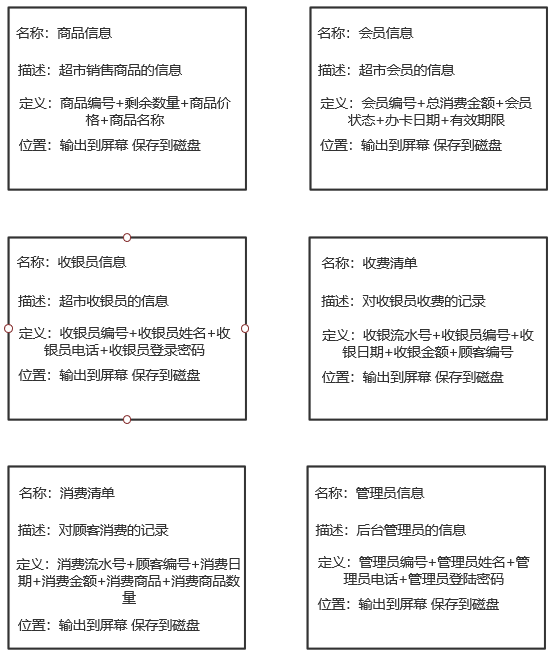


图4 . 2 超市收银系统数据字典

## 4.3 总体设计

1. 总体结构设计如下图4 . 3所示：

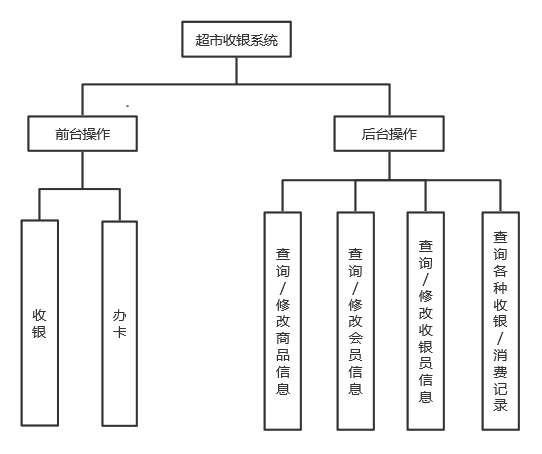


图4 . 3 总体结构设计

1. 系统流程图如下图4 . 4所示：

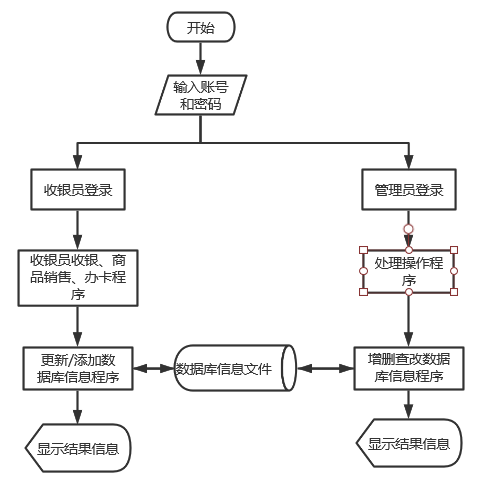


图4 . 4 超市收银系统流程图

## 4.4 数据库设计

1. 整体ER图如下图4 . 5所示：

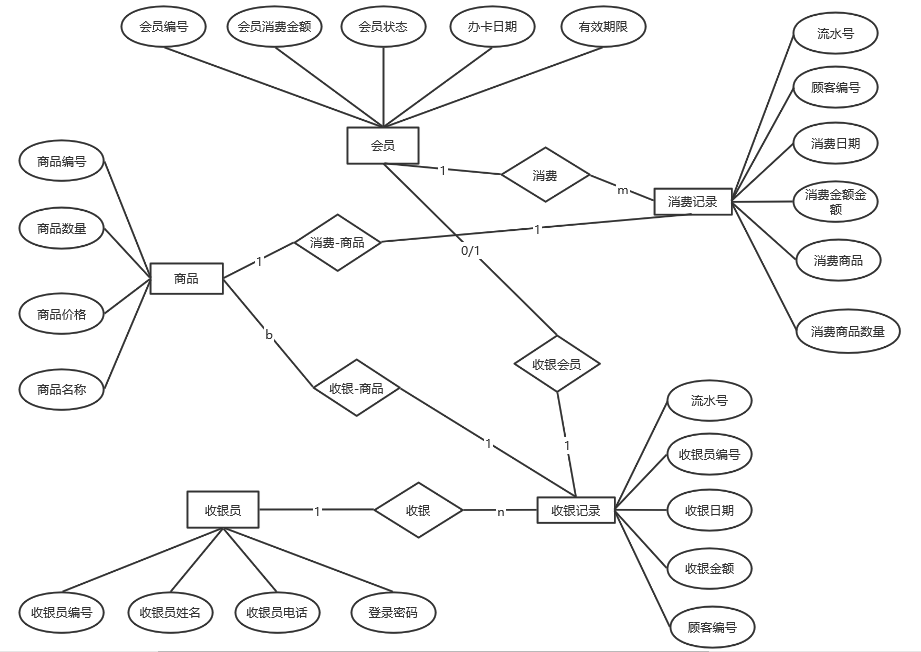


图4 . 5 超市收银系统ER图

1. 数据库逻辑结构设计
2. 商品信息表：商品编号、商品数量、商品价格、商品名称
3. 收银员信息表：收银员编号、收银员姓名、收银员电话、登录密码
4. 会员信息表：会员编号、会员消费金额、会员状态、办卡日期、有效期限
5. 收银记录表：收银流水号、收银员编号、收银日期、收银金额、顾客编号
6. 消费记录表：消费流水号、顾客编号、消费日期、消费金额、消费的商品编号、消费的商品数量
7. 管理员信息表：管理员编号、管理员姓名、管理员电话、登录密码
8. 数据库物理结构设计

表4 . 1 商品信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键/外键 | 是否可为空 | 索引 | 备注 |
| productcode | char(20) | 主键 | 否 | 是 |  |
| remainingnumber | int |  | 是 |  |  |
| price | double |  | 是 |  |  |
| ganme | char(20) |  | 是 |  |  |

表4 . 2 收银员信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键/外键 | 是否可为空 | 索引 | 备注 |
| jobid | char(10) | 主键 | 否 | 是 | 登陆账号 |
| ename | char(10) |  | 是 | 否 |  |
| phone | char(20) |  | 是 | 否 |  |
| passwd | char(6) |  | 是 | 否 |  |

表4 . 3 会员信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键/外键 | 是否可为空 | 索引 | 备注 |
| vipid | char(10) | 主键 | 否 | 是 |  |
| expense | double |  | 是 | 否 |  |
| state | smallint |  | 是 | 否 | 0：永久  1：有期限 |
| pd | date |  | 是 | 否 | 办卡日期 |
| exp | date |  | 是 | 否 | 截止日期 |

表4 . 4 收银记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键/外键 | 是否可为空 | 索引 | 备注 |
| id | int | 主键 | 否 | 是 |  |
| jobid | char(10) | 外键 | 否 | 否 |  |
| cdate | date |  | 是 | 否 | 收银日期 |
| chargeamount | double |  | 是 | 否 | 收银金额 |
| vipid | int | 外键 | 是 | 否 |  |

表4 . 5 消费记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键/外键 | 是否可为空 | 索引 | 备注 |
| id | int | 主键 | 否 | 是 |  |
| vipid | char(10) | 外键 | 是 | 否 |  |
| edate | date |  | 是 | 否 |  |
| monetary | double |  | 是 | 否 |  |
| goodid | char(20) | 外键 | 是 | 否 |  |
| goodnumber | int |  | 是 | 否 |  |

表4 . 6 管理员信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键/外键 | 是否可为空 | 索引 | 备注 |
| superid | char(10) | 主键 | 否 | 否 | 登录账号 |
| sname | char(10) |  | 是 | 否 |  |
| phone | char(20) |  | 是 | 否 |  |
| passwd | char(6) |  | 是 | 否 |  |

## 4.5 详细设计与实现

本次实验设计的是超市收银系统，不需要通过套接字便可实现信息的增删查改。如果要实现真正意义上的C/S模式，可以配置mysql接受远程连接，将客户端在远程连接数据库即可。实验所有的功能都是在Qt中基于对本地数据库的操作完成，功能的实现主要分为前台功能的实现和后台功能的实现。两个功能之间有一定程度的代码重复利用。

1. 相同功能部分

管理员和收银员的进行操作时都需要连接数据库，因此连接数据库具有相同的实现。这里将连接数据的操作封装在DBHelper类当中，封装了连接数据库和断开数据库连接的功能。DBHelper中只允许存在一个DBHelper类对象的实例。DBHelper声明如下所示：

|  |
| --- |
| class DBHelper  {  public :  static DBHelper \*getInstance();  void connectDatabase();  void disconnectDatabase();  private:  static DBHelper \*instance;  DBHelper();  QSqlDatabase db;  }; |

DBHelper的实现如下所示：

|  |
| --- |
| DBHelper\* DBHelper::instance = NULL;  DBHelper::DBHelper()  {  db = QSqlDatabase::addDatabase("QMYSQL");  db.setHostName("localhost");  db.setDatabaseName("shopcash");  db.setPort(3306);  db.setUserName("root");  db.setPassword("3714231999");  }  void DBHelper::connectDatabase()  {  bool ret = db.open();  if(ret){  qDebug() << "open db success!";  }else{  qDebug() << "open db failed!";  }  }  void DBHelper::disconnectDatabase()  {  db.close();  db.removeDatabase("shopcash");  delete instance;  instance = NULL;  }  DBHelper\* DBHelper::getInstance()  {  if(instance == NULL){  instance = new DBHelper();  return instance;  }else{  return instance;  }  } |

管理员和收银员登录程序具有类似的算法实现。在这里使用UserDao类封装了登录时判断账号密码是否正确的接口，并且使用UserDaoImp类继承UserDao并且重写UserDao类中的纯虚函数，使得程序具有更好的可读性以及模块化。UserDao类的声明和定义比较简单，UserDaoImp中，selectUserById(QString jobid)和selectSuperById(QString superid)函数分别负责通过账号来查询正确的登录密码，用作后面的判断。UserDaoImp类的实现如下所示：

|  |
| --- |
| UserDaoImp::UserDaoImp()  {}  //根据收银员id在数据库中查询  QSqlQuery UserDaoImp::selectUserById(QString jobid)  {  DBHelper \*helper = DBHelper::getInstance();  QSqlQuery query;  helper->connectDatabase();  //准备SQL语句  query.prepare("select \* from employee where jobid = :jobid");  query.bindValue(":jobid",jobid);  query.exec();  helper->disconnectDatabase();  return query;  }  QSqlQuery UserDaoImp::selectSuperById(QString superid){  DBHelper \*helper = DBHelper::getInstance();  QSqlQuery query;  helper->connectDatabase();  //准备SQL语句  query.prepare("select \* from super where superid = :superid");  query.bindValue(":superid",superid);  query.exec();  helper->disconnectDatabase();  return query;  } |

mainwindow.cpp主要负责登录界面跳转的逻辑，home.cpp负责实现收银员界面的功能，superhome.cpp负责实现后台管理员界面的功能。登录功能的ui界面如下图4 . 1所示：



图4 . 6登录功能的ui界面

登录功能的ui界面中，收银员登入按钮和管理员登陆按钮分别处理不同用户的登陆，用户输入账号和密码后，通过调用DBHelper中的函数连接数据库，并且通过实例化UserDaoImp，调用其中的函数实现账号密码的查询，将获得的密码与输入框的密码相比较，判断密码是否正确，以此来决定是否进行界面的跳转。不同的页面之间需要相同的参数，可以通过信号槽将主界面中的收银员id和管理员id发送到分支界面，以此来实现参数的传递。当从分支界面返回到主界面时，分支界面发送信号，主界面中的槽函数接受信号，进行页面的切换，主界面与分支界面切换的逻辑都是在mainwindow.cpp中完成。收银员界面的展示以及退出和管理员类似，这里以收银员为例展示一下实现代码，收银员界面的信号槽函数在mainwindow.cpp中绑定为

|  |
| --- |
| connect(this,SIGNAL(userLogin(const String&,const QString&)),home,SLOT(onuserLogin(const QString &,const QString &)))；connect(home,SIGNAL(userLogout()),this,SLOT(onuserLogout())); |

转到收银员界面和退出收银员界面的逻辑代码为

|  |
| --- |
| DBHelper \*helper = DBHelper::getInstance();  helper->connectDatabase();  emit userLogin(username,passwd); |
| mainwindow.cpp：  void MainWindow::onuserLogout(){  home->hide();  this->show();  }  home.cpp：  void Home::on\_exit\_Button\_clicked()  {  DBHelper \*helper = DBHelper::getInstance();  helper->disconnectDatabase();  emit userLogout();  } |

1. 前台功能

前台的ui界面如下图4 . 7所示：

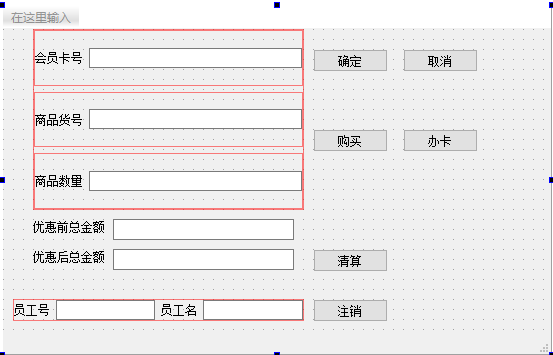


图4 . 7前台ui界面

|  |
| --- |
| class Home : public QMainWindow  {  Q\_OBJECT  public:  explicit Home(QWidget \*parent = nullptr);  ~Home();  private:  Ui::Home \*ui;  QString jobid;//保存收银员的ID  QString vipid = "";//保存当前读取的会员ID  double money = 0;//保存会员最新的消费金额总数  double discount = 1;//当前顾客享受的折扣  double money\_this = 0;//本次消费的总金额  int state = 3;//会员的状态  QDate exp;//会员卡的有效截止日期  int canrenew = 500;//消费满canrenew后有效期延长一年  public slots:  void onuserLogin(const QString &username,const QString &passwd);  signals:  void userLogout();  private slots:  void on\_exit\_Button\_clicked();//注销按钮  void on\_ok\_vipid\_button\_clicked();//确定会员号按钮  void on\_notok\_vipid\_button\_clicked();//取消会员号按钮  void on\_wantbut\_button\_clicked();//购买按钮，单个商品购买  void on\_finish\_button\_clicked();//清算按钮，本次收银结束  void on\_becomevip\_button\_clicked();//办卡按钮  }; |

各个按钮的槽函数实现都比较简单，读取输入框中的内容，并且从数据库中查询数据，然后对数据库中的数据进行增删查改。确定和取消会员按钮需要在购物前输入，才能享受折扣。其中比较复杂的实现时清算按钮，本次收银结束后，需要根据当前顾客的会员状态以及本次消费总金额和以往消费的总金额进行不同的处理。处理过程如下：

|  |
| --- |
| if(vipid != "" && state == 1){//本次消费的会员不是永久会员  QDate exp\_last = exp.addYears(-1);  exp\_last = exp\_last.addDays(1);//计算截至日期的前一年时间内的总消费  query.clear();  query.prepare("select sum(monetary) from expenserecord where vipid = :a and edate >= :b and edate <= :c");//计算一年内的消费总金额  query.bindValue(":a",vipid);  query.bindValue(":b",exp\_last);  query.bindValue(":c",exp);  query.exec();  query.next();  qDebug() << "消费总额!";  qDebug() << query.value(1).toInt();  if(query.value(0).toInt() >= canrenew){//消费金额满足要求  //将有效截止日期延期一年  exp = exp.addYears(1);  query.clear();  query.prepare("update vip set exp = :a where vipid = :b");  query.bindValue(":a",exp);  query.bindValue(":b",vipid);  query.exec();}  }//并且根据本次的消费金额来决定是否送卡  else if(vipid == ""){  if(current\_money >= 1000){  QMessageBox message(QMessageBox::Warning,"提示","是否送卡",QMessageBox::Yes|QMessageBox::No,NULL);  if (message.exec()==QMessageBox::Yes)  {  query.clear();  //读取最大的会员编号，将其加一作为新会员的编号  query.prepare("select \* from vip order by vipid DESC limit 1");  query.exec();  int last\_vipid = 0;  if(query.size() !=0){  query.next();  last\_vipid = query.value(0).toInt();}  last\_vipid++;  query.clear();  //插入新的会员数据，并且本次的消费不属于vip身份消费  query.prepare("insert into vip (vipid,expense,state,pd,exp) values (:a,:b,:c,:d,:e)");  query.bindValue(":a",last\_vipid);  query.bindValue(":b",0);  query.bindValue(":c",1);  query.bindValue(":d",QDate::currentDate());  query.bindValue(":e",QDate::currentDate().addYears(1));  query.exec();  QString mess = "卡号为"+QString::number(last\_vipid);  QMessageBox::critical(this,"办卡成功",mess);}  else{}  }  } |

其它功能的实现比较简单，这里不再赘述。

1. 后台功能

后台的ui界面主要有四个，分别如下：



图4 . 8查询记录界面



图4 . 9修改商品信息界面



图4 . 10修改收银员界面



图4 . 11修改会员界面

后台功能的实现与前台功能的实现具有类似的原理，涉及对数据库的增删查改，这里不再赘述。

1. 表的创建

建表语句如下：

|  |
| --- |
| -- 下面是会员信息  -- pd：办卡日期  -- state=0：该会员卡长期有效  -- state=1：该会员卡当前有效，截止日期为exp  -- state=2：该会员卡无效  CREATE TABLE vip(  vipid CHAR(10),  expense DOUBLE,  state SMALLINT,  pd date,  exp date,  PRIMARY KEY (vipid));  -- 下面是消费记录  -- id：消费流水号  -- edate：消费日期  -- monetary：消费金额  CREATE TABLE expenserecord(  id INT,  vipid CHAR(10),  edate date,  monetary DOUBLE,  goodid CHAR(20),  goodnumber INT,  PRIMARY KEY (id),  FOREIGN KEY (vipid) REFERENCES vip(vipid));  ALTER TABLE expenserecord  MODIFY COLUMN id INT NOT NULL AUTO\_INCREMENT FIRST;  -- 下面是收银员信息  CREATE TABLE employee(  jobid CHAR(10),  ename CHAR(10),  phone CHAR(20),  passwd CHAR(6),  PRIMARY KEY (jobid));  -- 下面是款项盘存  -- id：收费流水号  -- jobid：工号  -- cdate：收费日期  -- chargeamount：收费金额  CREATE TABLE chargerecord(  id INT,  jobid CHAR(10) NOT NULL,  cdate date,  chargeamount DOUBLE,  vipid INT,  PRIMARY KEY (id),  FOREIGN KEY (jobid) REFERENCES employee(jobid));  ALTER TABLE chargerecord  MODIFY COLUMN id INT NOT NULL AUTO\_INCREMENT FIRST;  -- 下面是商品信息  CREATE TABLE goods(  productcode CHAR(20),  remainingnumber INT,  price DOUBLE,  ganme CHAR(20),  PRIMARY KEY (productcode));  -- 管理员表  CREATE TABLE super(  superid CHAR(10),  sname CHAR(10),  phone CHAR(20),  passwd CHAR(6),  PRIMARY KEY (superid)); |

## 4.6 系统测试

一开始所有表初始状态除货物表和会员表外其它的表均为空，货物表和会员表的初始状态如下：



图4 . 12 货物表初始状态

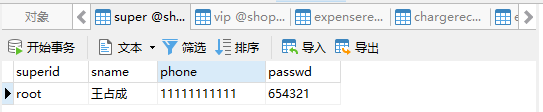


图4 . 13 会员表初始状态

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试界面 | 测试数据/测试操作 | 结果与预期是否一致 |
| 1 | 主界面 | 登录管理员界面，登录收银员界面 | 是 |
| 2 | 收银员 | 给新会员办卡 | 是 |
| 3 | 收银员 | 非会员购买商品共1000元(拒绝送卡) | 是 |
| 4 | 收银员 | 非会员购买商品共1000元(接受送卡) | 是 |
| 5 | 收银员 | 上述新送卡的会员消费700元(打折前) | 是 |
| 6 | 收银员 | 注销 | 是 |
| 7 | 管理员 | 查看2020.5.26收银员root收银记录 | 是 |
| 8 | 管理员 | 查看2020.5.26所有会员的消费记录 | 是 |
| 9 | 管理员 | 新增货物小西红柿(00003,2000,5) | 是 |
| 10 | 管理员 | 查询所有收银员信息 | 是 |
| 11 | 管理员 | 修改会员2的状态为0 | 是 |

测试一后弹出的界面：

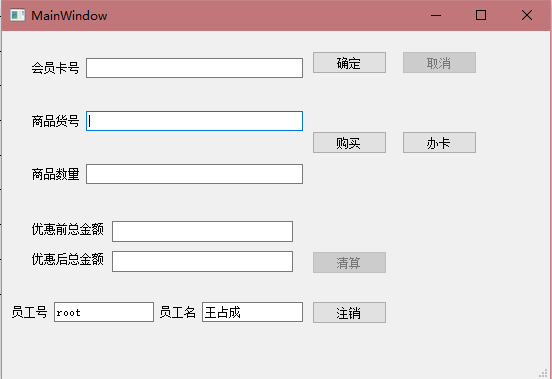


图4 . 14 收银员登陆成功

测试二后数据库中vip表：



图4 . 15

测试四后数据库中expenserecord表、chargerecord表、vip表如下：

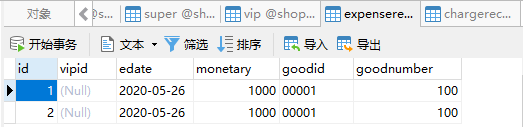


图4 . 16

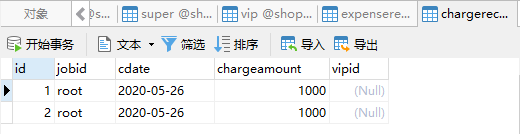


图4 . 17

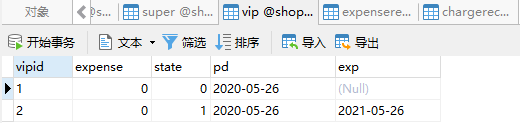


图4 . 18

测试五后数据库中expenserecord表、chargerecord表、vip表如下：

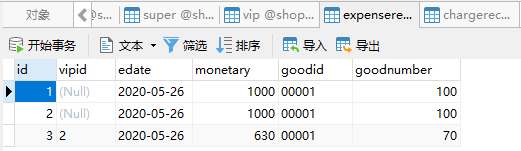


图4 . 19

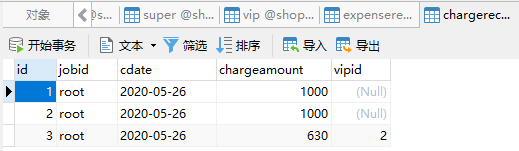


图4 . 20

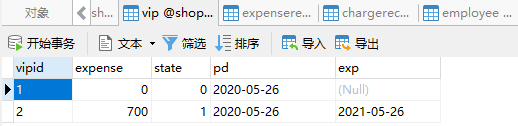


图4 . 21

测试七查询结果如下：

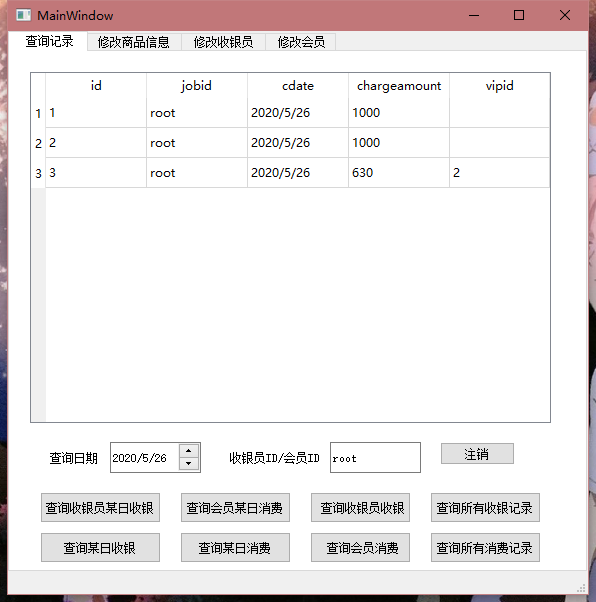


图4 . 22

测试八查询结果如下：

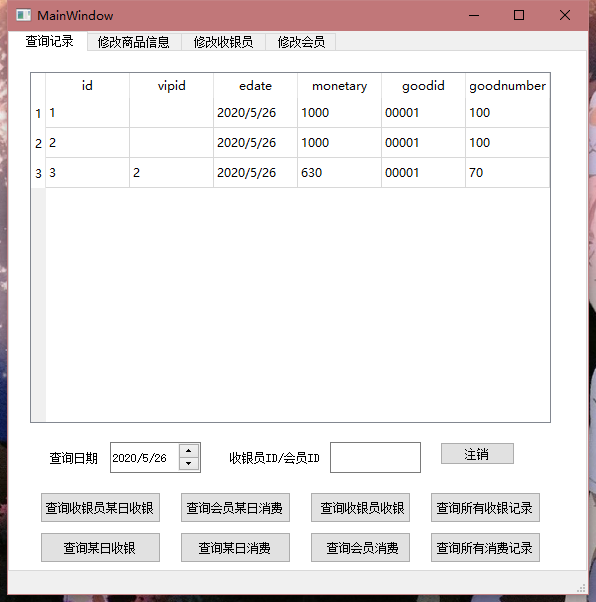


图4 . 23

测试九数据库goods表如下：



图4 . 24

测试十查询结果如下：



图4 . 25

测试十一数据库中vip表如下：

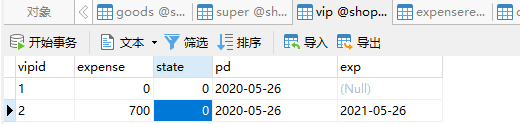


图4 . 26

## 4.7 系统设计与实现总结

在这次课程设计中，我对于一个C/S模式带数据库的小型超市收银系统进行了完整的设计。完成的主要工作如下：

* 完成了对于数据库结构以及软件的需求分析
* 完成了数据库的表格创建，数据的生成以及导入
* 设计并实现了模块化客户端程序
* 进行了较为完整的功能、性能测试

# 4 课程总结

这次课程设计让我熟悉了具体的数据库应用系统设计，了解了C/S系统如何进行开发，对于QT的使用更加熟练，对于小项目编写的更加模块化。譬如对于QT中界面的切换有个基本了解，对QT调用数据库有了基本的掌握。

这次实验也锻炼了我设计复杂与实现复杂系统的能力，从对一个程序的需求分析开始，一直到程序的设计完毕，本次实验让我更加熟悉了这个过程，现在已经能基本掌握这个流程，通过QT设计一个简单的程序。

# 附录