

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 实验一 |
| 姓 名： | 梁一飞 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1706 |
| 学 号： | U201714762 |
| 指导教师： | 丁晓峰 |

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

2020 年 5月 9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](#_Toc44021288)

[2 实验任务一 软件功能学习部分 1](#_Toc44021289)

[2.1 任务要求 1](#_Toc44021290)

[2.2 完成过程 1](#_Toc44021291)

[2.3任务总结 4](#_Toc44021292)

[3 实验任务二 Sql练习部分 4](#_Toc44021293)

[3.1.1 建表 4](#_Toc44021294)

[3.1.2 完成过程 5](#_Toc44021295)

[3.1.3任务总结 7](#_Toc44021296)

[3.2.1 数据更新 7](#_Toc44021297)

[3.2.2 完成过程 8](#_Toc44021298)

[3.2.3 任务总结 9](#_Toc44021299)

[3.3.1 查询 10](#_Toc44021300)

[3.3.2 完成过程 10](#_Toc44021301)

[3.3.3 任务总结 16](#_Toc44021302)

[附录 16](#_Toc44021303)

[4 综合实践任务 23](#_Toc44021304)

[4.1系统设计目标 23](#_Toc44021305)

[4.2 需求分析 23](#_Toc44021306)

[4.3 总体设计 25](#_Toc44021307)

[4.4 数据库设计 25](#_Toc44021308)

[4.5 详细设计与实现 30](#_Toc44021309)

[4.6 系统测试 30](#_Toc44021310)

[4.7 系统设计与实现总结 33](#_Toc44021311)

[4 课程总结 33](#_Toc44021312)

[附录 34](#_Toc44021313)

# 1 课程任务概述

一.实验要求

1.交互式实验一

软件的基本功能学习及其操作实验报告

2.交互式实验二

交互式使用SQL语言及其实验报告

二.实验目的

1.通过上机实践，熟悉一种大型数据库管理系统，了解DBMS的体系结构。

2.熟练掌握SQL的数据定义、数据操纵和数据控制语言的运用。

# 实验任务一 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 备份

1.软件功能备份

对需要备份的数据库点击右键，选择任务->备份,选择备份地址，点击确定，如图1所示：

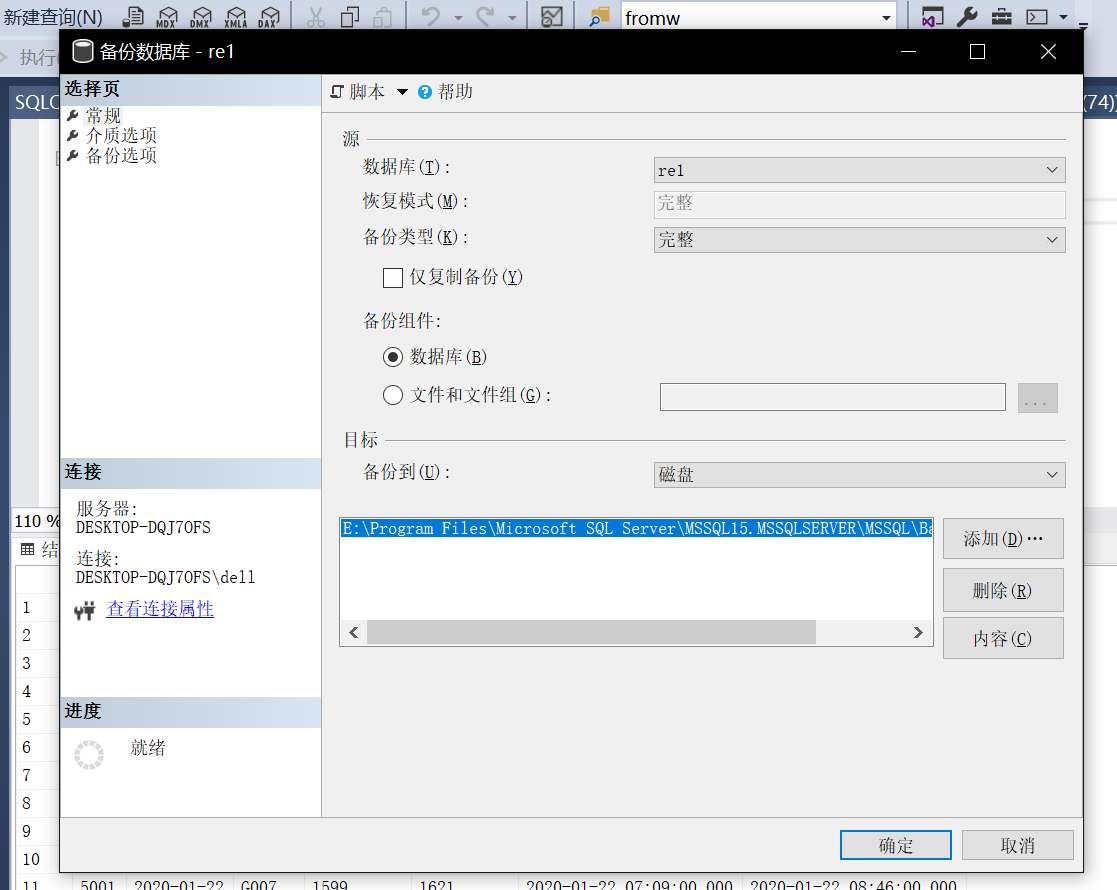


图1.软件功能备份

2.脱机文件备份

输入如下命令：

select file\_name,bytes from dba\_data\_files;(所有数据文件)

select member from v$logfile;(所有日志文件)

select name from v$controlfile;(所有控制文件)

获取所有文件位置，然后关闭数据库，复制所有文件到储存备份的位置即可。

### 2.2.2增加用户并配置权限

选择数据库->安全性->新建->用户，如图2所示：

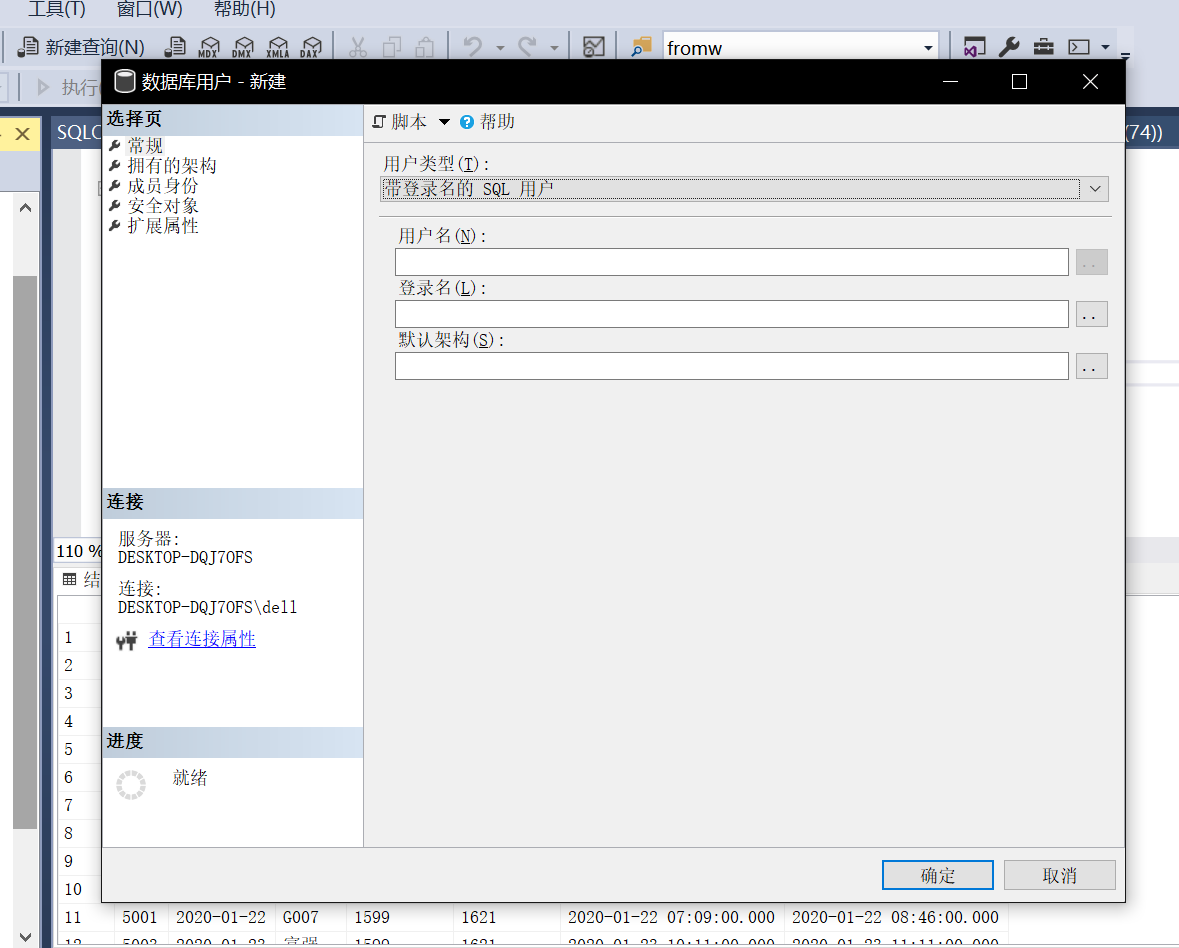


图2.增加用户

配置完成后，选择数据库->属性->权限，配置用户权限，如图3所示：

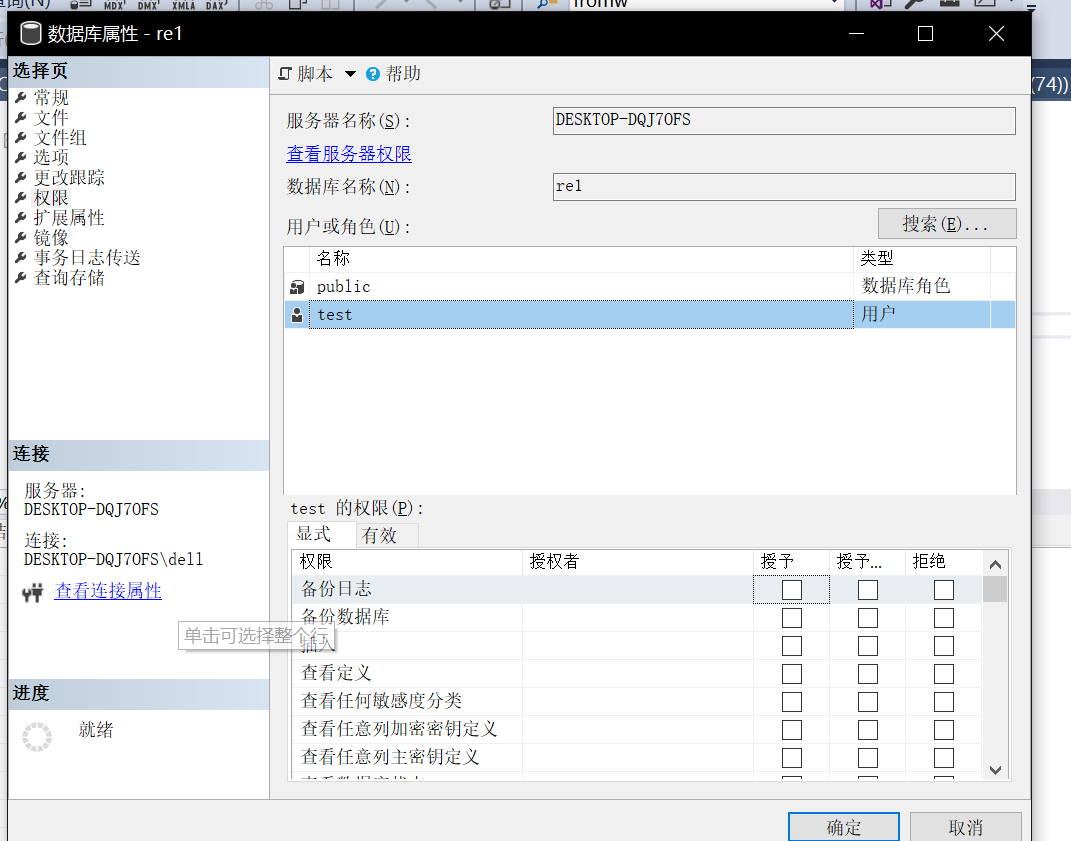


图3.配置用户权限

## 2.3任务总结

任务一较简单，对照教程备份和增加用户配置权限，未遇到问题。

# 3 实验任务二 Sql练习部分

## 3.1.1 建表

**1）创建下列跟“疫期乘坐列车”相关的关系，包括主码和外码的说明**

**车站表**【车站编号，车站名，所属城市】

Station (SID int, SName char(20), CityName char(20))

其中，主码为车站编号。

**车次表**【列车流水号，发车日期，列车名称，起点站编号，终点站编号，开出时刻，终点时刻】

Train (TID int, SDate date, TName char(20), SStationID int, AStationID int, STime datetime, ATime datetime)

其中，TID为主码，(列车名称，发车日期)为候选码；SStationID和AStationID都来源于车站表的SID

**车程表**【列车流水号，车站序号，车站编号，到达时刻，离开时刻】

TrainPass (TID int, SNo smallint, SID int, STime datetime, ATime datetime)

其中，主码为(TID,SNo)。SID来源于车站表的SID。

**乘客表**【乘客身份证号，姓名，性别，年龄】

Passenger (PCardID char(18), PName char(20), Sex bit, Age smallint)

其中，主码为乘客身份证号；性别取值为0/1（“1”表示“男”，“0”表示“女”）。

**乘车记录表**【记录编号，乘客身份证号，列车流水号，出发站编号，到达站编号，车厢号，席位排号，席位编号，席位状态】

TakeTrainRecord (RID int, PCardID char(18), TID int, SStationID int, AStationID int, CarrigeID smallint, SeatRow smallint，SeatNo char(1)，SStatus int)

其中，主码、外码请依据应用背景合理定义。

CarrigeID若为空，则表示“无座”；

SeatNo只能取值为’A’、’B’、’C’、’E’、’F’，或为空值；

SStatus只能取值’0’（退票）、’1’（正常）、’2’（乘客没上车）。

**诊断表**【诊断编号，病人身份证号，诊断日期，诊断结果，发病日期】

DiagnoseRecord (DID int, PCardID char(18), DDay date, DStatus smallint, FDay date)

其中，主码为DID；DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

**乘客紧密接触者表**【接触日期, 被接触者身份证号，状态，病患身份证号】

TrainContactor (CDate date, CCardID char(18), DStatus smallint, PCardID char(18))

其中，主码为全码。DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

**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 完成过程

1.建表

建表采用sql语句的办法，同时结合sql server自带的功能进行调整。其中，对车次表进行了微调，取消了(列车名称，发车日期)为候选码的设计，因为这样设计以后，导致不能导入多个日期的同一车和同一日期不同车的数据。建表结果如图4所示：

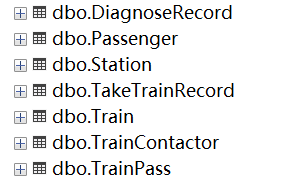


图4.建表结果

表内数据可见excel文件，如图5所示：

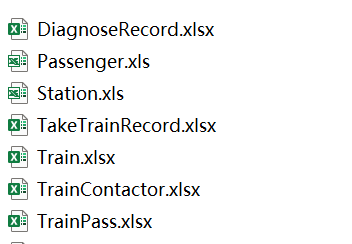


图5.数据

其中，车站表，乘客表用的是老师提供的数据，车次表，车程表是自己结合老师给的数据编的，确诊表，接触表，乘车记录表是在乘客表，车次表，车程表的基础上编的数据。截取乘车记录表如图6所示：

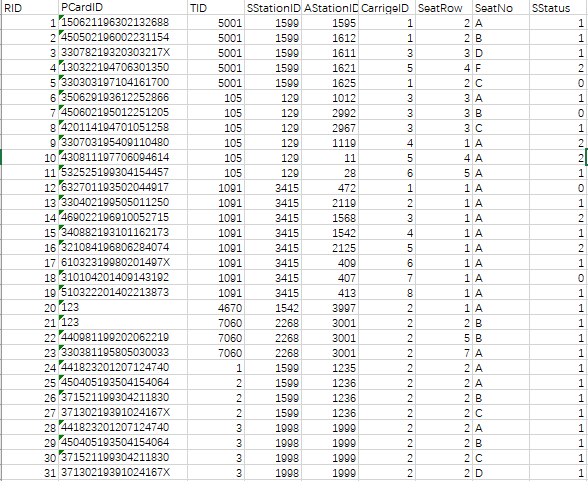


图6.乘车记录表

2.观察性实验  
建立外码必须要参考被参照关系的主码，否则sqlsever会报错，如图7所示：



图7.报错

3.数据准备

见excel文件。

可以在sql server中选择对应数据库->任务->导入数据，数据源选择excel，导入

到对应的表中即可，如图8所示：

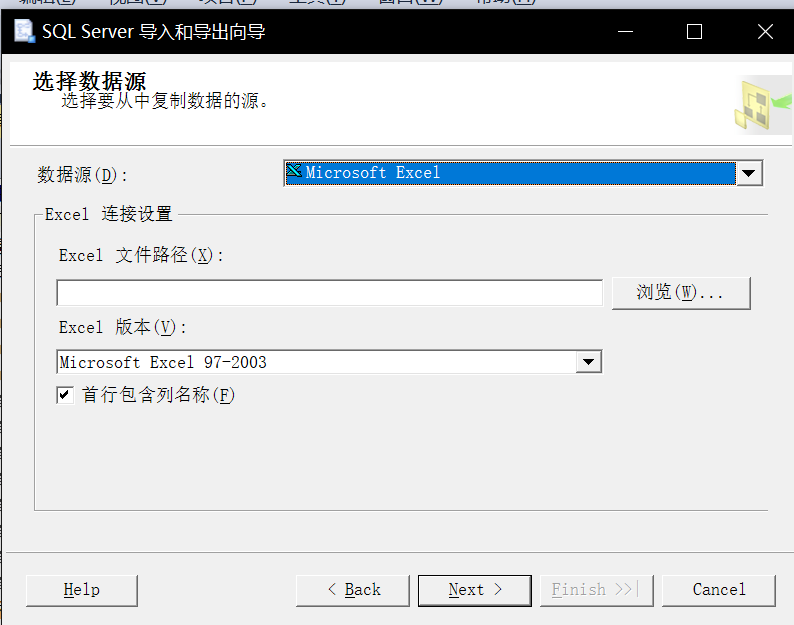


图8.导入数据

## 3.1.3任务总结

实验一最大的难点在于编数据，光是数据就编了我4个小时，同时在做后面的查询时还要回过头来重编。

建表比较简单，如果语句写错了，还能在sql sever里直接可视化调整，让我体会到了sql server的方便。

## 3.2.1 数据更新

**1）分别用一条sql语句完成对乘车记录表基本的增、删、改的操作；**

**2）批处理操作**

将乘车记录表中的从武汉出发的乘客的乘车记录插入到一个新表WH\_TakeTrainRecord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**

创建一个新冠确诊病人的乘火车记录视图，其中的属性包括：身份证号、姓名、年龄、乘坐列车编号、发车日期、车厢号，席位排号，席位编号。按身份证号升序排序，如果身份证号一致，按发车日期降序排序（注意，如果病人买了票但是没坐车，不计入在内）。

**6）触发器实验**

编写一个触发器，用于实现以下完整性控制规则：

1) 当新增一个确诊患者时，若该患者在发病前14天内有乘车记录，则将其同排及前后排乘客自动加入“乘客紧密接触者表”，其中：接触日期为乘车日期。

2) 当一个紧密接触者被确诊为新冠时，从“乘客紧密接触者表”中修改他的状态为“1”。

## 3.2.2 完成过程

1)语句如图9所示：

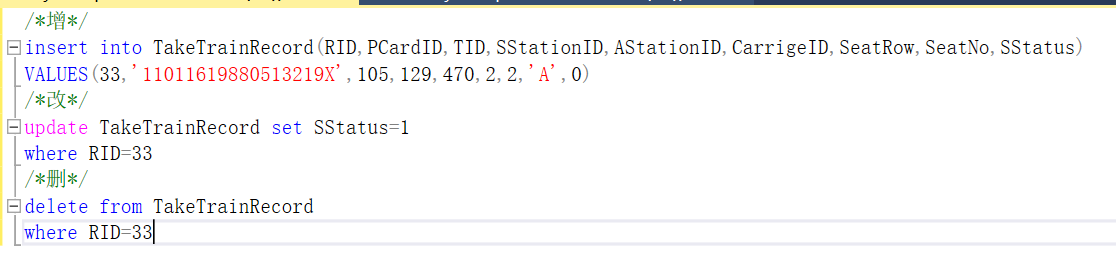


图9.增删改

2）语句如下：

create table WH\_TakeTrainRecord(

RID int PRIMARY KEY,

PCardID char(18) REFERENCES Passenger(PCardID),

TID int REFERENCES Train(TID),

SStationID int REFERENCES Station(SID),

AStationID int REFERENCES Station(SID),

CarrigeID smallint,

SeatRow smallint,

SeatNo char(1),

SStatus int

)

INSERT INTO WH\_TakeTrainRecord(RID,PCardID,TID,SStationID,AStationID,CarrigeID,SeatNo,SeatRow,SStatus)

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

FROM TakeTrainRecord

WHERE SStationID BETWEEN 1593 AND 1616

结果如图11所示：

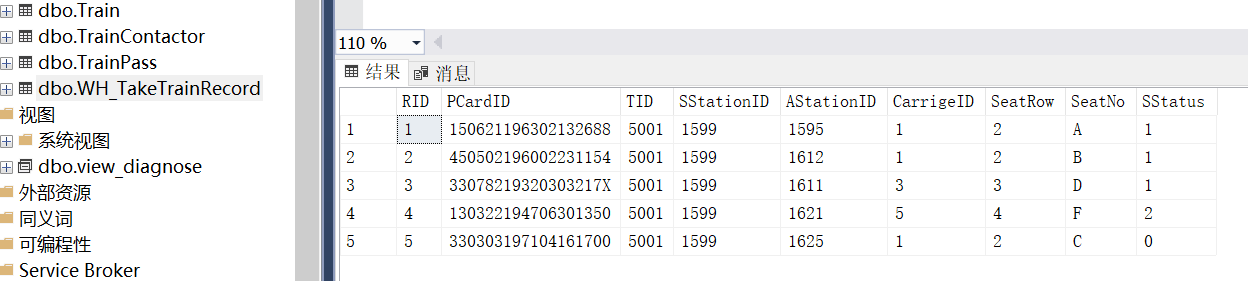


图11.批处理操作

3）见3.1.2数据准备

4）重复插入元组，已有元组都能正常显示，但在可视化视图中无法对已有元组进行修改，应该是不具备实体完整性的后果。

5）语句如下：

create view view\_diagnose

as

select DiagnoseRecord.PCardID,PName,Sex,TakeTrainRecord.TID,SDATE,CarrigeID,SeatRow,SeatNo

from Passenger,TakeTrainRecord,Train,DiagnoseRecord

where Passenger.PCardID=TakeTrainRecord.PCardID

and Passenger.PCardID=DiagnoseRecord.PCardID

and TakeTrainRecord.PCardID=DiagnoseRecord.PCardID

and TakeTrainRecord.TID=Train.TID

and DiagnoseRecord.DStatus=1

and TakeTrainRecord.SStatus=1

go

结果如图12所示：

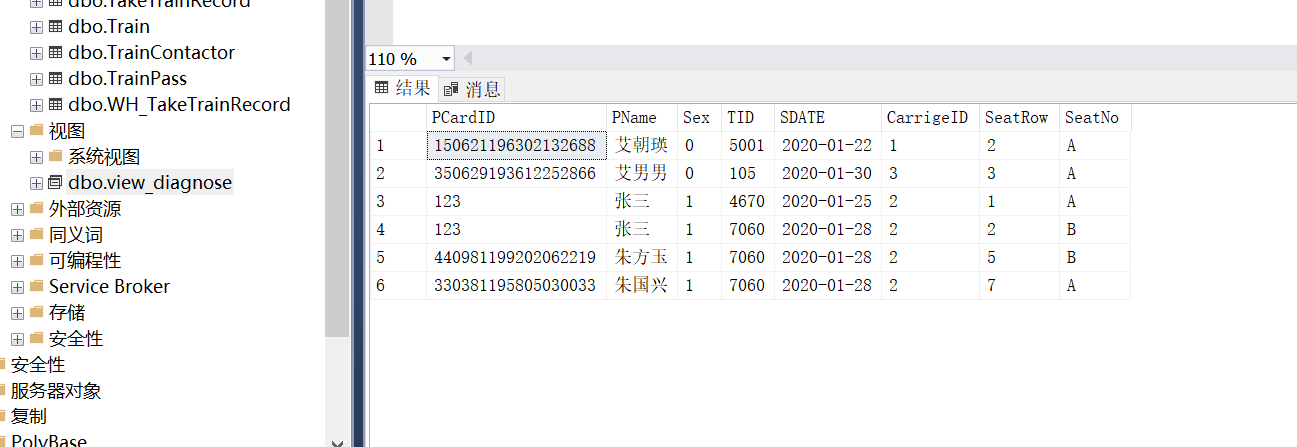


图12.视图

由于创建视图时，sql server不支持排序，所以没有进行排序。

6）没做出来

## 3.2.3 任务总结

在做批处理操作时，对从武汉出发的乘客如何表示产生了疑惑，因为武汉有许多个火车站点，经查表1593~1616都是武汉的站点，故选取SStationID在1593~1616间的乘客插入到新表中，但在做后面的查询时又想到了新的解决方法，可以引入车站表，因为车站表中记录了所有站点的所属城市，故可选择出从武汉出发的所有乘客。遗憾的是触发器实验没有做出来。

## 3.3.1 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）查询确诊者“张三”的在发病前14天内的乘车记录；

2）查询所有从城市“武汉”出发的乘客乘列车所到达的城市名；

3）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；

4）查询“2020-01-22”从“武汉”发出的所有列车；

5）查询“2020-01-22”途经“武汉”的所有列车；

6）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；

7）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。

8）查询2020年1月到达武汉的所有人员；

9） 查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）；

10）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）。

11）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；

12）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；

13）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；

14）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；

15）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；

16）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。

## 3.3.2 完成过程

见下表：

|  |  |
| --- | --- |
| 任务 | SQL语句 |
| 1 | select TakeTrainRecord.\*  from TakeTrainRecord,Passenger,Train,DiagnoseRecord  where TakeTrainRecord.PCardID=Passenger.PCardID  and Passenger.PName='张三'  and TakeTrainRecord.TID=train.TID  and DiagnoseRecord.PCardID=Passenger.PCardID  and Train.SDate  between DATEADD(DAY,-15,DiagnoseRecord.FDay) and DiagnoseRecord.FDay |
| 2 | select s2.cityname  from TakeTrainRecord ,Station s1 ,Station s2  where TakeTrainRecord.SStationID=s1.SID  and TakeTrainRecord.AStationID=s2.SID  and s1.CityName='武汉' |
| 3 | select Passenger.\*,data=DATEDIFF("d",DiagnoseRecord.FDay,DiagnoseRecord.DDay)  from DiagnoseRecord,Passenger  where DiagnoseRecord.DStatus=1  and DiagnoseRecord.PCardID=Passenger.PCardID  order by DATEDIFF("d",DiagnoseRecord.FDay,DiagnoseRecord.DDay) DESC; |
| 4 | select Train.TName  from Train,Station  where Train.SDATE='2020-1-22'  and Train.SStationID=Station.SID  and Station.CityName='武汉'; |
| 5 | select Train.TName  from TrainPass,Station,Train  where TrainPass.SID = Station.SID  and Station.CityName='武汉'  and TrainPass.TID=Train.TID  and TrainPass.STime >='2020-01-27 00:00:00.000'and TrainPass.STIME<='2020-01-28 00:00:00.000' |
| 6 | select Passenger.PCardID,astation.CityName,atrainpass.STime  from Passenger,Station sstation,Station astation,TrainPass strainpass,TrainPass atrainpass,TakeTrainRecord  where Passenger.PCardID=TakeTrainRecord.PCardID  and sstation.SID=TakeTrainRecord.SStationID  and astation.SID=TakeTrainRecord.AStationID  and strainpass.TID=TakeTrainRecord.TID  and strainpass.SID=TakeTrainRecord.SStationID  and atrainpass.TID=TakeTrainRecord.TID  and atrainpass.SID=TakeTrainRecord.AStationID  and sstation.CityName='武汉'  and strainpass.Atime >='2020-01-22 00:00:00.000' and strainpass.ATime<='2020-01-23 00:00:00.000'; |
| 7 | select astation.CityName,count( \*) as sum  from Passenger,Station sstation,Station astation,  TrainPass strainpass,TrainPass atrainpass,TakeTrainRecord  where Passenger.PCardID=TakeTrainRecord.PCardID  and sstation.SID=TakeTrainRecord.SStationID  and astation.SID=TakeTrainRecord.AStationID  and strainpass.TID=TakeTrainRecord.TID  and strainpass.SID=TakeTrainRecord.SStationID  and atrainpass.TID=TakeTrainRecord.TID  and atrainpass.SID=TakeTrainRecord.AStationID  and sstation.CityName='武汉'  and strainpass.Atime >='2020-01-22 00:00:00.000' and strainpass.ATime<='2020-01-23 00:00:00.000'  group by astation.CityName; |
| 8 | select Passenger.PName  from TakeTrainRecord,Station,TrainPass,Passenger  where Station.SID=TakeTrainRecord.AStationID  and Passenger.PCardID=TakeTrainRecord.PCardID  and Station.CityName='武汉'  and TrainPass.TID=TakeTrainRecord.TID  and TrainPass.SID=TakeTrainRecord.SStationID  and TrainPass.ATime between '2020-01-01' and '2020-01-31'; |
| 9 | select distinct Passenger.PName  from (select TrainPass.TID,SNo  from TrainPass,Station  where TrainPass.SID=Station.SID  and Station.CityName='武汉'  and TrainPass.STime between '2020-1-1'and'2020-1-31')  as passbywh(tid,sno),TakeTrainRecord,Passenger,TrainPass snum,TrainPass anum  where TakeTrainRecord.TID = passbywh.tid  and snum.TID=TakeTrainRecord.TID  and snum.SID=TakeTrainRecord.SStationID  and anum.TID=TakeTrainRecord.TID  and anum.SID=TakeTrainRecord.AStationID  and passbywh.sno between snum.SNo and anum.SNo  and Passenger.PCardID=TakeTrainRecord.PCardID  and Passenger.PCardID not like '420%'; |
| 10 | select count(distinct p1.PName) as sum  from Passenger p1,Passenger p2,TakeTrainRecord t1,TakeTrainRecord t2,DiagnoseRecord,Train  where p2.PCardID=t2.PCardID  and Train.TID=t2.TID  and Train.TName='G007'  and Train.SDate='2020-01-22'  and t2.SStatus=1  and p1.PCardID=t1.PCardID  and Train.TID=t1.TID  and Train.TName='G007'  and Train.SDate='2020-01-22'  and t1.SStatus=1  and p2.PCardID=DiagnoseRecord.PCardID  and DiagnoseRecord.DStatus=1  and t1.CarrigeID=t2.CarrigeID |
| 11 | select Train.TName,Train.SDATE,inftrain.CarrigeID  from Train, (select TakeTrainRecord.TID,TakeTrainRecord.CarrigeID  from TakeTrainRecord,DiagnoseRecord  where TakeTrainRecord.PCardID=DiagnoseRecord.PCardID  and DiagnoseRecord.DStatus=1  group by TakeTrainRecord.TID,TakeTrainRecord.CarrigeID  having count(\*)>2)  as inftrain  where Train.TID=inftrain.TID; |
| 12 | select DISTINCT Passenger.PCardID,PName,leavetime=TrainPass.ATime  from Passenger,TakeTrainRecord,TrainPass,DiagnoseRecord,TrainContactor  where Passenger.PCardID=TakeTrainRecord.PCardID  and TakeTrainRecord.SStatus=1  and Passenger.PCardID=DiagnoseRecord.PCardID  and DiagnoseRecord.DStatus=1  and Passenger.PCardID=TrainContactor.PCardID  and TrainContactor.DStatus=3  and TakeTrainRecord.TID=TrainPass.TID  and TakeTrainRecord.SStationID=TrainPass.SID |
| 13 | select Train.TName,Station.CityName  from Train,Station  where Station.SID=Train.AStationID  and Station.CityName in('上海','北京','广州') |
| 14 | select fromwh.PName,fromwh.PCardID,fromwh.TID  from TakeTrainRecord,Station,  (select Passenger.PCardID,Passenger.PName,strainpass.TID  from Passenger,Station sstation,Station astation,TrainPass strainpass,TrainPass atrainpass,TakeTrainRecord  where Passenger.PCardID=TakeTrainRecord.PCardID  and sstation.SID=TakeTrainRecord.SStationID  and astation.SID=TakeTrainRecord.AStationID  and strainpass.TID=TakeTrainRecord.TID  and strainpass.SID=TakeTrainRecord.SStationID  and atrainpass.TID=TakeTrainRecord.TID  and atrainpass.SID=TakeTrainRecord.AStationID  and sstation.CityName='武汉'  and strainpass.ATime between '2020-1-22' and '2020-1-23') as fromwh  where TakeTrainRecord.SStationID=Station.SID  and station.cityname!='武汉'  and TakeTrainRecord.PCardID=fromwh.PCardID  order by fromwh.TID desc,fromwh.PCardID |
| 15 | select Passenger.PName,Passenger.PCardID,Train.TName,Train.SDate  from Passenger left join TakeTrainRecord on Passenger.PCardID=TakeTrainRecord.PCardID  left join Train on Train.TID=TakeTrainRecord.TID  and Train.ATime between '2020.1.1' and '2020.12.31'  inner join DiagnoseRecord on DiagnoseRecord.PCardID=Passenger.PCardID  and DiagnoseRecord.DStatus=1 |
| 16 | select d1.DDay,d1.FDay ,count (\*) as sum  from DiagnoseRecord d1 where d1.DStatus=1  group by d1.DDay ,d1.FDay  order by d1.DDay desc ,d1.FDay desc |

表1.查询语句

|  |  |
| --- | --- |
| 任务 | 结果 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |

表2.查询结果

## 3.3.3 任务总结

在做查询的时候，回去补充了不少数据，第5题中查询1.22号路过武汉的所有列车，由于我的数据中没有1.22号路过武汉的车，但是有1.27号路过武汉的车，所以将题目修改为查询1.27号路过武汉的所有车。在实验中遇到的问题是日期问题，由于车程表中的日期精确到了毫秒，所以查询某一日时会出现不匹配问题，一开始我写的语句是strainpass.Atime >='2020-01-22 00:00:00.000' and strainpass.ATime<='2020-01-23 00:00:00.000'来解决这个问题，后面觉得这样表述不太简洁，于是改成了strainpass.ATime between '2020-1-22' and '2020-1-23'。同时部分题中的‘出发’和‘离开’有些麻烦，因为车程表中记录的第一站是没有抵达，只有离开的。故而此时‘出发’对应的是‘离开’，这个细节卡了我一些时间。

# 附录

/\*Passenger\*/

create TABLE Passenger(

PCardID char(18) PRIMARY KEY,

PName char(20),

Sex bit check (Sex=0 or Sex=1),

Age smallint

)

/\*Station\*/

create TABLE Station(

SID int PRIMARY KEY,

SName char(20),

CityName char(20)

)

/\*Train\*/

create TABLE Train(

TID int PRIMARY KEY,

SDATE date,

TName char(20),

SStationID int FOREIGN KEY REFERENCES Station(SID),

AStationID int FOREIGN KEY REFERENCES Station(SID),

STime datetime,

ATime datetime,

)

/\*TrainPass\*/

create TABLE TrainPass(

TID int,

SNO smallint,

SID int FOREIGN KEY REFERENCES STation(SID),

STime datetime,

ATime datetime,

PRIMARY KEY(TID,SNO)

)

/\*TakeTrainRecord\*/

create TABLE TakeTrainRecord(

RID int PRIMARY KEY,

PCardId char(18) FOREIGN KEY REFERENCES Passenger(PCardID),

TID int FOREIGN KEY REFERENCES Train(TID),

SStationID int FOREIGN KEY REFERENCES Station(SID),

AStationID int FOREIGN KEY REFERENCES Station(SID),

CarrigeID smallint,

SeatRow smallint,

SeatNo char(1) check(SeatNo in('A','B','C','D','E')),

SStatus int check (SStatus in (0,1,2))

)

/\*DiagnoseRecord\*/

create TABLE DiagnoseRecord(

DID int PRIMARY KEY,

PCardID char(18) FOREIGN KEY REFERENCES Passenger(PCardID),

DDay date,

DStatus smallint check(DStatus in (1,2,3)),

FDay date

)

/\*TrainContactor\*/

create TABLE TrainContactor(

CDate date,

CCardID char(18),

DStatus smallint check (DStatus in (1,2,3)),

PCardID char(18),

PRIMARY KEY(CDate,CCardID,DStatus,PCardID)

)

/\*增\*/

insert into TakeTrainRecord(RID,PCardID,TID,SStationID,AStationID,CarrigeID,SeatRow,SeatNo,SStatus)

VALUES(33,'11011619880513219X',105,129,470,2,2,'A',0)

/\*改\*/

update TakeTrainRecord set SStatus=1

where RID=33

/\*删\*/

delete from TakeTrainRecord

where RID=33

/\*批处理操作\*/

create table WH\_TakeTrainRecord(

RID int PRIMARY KEY,

PCardID char(18) REFERENCES Passenger(PCardID),

TID int REFERENCES Train(TID),

SStationID int REFERENCES Station(SID),

AStationID int REFERENCES Station(SID),

CarrigeID smallint,

SeatRow smallint,

SeatNo char(1),

SStatus int

)

INSERT INTO WH\_TakeTrainRecord(RID,PCardID,TID,SStationID,AStationID,CarrigeID,SeatNo,SeatRow,SStatus)

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

FROM TakeTrainRecord

WHERE SStationID BETWEEN 1593 AND 1616

/\*创建视图\*/

create view view\_diagnose

as

select DiagnoseRecord.PCardID,PName,Sex,TakeTrainRecord.TID,SDATE,CarrigeID,SeatRow,SeatNo

from Passenger,TakeTrainRecord,Train,DiagnoseRecord

where Passenger.PCardID=TakeTrainRecord.PCardID

and Passenger.PCardID=DiagnoseRecord.PCardID

and TakeTrainRecord.PCardID=DiagnoseRecord.PCardID

and TakeTrainRecord.TID=Train.TID

and DiagnoseRecord.DStatus=1

and TakeTrainRecord.SStatus=1

go

/\*触发器

create trigger newpatient

on DiagnoseRecord

after insert

as

begin

insert into TrainContactor(CDate,CCardID,DStatus,PCardID)

select Train.SDATE,DiagnoseRecord.PCardID,2,TakeTrainRecord.PCardID

FROM DiagnoseRecord,TakeTrainRecord,Train

WHERE DiagnoseRecord.DStatus=1

and DiagnoseRecord.PCardID=TakeTrainRecord.PCardID

and TakeTrainRecord.TID=Train.TID

and

end

create trigger newcontactor

on DiagnoseRecord

after insert

as

begin

end

\*/

/\*1.查询确诊者“张三”的在发病前14天内的乘车记录\*/

select TakeTrainRecord.\*

from TakeTrainRecord,Passenger,Train,DiagnoseRecord

where TakeTrainRecord.PCardID=Passenger.PCardID

and Passenger.PName='张三'

and TakeTrainRecord.TID=train.TID

and DiagnoseRecord.PCardID=Passenger.PCardID

and Train.SDate

between DATEADD(DAY,-15,DiagnoseRecord.FDay) and DiagnoseRecord.FDay

/\*2.查询所有从城市“武汉”出发的乘客乘列车所到达的城市名\*/

select s2.cityname

from TakeTrainRecord ,Station s1 ,Station s2

where TakeTrainRecord.SStationID=s1.SID

and TakeTrainRecord.AStationID=s2.SID

and s1.CityName='武汉'

/\*3.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\*/

select Passenger.\*,data=DATEDIFF("d",DiagnoseRecord.FDay,DiagnoseRecord.DDay)

from DiagnoseRecord,Passenger

where DiagnoseRecord.DStatus=1

and DiagnoseRecord.PCardID=Passenger.PCardID

order by DATEDIFF("d",DiagnoseRecord.FDay,DiagnoseRecord.DDay) DESC;

/\*4.查询“2020-01-22”从“武汉”发出的所有列车\*/

select Train.TName

from Train,Station

where Train.SDATE='2020-1-22'

and Train.SStationID=Station.SID

and Station.CityName='武汉';

/\*5.查询“2020-01-27”途经“武汉”的所有列车\*/

select Train.TName

from TrainPass,Station,Train

where TrainPass.SID = Station.SID

and Station.CityName='武汉'

and TrainPass.TID=Train.TID

and TrainPass.STime >='2020-01-27 00:00:00.000'and TrainPass.STIME<='2020-01-28 00:00:00.000'

/\*6.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\*/

select Passenger.PCardID,astation.CityName,atrainpass.STime

from Passenger,Station sstation,Station astation,TrainPass strainpass,TrainPass atrainpass,TakeTrainRecord

where Passenger.PCardID=TakeTrainRecord.PCardID

and sstation.SID=TakeTrainRecord.SStationID

and astation.SID=TakeTrainRecord.AStationID

and strainpass.TID=TakeTrainRecord.TID

and strainpass.SID=TakeTrainRecord.SStationID

and atrainpass.TID=TakeTrainRecord.TID

and atrainpass.SID=TakeTrainRecord.AStationID

and sstation.CityName='武汉'

and strainpass.Atime >='2020-01-22 00:00:00.000' and strainpass.ATime<='2020-01-23 00:00:00.000';

/\*7.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\*/

select astation.CityName,count( \*) as sum

from Passenger,Station sstation,Station astation,

TrainPass strainpass,TrainPass atrainpass,TakeTrainRecord

where Passenger.PCardID=TakeTrainRecord.PCardID

and sstation.SID=TakeTrainRecord.SStationID

and astation.SID=TakeTrainRecord.AStationID

and strainpass.TID=TakeTrainRecord.TID

and strainpass.SID=TakeTrainRecord.SStationID

and atrainpass.TID=TakeTrainRecord.TID

and atrainpass.SID=TakeTrainRecord.AStationID

and sstation.CityName='武汉'

and strainpass.Atime >='2020-01-22 00:00:00.000' and strainpass.ATime<='2020-01-23 00:00:00.000'

group by astation.CityName;

/\*8.查询2020年1月到达武汉的所有人员\*/

select Passenger.PName

from TakeTrainRecord,Station,TrainPass,Passenger

where Station.SID=TakeTrainRecord.AStationID

and Passenger.PCardID=TakeTrainRecord.PCardID

and Station.CityName='武汉'

and TrainPass.TID=TakeTrainRecord.TID

and TrainPass.SID=TakeTrainRecord.SStationID

and TrainPass.ATime between '2020-01-01' and '2020-01-31';

/\*9.查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）\*/

select distinct Passenger.PName

from (select TrainPass.TID,SNo

from TrainPass,Station

where TrainPass.SID=Station.SID

and Station.CityName='武汉'

and TrainPass.STime between '2020-1-1'and'2020-1-31')

as passbywh(tid,sno),TakeTrainRecord,Passenger,TrainPass snum,TrainPass anum

where TakeTrainRecord.TID = passbywh.tid

and snum.TID=TakeTrainRecord.TID

and snum.SID=TakeTrainRecord.SStationID

and anum.TID=TakeTrainRecord.TID

and anum.SID=TakeTrainRecord.AStationID

and passbywh.sno between snum.SNo and anum.SNo

and Passenger.PCardID=TakeTrainRecord.PCardID

and Passenger.PCardID not like '420%';

/\*10.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）\*/

select count(distinct p1.PName) as sum

from Passenger p1,Passenger p2,TakeTrainRecord t1,TakeTrainRecord t2,DiagnoseRecord,Train

where p2.PCardID=t2.PCardID

and Train.TID=t2.TID

and Train.TName='G007'

and Train.SDate='2020-01-22'

and t2.SStatus=1

and p1.PCardID=t1.PCardID

and Train.TID=t1.TID

and Train.TName='G007'

and Train.SDate='2020-01-22'

and t1.SStatus=1

and p2.PCardID=DiagnoseRecord.PCardID

and DiagnoseRecord.DStatus=1

and t1.CarrigeID=t2.CarrigeID

/\*11.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\*/

select Train.TName,Train.SDATE,inftrain.CarrigeID

from Train, (select TakeTrainRecord.TID,TakeTrainRecord.CarrigeID

from TakeTrainRecord,DiagnoseRecord

where TakeTrainRecord.PCardID=DiagnoseRecord.PCardID

and DiagnoseRecord.DStatus=1

group by TakeTrainRecord.TID,TakeTrainRecord.CarrigeID

having count(\*)>2)

as inftrain

where Train.TID=inftrain.TID;

/\*12.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\*/

select DISTINCT Passenger.PCardID,PName,leavetime=TrainPass.ATime

from Passenger,TakeTrainRecord,TrainPass,DiagnoseRecord,TrainContactor

where Passenger.PCardID=TakeTrainRecord.PCardID

and TakeTrainRecord.SStatus=1

and Passenger.PCardID=DiagnoseRecord.PCardID

and DiagnoseRecord.DStatus=1

and Passenger.PCardID=TrainContactor.PCardID

and TrainContactor.DStatus=3

and TakeTrainRecord.TID=TrainPass.TID

and TakeTrainRecord.SStationID=TrainPass.SID

/\*13.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\*/

select Train.TName,Station.CityName

from Train,Station

where Station.SID=Train.AStationID

and Station.CityName in('上海','北京','广州')

/\*14.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\*/

select fromwh.PName,fromwh.PCardID,fromwh.TID

from TakeTrainRecord,Station,

(select Passenger.PCardID,Passenger.PName,strainpass.TID

from Passenger,Station sstation,Station astation,TrainPass strainpass,TrainPass atrainpass,TakeTrainRecord

where Passenger.PCardID=TakeTrainRecord.PCardID

and sstation.SID=TakeTrainRecord.SStationID

and astation.SID=TakeTrainRecord.AStationID

and strainpass.TID=TakeTrainRecord.TID

and strainpass.SID=TakeTrainRecord.SStationID

and atrainpass.TID=TakeTrainRecord.TID

and atrainpass.SID=TakeTrainRecord.AStationID

and sstation.CityName='武汉'

and strainpass.ATime between '2020-1-22' and '2020-1-23') as fromwh

where TakeTrainRecord.SStationID=Station.SID

and station.cityname!='武汉'

and TakeTrainRecord.PCardID=fromwh.PCardID

order by fromwh.TID desc,fromwh.PCardID

/\*15.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\*/

select Passenger.PName,Passenger.PCardID,Train.TName,Train.SDate

from Passenger left join TakeTrainRecord on Passenger.PCardID=TakeTrainRecord.PCardID

left join Train on Train.TID=TakeTrainRecord.TID

and Train.ATime between '2020.1.1' and '2020.12.31'

inner join DiagnoseRecord on DiagnoseRecord.PCardID=Passenger.PCardID

and DiagnoseRecord.DStatus=1

/\*16.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\*/

select d1.DDay,d1.FDay ,count (\*) as sum

from DiagnoseRecord d1 where d1.DStatus=1

group by d1.DDay ,d1.FDay

order by d1.DDay desc ,d1.FDay desc

# 4 综合实践任务

## 4.1系统设计目标

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。

## 4.2 需求分析

（1）功能需求

见系统设计目标

（2）安全性与完整性需求

1.安全性需求：数据备份、数据恢复、防止系统故障造成损失

2.完整性需求：各种信息记录的完整性，信息记录内容不能为空、各种数据间相互联系的正确性、相同的数据在不同记录中的一致性

（3）数据流图

1.会员信息图

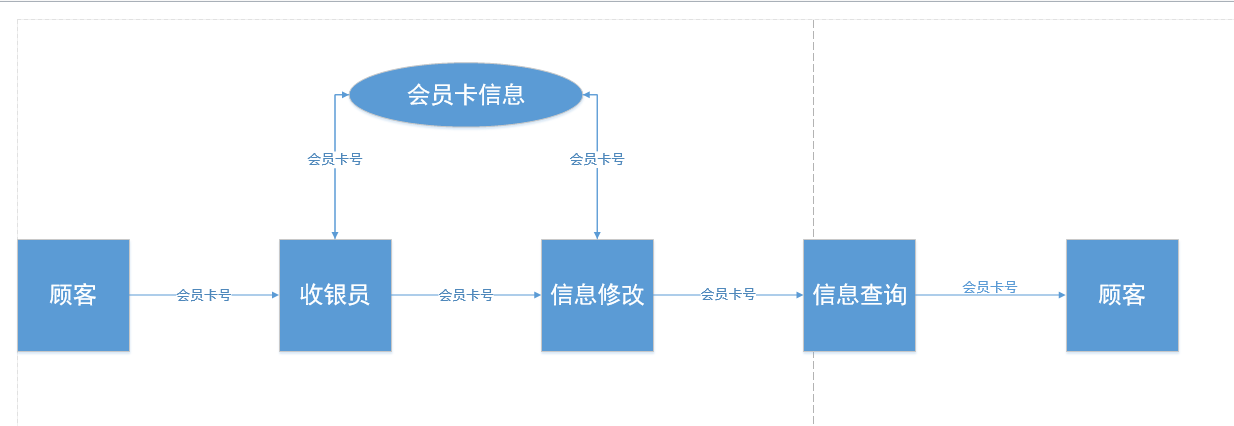


图1.会员信息图

2.交易流程图

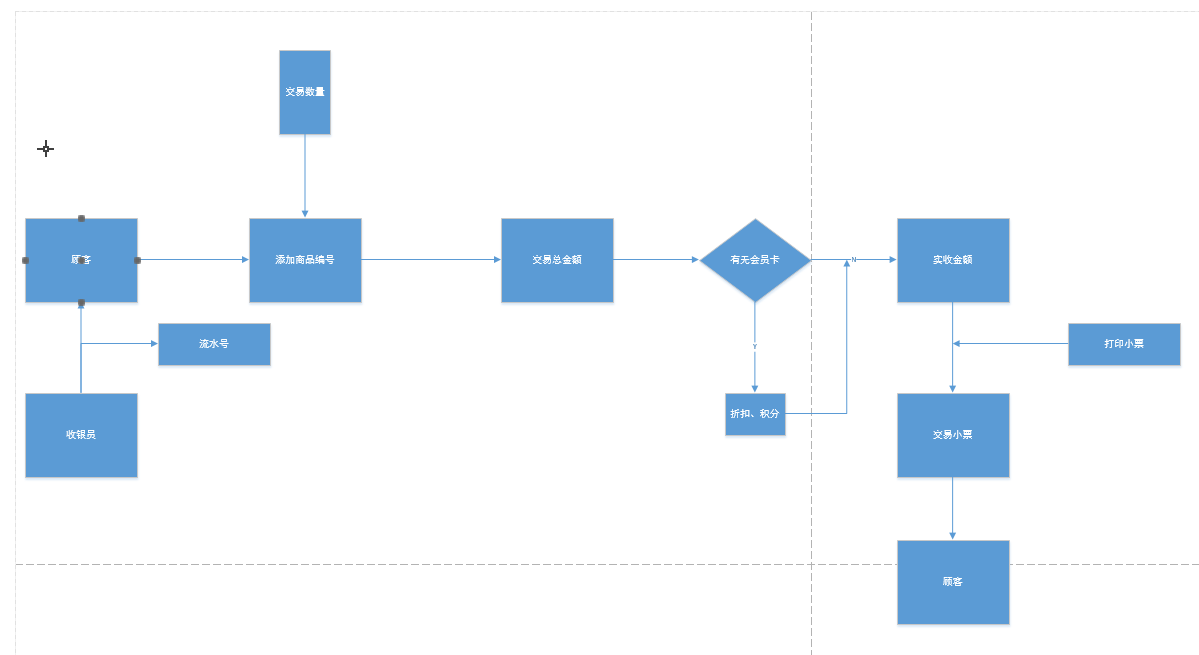


图2.交易流程图

（3）数据字典

1.顾客数据字典

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 会员卡号 | 会员卡号 | char | 30 |
| 顾客姓名 | 顾客姓名 | char | 10 |
| 顾客性别 | 顾客性别 | char | 2 |
| 积分信息 | 积分信息 | int |  |

2.收银员数据字典

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 收银员编号 | 收银员编号 | char | 30 |
| 收银员姓名 | 收银员姓名 | char | 10 |
| 收银员性别 | 收银员性别 | char | 2 |
| 登录名 | 登录名 | char | 20 |
| 密码 | 密码 | char | 16 |

3.商品信息数据字典

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 商品编号 | 商品编号 | char | 30 |
| 商品名称 | 商品名称 | char | 20 |
| 单价 | 单价 | float |  |

4.交易数据字典

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 收银员编号 | 收银员编号 | char | 30 |
| 收银金额 | 收银金额 | float |  |
| 登陆时间 | 登陆时间 | date |  |
| 注销时间 | 注销时间 | date |  |

4.订单数据字典

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 收银员编号 | 收银员编号 | char | 30 |
| 商品编号 | 商品编号 | char | 30 |
| 交易数量 | 交易数量 | int |  |
| 交易时间 | 交易时间 | date |  |
| 总金额 | 总金额 | float |  |
| 实收金额 | 实收金额 | float |  |

## 4.3 总体设计

采用mysql+java实现。具体功能见上。

## 4.4 数据库设计

1.实体：收银员，顾客，商品，订单

2.局部E-R图：

（1）收银员：

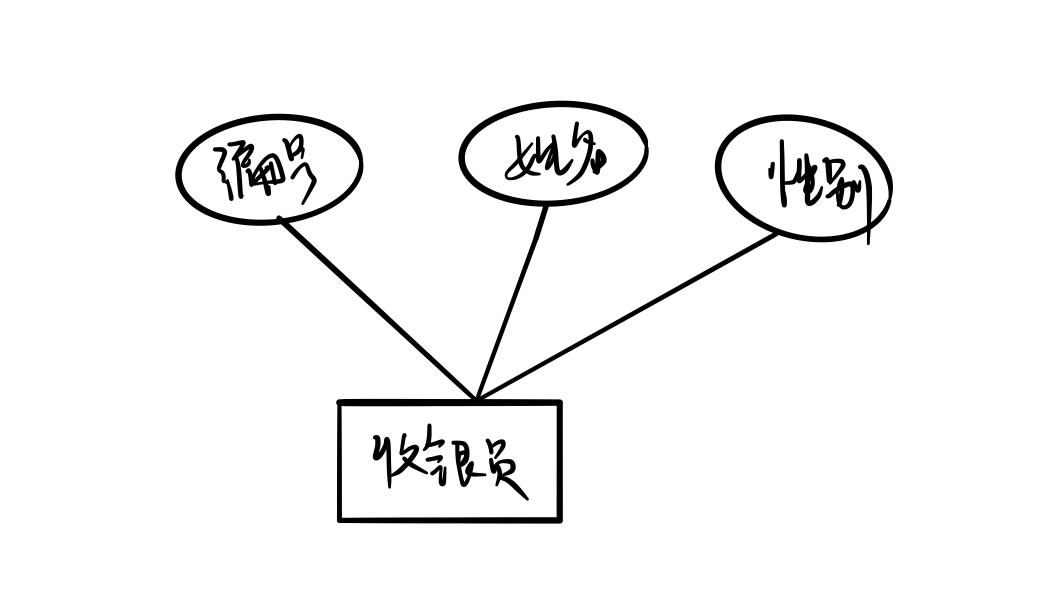


图3.收银员E-R图

（2）会员：

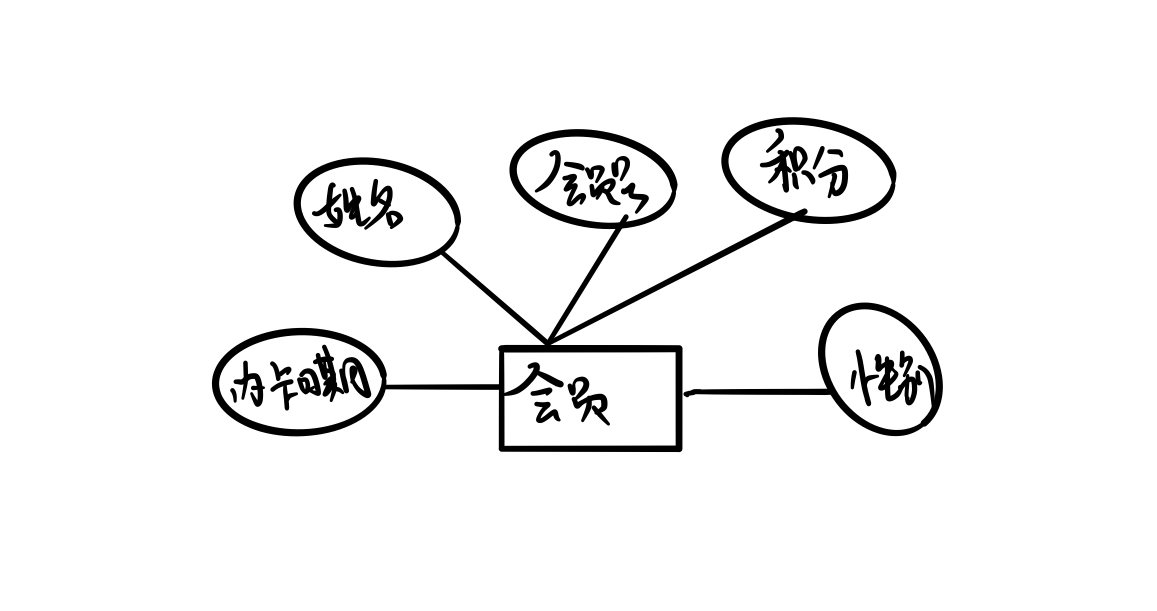


图4.会员E-R图

（3）商品：

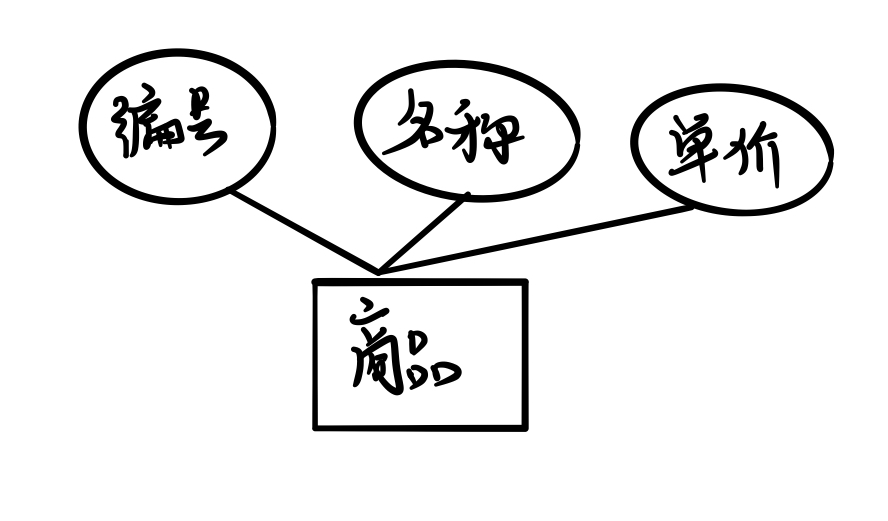


图5.商品E-R图

3.整体E-R图：

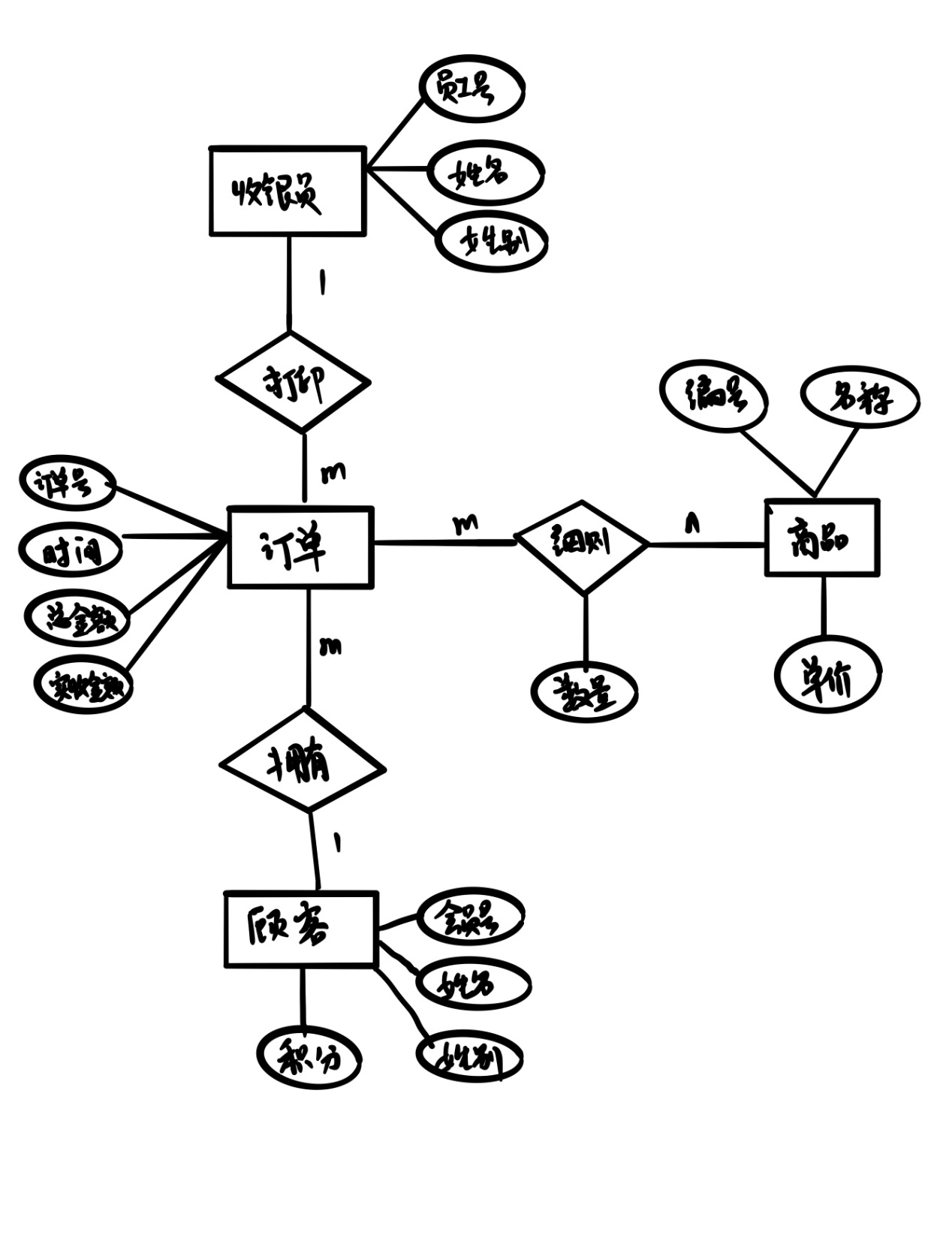


图6.整体E-R图

4.关系模型（下划线代表主键，红色代表外键）

1. 会员（会员卡号、姓名、性别、办卡日期、积分）
2. 收银员（收银员编号、姓名、性别、登录名、密码）
3. 商品（商品编号、商品名、单价）
4. 订单（订单号、收银员编号、会员号、总金额、实收金额、交易时间）
5. 订单细则（订单号、商品编号、交易数量）

5.设计表与字段

* 会员表（Member）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段类型 | 说明 | 是否允许为空 | 默认值 |
| 1 | MNum | char（30） | 会员卡号（主键） | 否 |  |
| 2 | MName | char（10） | 会员姓名 | 否 |  |
| 3 | MSex | char（2） | 会员性别 | 是 |  |
| 4 | MPoint | int | 会员积分 | 是 | 0 |
| 5 | MDate | Date | 办卡日期 | 是 |  |

* 商品表（Goods）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段类型 | 说明 | 是否允许为空 | 默认值 |
| 1 | GNum | char（30） | 商品编号（主键） | 否 |  |
| 2 | GName | char（10） | 商品名 | 否 |  |
| 3 | GPrice | int | 商品价格 | 否 |  |
| 4 | GPoint | int | 商品积分 | 否 | 0 |

* 收银员表（Cashier）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段类型 | 说明 | 是否允许为空 | 默认值 |
| 1 | CNum | char（30） | 收银员编号（主键） | 否 |  |
| 2 | CName | char（10） | 收银员姓名 | 否 |  |
| 3 | CSex | char(2) | 收银员性别 | 是 |  |
| 4 | CUser | char（20） | 登录名 | 否 |  |
| 5 | CPassword | char（16） | 登陆密码 | 否 |  |

* 订单记录表（Order）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段类型 | 说明 | 是否允许为空 | 默认值 |
| 1 | ONum | char（30） | 订单编号（主键） | 否 |  |
| 2 | MNum | char（30） | 会员标号（外键） | 否 |  |
| 3 | CNum | char（30） | 收银员编号（外键） | 否 |  |
| 4 | AllPrice | Double | 总金额 | 否 |  |
| 5 | TotalPrice | Double | 实收金额 | 否 |  |
| 6 | ODate | Date | 交易时间 | 否 |  |

* 订单细则表（Detail）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段类型 | 说明 | 是否允许为空 | 默认值 |
| 1 | ONum | char（30） | 订单编号（外键） | 否 |  |
| 2 | GNum | char（30） | 商品编号（外键） | 否 |  |
| 3 | Quantity | int | 交易数量 | 否 |  |

## 4.5 详细设计与实现

1.业务流程

首先收银员使用账户密码登陆，登陆后来到主界面，可以查看客户信息，订单信息，商品信息，以及收银，还可选择退出登陆或者关闭程序。

2.会员机制实现

为了简单起见，所有客户都储存在会员表里，购物可以获得积分，积分大于1000以上的为会员，会员购物打九折。为了满足设计要求，客户只有是会员（即积分大于1000）购物才会累加积分，或者购物金额大于1000时累加积分成为会员。故在生成订单时，加了一个判断，若客户现有积分+购物积分大于1000，则客户积分=现有积分+购物积分，否则不加积分。由于客户可以花费50办理会员卡，所以在商品表里添加了一个会员卡商品，金额为50，积分为1000，客户购买后会满足前面积分添加规则， 成为会员。

3.订单生成实现

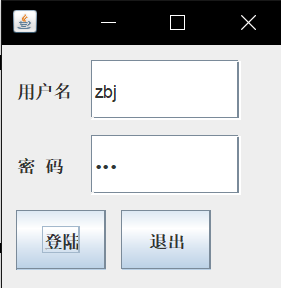
收银功能中提供客户编码，收银员编码，以及商品选择，商品选择在订单细则表里，最终生成的订单可在收银员端查看，收银员只能查看自己收银的订单。

4.技术实现

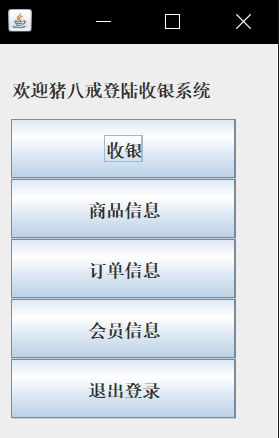
采用javaswing编写，数据库连接使用mysql-connect包里的接口实现。

## 4.6 系统测试

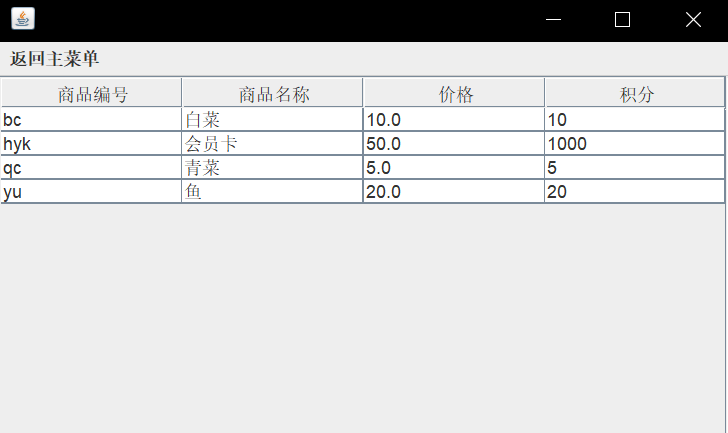
1.首先登陆



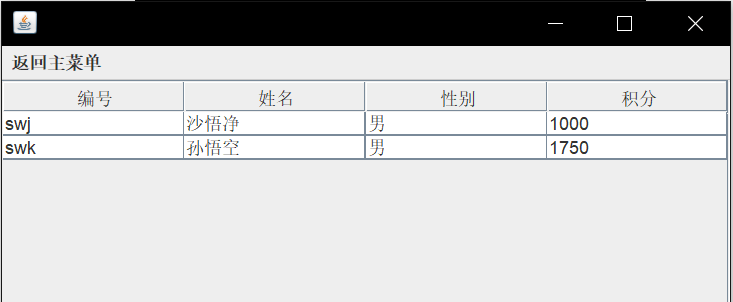
2.来到主界面



3.查看商品信息



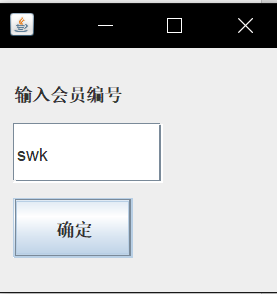
4.查看会员信息



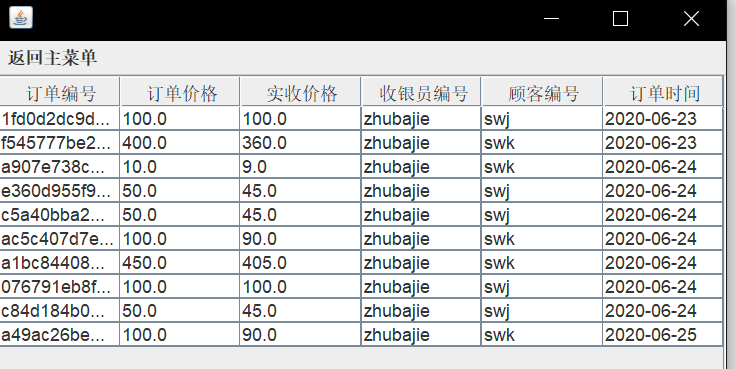
5.收银界面



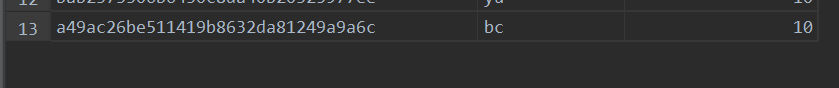
6.购买10颗白菜，用会员编号结算



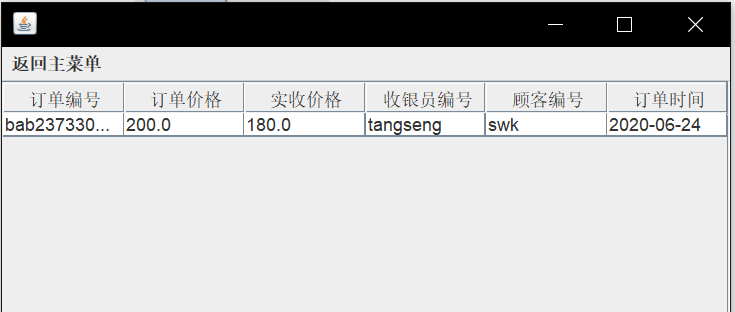
7.查看订单信息



8.查看订单细则是否收录



9.换一个收银员登陆，查询不到刚才的订单信息（不属于自己）



## 4.7 系统设计与实现总结

1.建立数据库，由于java对于sql server的操作特别复杂，所以在本次实验中使用mysql的数据库

2.设计表，建立表，采用navicat工具建表

3.实现展示，由于本次实验采用java swing+mysql形式，故触发器不储存在数据库里，而以后端代码形式实现。主要是实现订单和订单细则的生成，以及成为会员和会员积分的增加。

4.一开始是打算进行会员管理功能的实现，但由于建表时，订单表里需绑定会员编号，由于这个是外键，不能为空，故将所有的顾客都储存在member表里，积分大于1000识别为会员。

5.在实现订单细则功能时，由于外码的设置，导致我需要先生成订单再生成订单细则， 否则订单细则表没有可以引用的订单号，由于一开始没有注意到这一点，导致我生成的订单价格出现问题，再仔细研究后发现问题，修改了代码逻辑，在最后生成订单细则。

# 4 课程总结

这次实验采用的java+mysql的形式实现，一次性加强了两门课的知识掌握。主要技术采用java swing，在实验设计中，我主要实现了登陆功能，收银功能，会员查看，商品查看，订单查看功能。注册会员的功能被我简化为了根据积分判断是否为会员，顾客可以花50购买会员卡这一商品直接加1000积分成为会员。为了满足订单中顾客编号不为空的设计要求，我将所有的顾客信息都存在了member表里，但实际收银中不可能这样实现，因为很多人不是长期顾客。我之后想到了一个办法，就是在member表里添加一个游客信息，非会员的编号都填这个，但是这样的话我得另外增加一个会员注册功能，由于时间原因，我无法实现，希望以后可以继续完善这个系统，让其能完全满足小型超市的收银需求。

# 附录

收银代码:

package service;  
import dao.DetailDao;  
import dao.MemberDao;  
import dao.OrderDao;  
import entity.\*;  
import util.DBUtil;  
import util.OrderUtil;  
  
import java.util.Date;  
import java.util.List;  
  
public class OrderService {  
 private OrderDao orderDao = new OrderDao();  
 private MemberDao memberDao = new MemberDao();  
 private DetailDao detailDao = new DetailDao();  
 public int addOrder(Cashier c, Member m){  
 List<Detail> list = OrderUtil.getGoodsList();  
 Order o = new Order();  
 o.setOnum(DBUtil.getUUID());  
 o.setOdate(new Date());  
 o.setCnum(c.getCnum());  
 o.setMnum(m.getMnum());  
 double allprice = 0;  
 double totalprice = 0;  
 for (Goods goods : OrderUtil.goodsList) {  
 if(goods.getGname().equals("会员卡")){  
 memberDao.update(m.getMnum(),950);  
 }  
 for (Detail detail : list) {  
 detail.setOnum(o.getOnum());  
 if(detail.getGnum().equals(goods.getGnum())){  
 allprice += goods.getGprice()\*detail.getQuantity();  
 System.*out*.println("计算结果:"+allprice);  
 }  
 }  
 }  
 m = memberDao.findByNum(m.getMnum());  
 if(m.getMpoint()+allprice>=1000){  
 totalprice = allprice \* 0.9;  
 memberDao.update(m.getMnum(),(int)allprice);  
 }else{  
 totalprice = allprice;  
 }  
 o.setTotalprice(totalprice);  
 o.setAllprice(allprice);  
 orderDao.insert(o);  
 for (Detail detail : list) {  
 detailDao.insert(detail);  
 }  
 OrderUtil.clear();  
 return 0;  
 }  
}