



华中科技大学

数据库系统原理实践报告

项目名称： 连锁品牌零售店库存管理系统

姓 名： 熊逸钦

专 业： 计算机科学与技术

班 级： CS1701

学 号： U201714501

指导教师： 潘鹏

分数	
教师签名	

2020 年 5 月 18 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
1.1 软件功能学习部分	1
1.2 SQL 练习部分	1
1.3 数据库应用系统设计	4
2 软件功能学习	5
2.1 任务要求	5
2.2 完成过程	5
2.3 任务总结	8
3 SQL 练习	9
3.1 任务要求	9
3.2 完成过程	9
3.3 任务总结	27
4 综合实践任务	28
4.1 系统设计目标	28
4.2 需求分析	28
4.3 总体设计	33
4.4 数据库设计	35
4.5 详细设计与实现	38
4.6 系统测试	49
4.7 系统设计与实现总结	57
5 课程总结	59
附录	60

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。

本次数据库课程任务分为三个主要部分，第一部分为数据库软件功能学习部分，具体任务描述对应下述第 1.1 节；第二部分为 SQL 语句练习部分，具体任务描述对应下述第 1.2 节；第三部分为综合实践任务，要求基于数据库的相关知识设计并实现一个 B/S 或 C/S 架构的应用系统，具体任务描述对应下述第 1.3 节。

1.1 软件功能学习部分

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。

1.2 SQL 练习部分

1.2.1 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【列车编号，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，主码为(列车编号)；SStationID 和 AStationID 都来源于车站表的 SID。

车程表【列车编号，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(列车编号，车站序号)。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，列车编号，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, SDate date, 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。

1.2.2 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 1.2.1 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”。

1.2.3 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。

1.2.4 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。

1.2.5 DBMS 函数及存储过程和事务 (选做)

1) 编写一个依据乘客身份证号计算其在指定年乘列车的乘车次数的自定义函数, 并利用其查询 2020 年至少乘车过 3 次的乘客。

2) 尝试编写 DBMS 的存储过程, 建立每趟列车的乘坐人数的统计表, 并通过存储过程更新该表。

3) 尝试在 DBMS 的交互式界面中验证事务机制的执行效果。

1.3 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。

2 软件功能学习

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 实验环境

- 1) 操作系统：Windows 10 (x64)
- 2) 数据库软件：SQL Server 2008 R2
- 3) 数据库管理系统：SQL Server Management Studio 2008 R2

2.2.2 练习备份方式

- 1) 通过数据库软件备份

首先在 SQL Server Management Studio 中的服务器对象->备份设备页面设置备份设备，如下图 2.1 所示，将备份设备的名字设置为“bak_dev1”：



图 2.1 设置备份设备

然后右键点击需要备份的数据库，选择任务->备份，设置源和备份集的属性之后，在目标中选择之前设置的备份设备“bak_dev1”，如下图 2.2 所示：

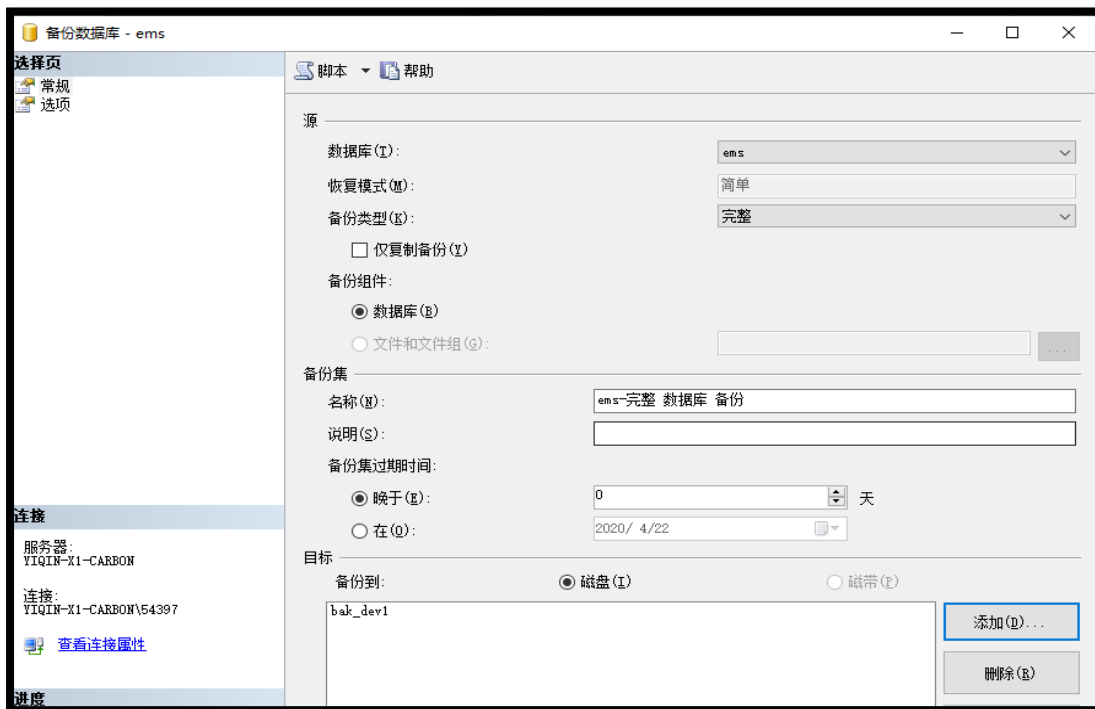


图 2.2 备份任务设置

设置完成后，点击确定，结果如下图 2.3 所示，可知成功完成了软件备份。



图 2.3 软件备份成功

2) 通过文件形式进行脱机备份

右键点击需要备份的数据库，选择任务->导出数据，进入 SQL Server 导入和导出向导界面，选择导出目标为平面文件目标，如下图 2.4 所示：



图 2.4 数据库导出为平面文件

导出结果为一个 csv 格式文件，文件中使用逗号作为数据的分隔符，还原数据时只需要对目标数据库右键，任务->导入数据，选择下图 2.5 所示文件即可。

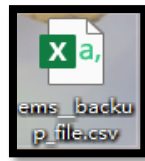


图 2.5 导出成功后的备份文件

2.2.3 练习增加用户并配置权限的操作

1) 增加用户

在 SQL Server Management Studio 的安全性选项卡下右键登录名，选择新建登录名，按下图 2.6 所示进行设置，密码自定。

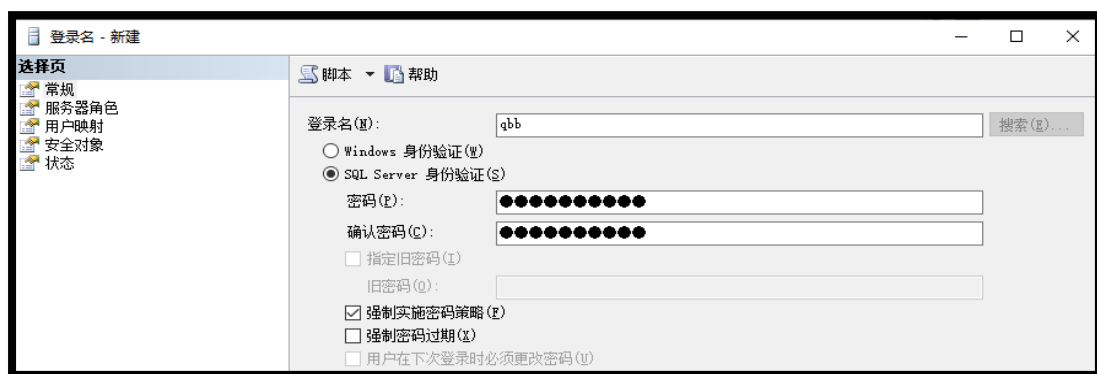


图 2.6 新建登录名

2) 增加权限

给用户 qbb 增加访问数据库 qbb 的权限，执行语句为 EXEC sp_grantdbaccess 'qbb', 'qbb'，执行结果如下图 2.7 所示：



图 2.7 增加数据库访问权限结果

给用户 qbb 新增在 employee 表上查询的权限，执行语句为 grant select on employee to qbb，执行结果如下图 2.8 所示：

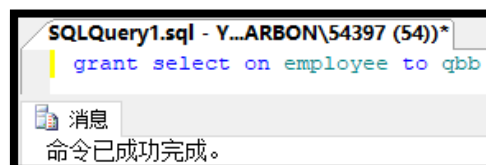


图 2.8 增加表查询权限

2.3 任务总结

在上述实验中，我通过对 SQL Server 2008 R2 数据库软件中的特定数据库进行备份操作以及进行增加用户并配置权限的操作，使我对该数据库软件以及配套的数据库管理工具的软件功能有了较为深刻的了解。

实验完成了对数据库的软件备份和脱机备份，软件备份需要设置一个备份文件，本质上和脱机备份一样，都是将数据库中的数据定期存储在磁盘上，但他们的不同点在于，软件备份可以在数据库提供服务期间完成，并可以自动地将数据备份至备份文件，并设置还原点，是增量备份。而脱机备份则是将数据库中的数据导出为平面文件，无法实现备份数据的实时更新。

在给新增用户提供访问数据库的权限时我遇到了困难，因为 SQL 语句没有针对用户与数据库之间权限的特定语句，因此需要使用到 EXEC 动态语句，并使用 sp_grantdbaccess 存储过程来实现对用户的权限赋予操作。

3 SQL 练习

3.1 任务要求

按照第 1.2 节中的内容，以“疫情期间乘坐列车”为主题，建立数据库并使用 SQL 语句实现建表、数据更新和查询这些动作。

3.2 完成过程

在 SQL Server Management Studio 中手动建立名为“Epidemic_Train”的数据库，然后针对这个数据库进行建表、数据更新和查询操作。

3.2.1 建表

1) 创建跟“疫期乘坐列车”相关的关系，包括主码和外码的说明

总共需要建立车站表、车次表、车程表、乘客表、乘车记录表、诊断表和乘客紧密接触者表这 7 张表，建表语句如下图 3.1 所示。

<pre>--车站表【车站编号，车站名，所属城市】-- CREATE TABLE Station(SID INT PRIMARY KEY, --车站编号 SName CHAR(20), --车站名 CityName CHAR(30) --所属城市) --车次表【列车编号，发车日期，列车名称，起点站编号，终点站 CREATE TABLE Train(TID INT PRIMARY KEY, --列车编号 SDate DATE, --发车日期 TName CHAR(20) NOT NULL, --列车名称 SStationID INT, --起点站编号 AStationID INT, --终点站编号 STime DATETIME, --开出时刻 ATime DATETIME, --终点时刻 FOREIGN KEY (SStationID) REFERENCES Station(SID), FOREIGN KEY (AStationID) REFERENCES Station(SID)) --车程表【列车编号，车站序号，车站编号，到达时刻， CREATE TABLE TrainPass(TID INT, --列车编号 SNO SMALLINT, --车站序号 SID INT, --车站编号 STime DATETIME, --到达时刻 ATime DATETIME, --离开时刻 PRIMARY KEY (TID, SNO), FOREIGN KEY (SID) REFERENCES Station(SID)) --乘客表【乘客身份证号，姓名，性别，年龄】-- CREATE TABLE Passenger(PCardID CHAR(18) PRIMARY KEY, --乘客身份证号 PName CHAR(20), --姓名 Sex bit, --性别(1男0女) Age SMALLINT --年龄)</pre>	<pre>--乘车记录表【记录编号，乘客身份证号，列车编号，出发站编号，到达站编号，车厢 CREATE TABLE TakeTrainRecord(RID INT, --记录编号 PCardID CHAR(18), --乘客身份证号 TID INT, --列车编号 SStationID INT, --出发站编号 AStationID INT, --到达站编号 CarriageID SMALLINT, --车厢号 (比如2车16F中的2, NULL表示无座) SeatRow SMALLINT, --席位排号 (比如2车16F中的16) SeatNo CHAR(1), --席位编号 (比如2车16F中的F, A\B\C\E\F或NULL) SStatus INT, --席位状态 (0--退票, 1--正常, 2--乘客没上车) 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)) --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】-- CREATE TABLE DiagnoseRecord(DID INT PRIMARY KEY, --诊断编号 PCardID CHAR(18), --病人身份证号 DDay DATE, --诊断日期 DStatus SMALLINT, --诊断结果 (1--确诊, 2--疑似, 3--排除新冠) FDay DATE, --发病日期) --乘客紧密接触者表【接触日期，被接触者身份证号，接触者状态，病患身份证号】-- CREATE TABLE TrainContactor(CDate DATE, --接触日期 CCardID CHAR(18), --被接触者身份证号 DStatus SMALLINT, --接触者状态 (1--确诊, 2--疑似, 3--排除新冠) PCardID CHAR(18), --接触的病患的身份证号 PRIMARY KEY (CDate, CCardID, DStatus, PCardID), FOREIGN KEY (PCardID) REFERENCES Passenger(PCardID), FOREIGN KEY (CCardID) REFERENCES Passenger(PCardID))</pre>
---	---

图 3.1 建表 SQL 语句

在执行上述建表语句后，进入 Management Studio 中查看数据库中各表建立情况，如图 3.2 所示。

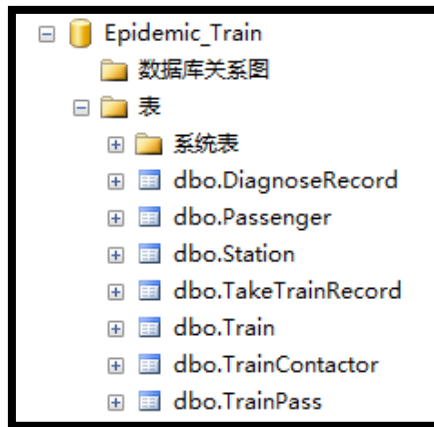


图 3.2 建表结果

2) 观察性实验

在建立外码时，必须要参考被参照关系的主码，下面的例子中，Station 表的主码是 SID，TEST 表中的 SName 这个外码参照了 Station 表中的 SName 属性，SName 属性并不是 Station 表的主码，运行结果出错，如下图 3.3 所示。

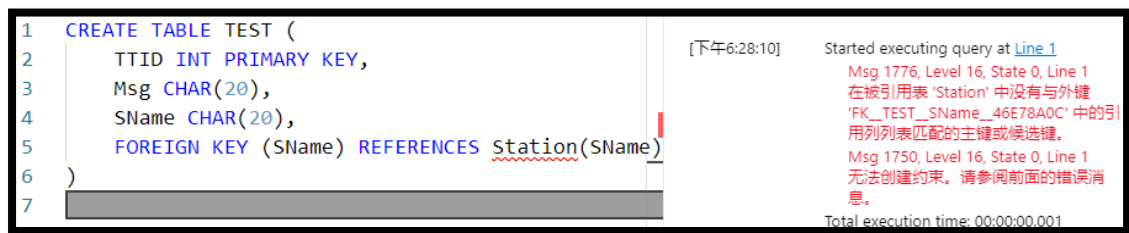


图 3.3 观察性实验结果

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。

使用 Excel 公式进行数据录入，生成下列 Excel 表：

名称	修改日期	类型	大小
DiagnoseRecord诊断表.xls	2020/4/28 17:16	Microsoft Excel ...	4,450 KB
Passenger乘客表.xls	2020/4/28 16:32	Microsoft Excel ...	4,565 KB
Station车站表.xls	2020/4/28 16:21	Microsoft Excel ...	749 KB
TakeTrainRecord乘车记录表.xls	2020/4/28 16:30	Microsoft Excel ...	14,117 KB
TrainContactor乘客紧密接触者表.xls	2020/4/29 13:10	Microsoft Excel ...	6,256 KB
TrainPass车程表.xls	2020/4/28 17:21	Microsoft Excel ...	2,801 KB
Train车次表.xls	2020/4/28 16:19	Microsoft Excel ...	990 KB

图 3.4 导入数据使用的 Excel 表

这些 Excel 表由老师所提供的仿真数据整理得到，部分缺失数据使用了 Excel 的公式随机生成。

由于篇幅限制，下面以 TakeTrainRecord 乘车记录表为例进行说明。
该表由 Excel 自动生成的数据如下图 3.5 所示：

	A	B	C	D	E	F	G	H	I
	RID	PCardID	TID	SStationID	AStationID	CarrigeID	SeatRow	SeatNo	SStatus
1									
2	1	150621196302132688	344	1597	3860	15	6	A	2
3	2	450502196002231154	181	1646	1621	4	7	B	2
4	3	33078219320303217X	36	1597	1556	5	7	E	2
5	4	130322194706301350	300	404	1597	2	17	C	2
6	5	330303197104161700	402	403	1599				2
7	6	350629193612252866	127	2020	1	13	20	A	1
8	7	450602195012251205	119	1615	1634	14	8	B	2
9	8	420114194701051258	342	1597	129	8	13	A	2
10	9	330703195409110480	326	1597	129	13	12	B	2
11	10	430811197706094614	45	1615	3997				1
12	11	532525199304154457	5	1599	1621	7	8	C	0
13	12	632701193502044917	47	118	1344	9	10	F	0
14	13	330402199505011250	324	1597	129	16	15	F	0
15	14	469022196910052715	114	1621	1646	2	16	F	0
16	15	340882193101162173	131	1599	1540				1
17	16	321084196806284074	388	1634	3016	14	20	A	1
18	17	61032319980201497X	323	1615	129	8	4	F	0
19	18	310104201409143192	144	1597	1540	6	14	A	1
20	19	510322201402213873	51	1615	3365	12	14	F	0
21	20	331082195404103159	271	1597	404				0
22	21	44130319620307441X	445	1599	129	12	18	F	0
23	22	130984195708150126	168	1597	3415	6	5	E	0
24	23	370522194808092081	387	1637	3388	12	20	C	0
25	24	621224197812022513	393	1599	2014	7	17	F	2
26	25	341002197010160117	273	1597	404				2
27	26	440705200201301351	338	1597	129	13	16	A	1
28	27	610502200704182472	197	1597	471	6	16	C	0
29	28	130131200305162576	400	3414	161	5	6	E	0
30	29	140826201006023553	138	1615	1661	6	2	F	1

图 3.5 自动生成 Excel 乘车记录表数据

图 3.5 中的数据依据下面的分析得到：

- ①RID 为流水号，使用 Excel 中的行号即可唯一标识，因此在单元格 An 使用公式 ROW(An)-1 即可。
- ②PCardID 为乘客身份证号，可从乘客表的身份证号信息列使用 INDEX 函数获取值，对列长度以内的数字使用 RANDBETWEEN 函数进行随机，得到随机的乘客身份证号。
- ③TID 为列车编号，根据车次表的信息可知列车编号在 1 至 447 之间，因此只需使用 RANDBEWTWEEN(1,447)即可。
- ④SStationID 和 AStationID 分别是始发站车站号和终点站车站号，可以根据列车编号 TID 在车程表中查询对应的车站号，因此使用 VLOOKUP 函数，以 TID 列信息为索引，将车程表复制到该表的 Sheet2 中，在 Sheet2 中进行匹配查找，将匹配的 SStationID 和 AStationID 填入表中，具体公式形如 =VLOOKUP(C2,Sheet2!L:L,3,TRUE)。
- ⑤CarrigeID 表示车厢号，可以使用 RANDBETWEEN(1,16)进行随机，但由于要考虑无座票的情况，因此需要设置一个 IF 条件，设置为每 5 个乘车记录出

现 1 次无票乘车记录，具体公式形如=IF(MOD(A2,5)<>0,RANDBETWEEN(1,16),"")。

⑥SeatRow 与 CarriageID 类似，随机范围改为 1 至 20。

⑦SeatNo 也是类似的处理，不同的是 SeatNo 的范围是字母” A” \” B” \” C” \” E” \” F”。因此需要使用 INDEX 函数，将这五个字母用大括号括起来表示集合，然后再用 1 至 5 的随机数随机产生，具体公式形如=IF(F2<>"",INDEX({"A","B","C","E","F"},RANDBETWEEN(1,5)),"")。

生成表之后进行数据导入，右键点击数据库，选择任务->导入数据，选好 Excel 表和数据库的映射关系之后执行导入任务，如下图 3.6 所示：

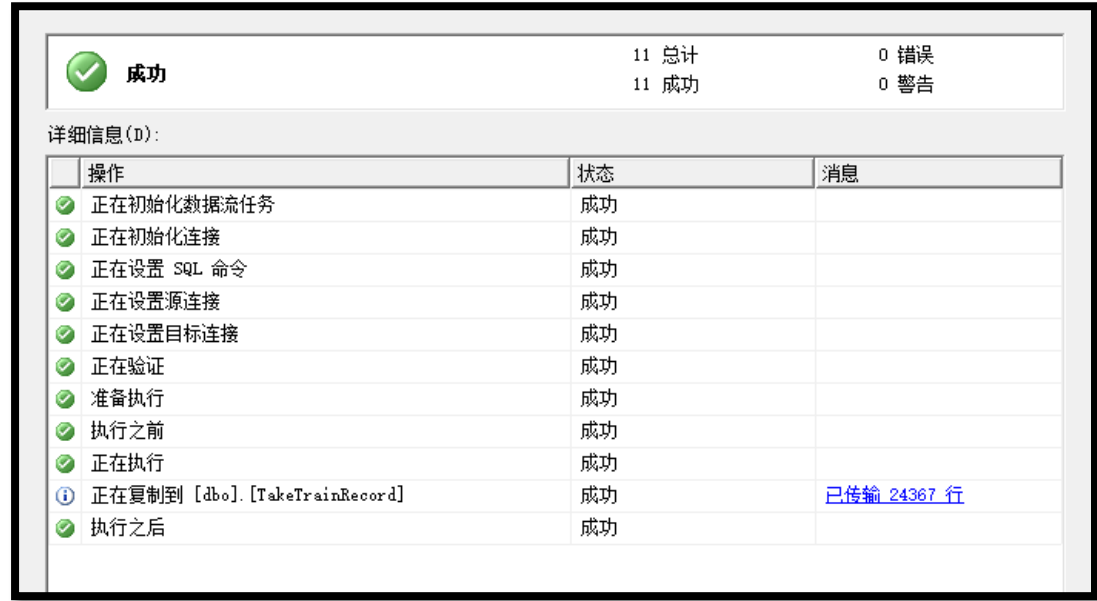


图 3.6 成功导入数据

其余各表与上述 TakeTrainRecord 乘车记录表的处理方法类似，按上述流程即可以完成数据库的数据准备过程。

3.2.2 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

①对乘车记录表的增操作 SQL 语句如图 3.7 所示：

```
1  -- 增
2  INSERT INTO TakeTrainRecord
3  VALUES
4  (24370,'360123199904112914',230,1597,1533,9,3,'E',1),
5  (24371,'360123199904112914',2,1533,231,7,13,'C',1)
```

图 3.7 对乘车记录表的增操作

②对乘车记录表的删操作 SQL 语句如图 3.8 所示：


```

1  -- 删
2  DELETE FROM TakeTrainRecord
3  WHERE RID = 24367

```

图 3.8 对乘车记录表的删操作

③对乘车记录表的改操作 SQL 语句如图 3.9 所示：

```

1  -- 改
2  UPDATE TakeTrainRecord
3  SET TID = 78
4  WHERE RID = 24371

```

图 3.9 对乘车记录表的改操作

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。

思路是将乘车记录表和车站表做一个等值连接，取出发站的 SID 作为等值条件，然后取城市名为武汉的记录投影到 WH_TakeTrainRecord 表中。

SQL 语句如下图 3.10 所示：

```

1  SELECT RID,PCardID,TID,SStationID,ASTationID,CarrigeID,SeatRow,SeatNo,SStatus
2  INTO WH_TakeTrainRecord
3  FROM TakeTrainRecord tr,Station s
4  WHERE tr.SStationID = s.SID
5  AND CityName = '武汉'

```

图 3.10 批处理操作 SQL 语句

然后查询 WH_TakeTrainRecord 表的前 1000 行，操作结果如下图 3.11 所示：

RID	PCardID	TID	SStationID	ASTationID	CarrigeID	SeatRow	SeatNo	SStatus
1	150621196302132688	73	1597	161	6	12	B	0
2	450502196002231154	68	1597	2125	1	13	E	0
3	33078219320303217X	439	1599	3388	3	3	F	1
4	330303197104161700	245	1597	3015	NULL	NULL		2
5	450602195012251205	299	1597	404	3	16	A	1
6	330703195409110480	45	1615	3997	2	9	C	1
7	430811197706094614	42	1615	3898	NULL	NULL		2
8	532525199304154457	82	1597	2020	12	15	A	0
9	330402199505011250	10	1599	1621	8	13	F	1
10	469022196910052715	58	1615	3878	11	3	E	1
11	340882193101162173	67	1597	2127	NULL	NULL		0
12	61032319980201497X	97	1615	3365	12	20	C	0

图 3.11 批处理操作结果

3) 数据导入导出

①数据导出

右键目标数据库，选择任务->导出数据，进入如图 3.12 所示页面：



图 3.12 导出向导

导出目标文件选择 Excel 文件，由上图可见每张表对应 Excel 的一个 Sheet，导出后打开 Excel 文件查看结果，如下图 3.13 所示：

1	DID	PCardID	DDay	DStatus	FDay	
2		1 150621196302132688	2020-01-11	2		
3		2 450502196002231154	2020-01-20	3		
4		3 33078219320303217X	2020-01-22	1	2020-01-19	
5		4 130322194706301350	2020-01-10	1	2020-01-02	
6		5 330303197104161700	2020-01-19	1	2020-01-10	
7		6 350629193612252866	2020-01-18	1	2020-01-17	
8		7 450602195012251205	2020-01-25	3		
9		8 420114194701051258	2020-01-30	3		
10		9 330703195409110480	2020-01-11	2		
11		10 430811197706094614	2020-01-23	1	2020-01-18	
12		11 532525199304154457	2020-01-27	3		
13		12 632701193502044917	2020-01-15	2		
14		13 330402199505011250	2020-01-24	3		
15		14 469022196910052715	2020-01-29	3		
16		15 340882193101162173	2020-01-24	1	2020-01-19	
17		16 321084196806284074	2020-01-26	2		
18		17 61032319980201497X	2020-01-23	3		
19		18 310104201409143192	2020-01-13	3		
20		19 510322201402213873	2020-01-18	2		
21		20 331082195404103159	2020-01-23	2		
22		21 44130319620307441X	2020-01-16	3		
23		22 130984195708150126	2020-01-12	2		
24		23 370522194808092081	2020-01-12	2		
25		24 621224197812022513	2020-01-26	1	2020-01-23	
26		25 341002197010160117	2020-01-25	3		
27		26 440705200201301351	2020-01-27	1	2020-01-15	
28		27 610502200704182472	2020-01-20	3		
29		28 130131200305162576	2020-01-29	2		
30		29 140826201006023553	2020-01-11	1	2020-01-01	
31		30 34040319880709303X	2020-01-23	2		
32		31 360404194207120139	2020-01-18	2		
33		32 540127197505263344	2020-01-23	3		
34		33 500240200806252765	2020-01-28	3		
35		34 360681195104064259	2020-01-12	2		
36		35 533124199004161279	2020-01-18	3		

图 3.13 导出结果

②数据导入

右键目标数据库，选择任务->导入数据，进入如图 3.14 所示页面：

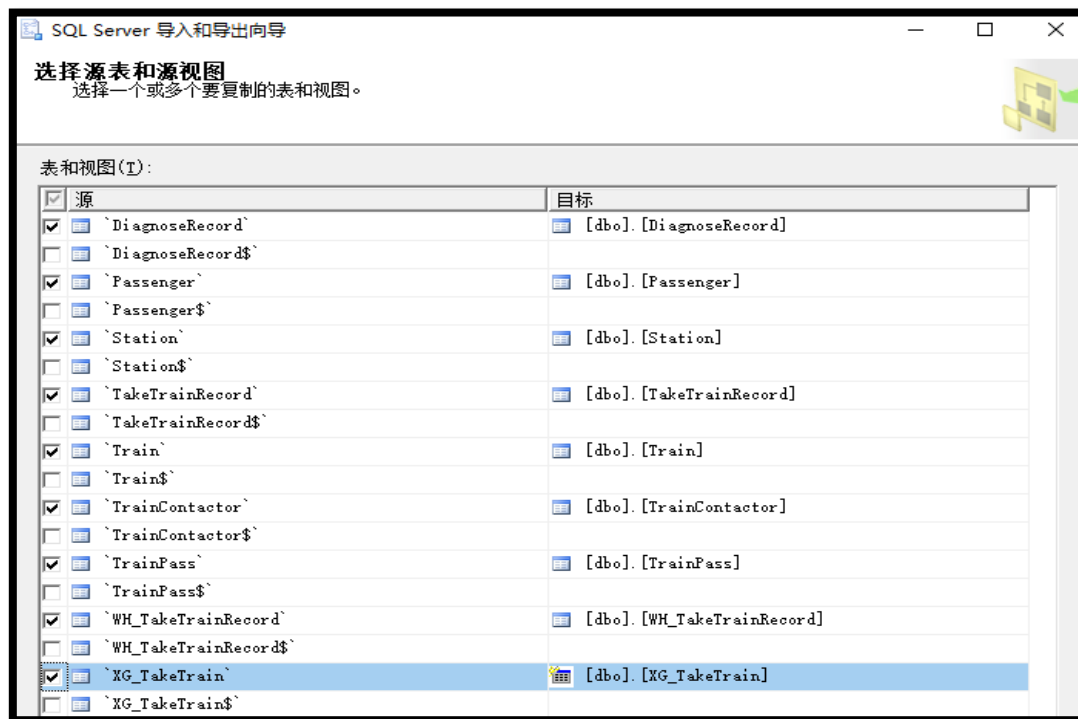


图 3.14 导入向导

导入目标文件选择 Excel 文件， Excel 的 Sheet 与数据库的表可以建立上图的映射关系，导入后进入 Management Studio 查看结果，如下图 3.15 所示：

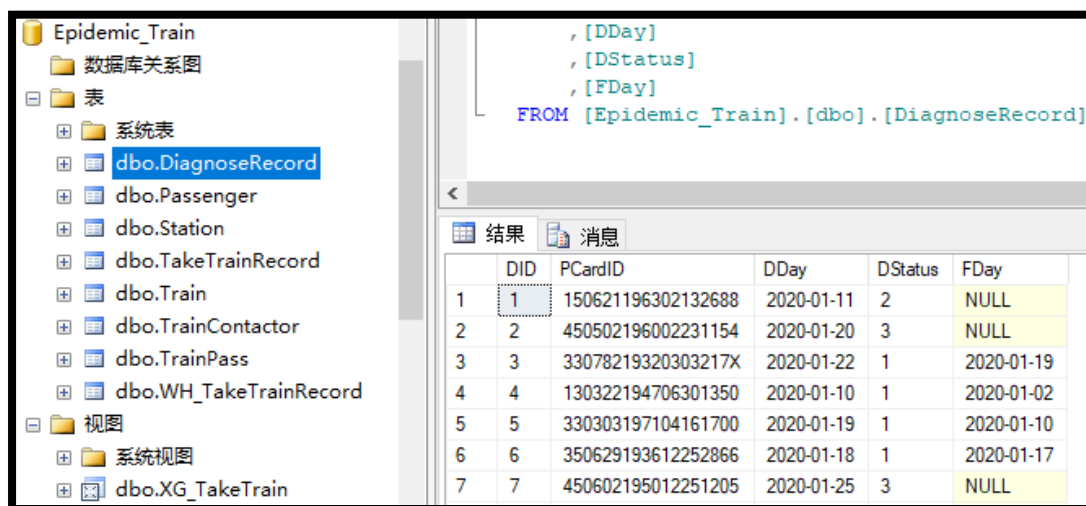


图 3.15 导入结果

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。

建立一个 NoPrimaryKey 关系，并插入 3 条重复数据，如下图 3.16 所示：

```

1 CREATE TABLE NoPrimaryKey(
2     ID INT,
3     SCORE INT
4 )
5
6 INSERT INTO NoPrimaryKey
7 VALUES
8 (666,100),
9 (666,100),
10 (666,100)

```

图 3.16 建立关系并插入重复数据 SQL 语句

①修改

在 Management Studio 中尝试修改表中的数据，操作结果如图 3.17 所示：



图 3.17 修改数据的操作结果

②删除

在 Management Studio 中尝试删除表中的数据，操作结果如图 3.18 所示：

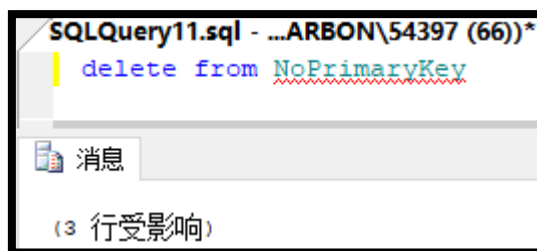


图 3.18 删除数据的操作结果

综上所述，删除可以正常进行，但可能由于该表不具有实体完整性，无法对表进行修改。

5) 创建视图

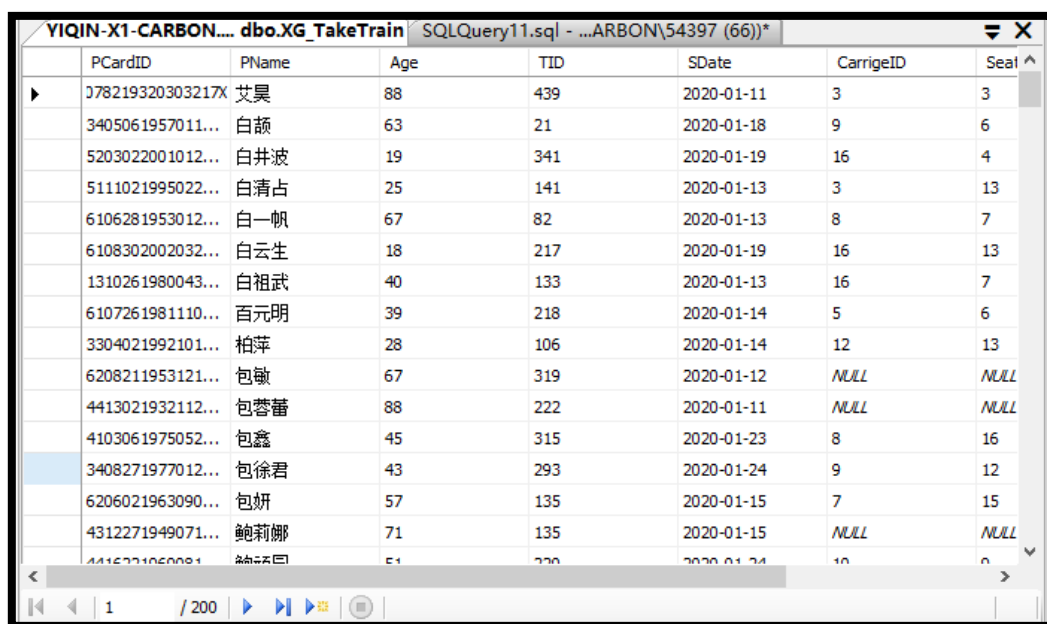
创建一个新冠确诊病人的乘火车记录视图，其中的属性包括：身份证号、姓名、年龄、乘坐列车编号、发车日期、车厢号，席位排号，席位编号。按身份证号升序排序，如果身份证号一致，按发车日期降序排序（注意，如果病人买了票但是没坐车，不计入在内）。

创建该视图的 SQL 语句如下图 3.19 所示：

```
1 CREATE VIEW XG_TakeTrain AS
2 SELECT TOP 100 PERCENT dr.PCardID,PName,Age,tr.TID,SDate,CarrigeID,SeatRow,SeatNo
3 FROM DiagnoseRecord dr,Passenger p,TakeTrainRecord tr,Train t
4 WHERE dr.DStatus = 1
5 AND dr.PCardID = p.PCardID
6 AND dr.PCardID = tr.PCardID
7 AND tr.TID = t.TID
8 AND SStatus = 1
9 ORDER BY dr.PCardID,SDate DESC
```

图 3.19 创建视图的 SQL 语句

创建后在 Management Studio 中查看结果，如下图 3.20 所示：



PCardID	PName	Age	TID	SDate	CarrigeID	Seat
078219320303217X	艾昊	88	439	2020-01-11	3	3
3405061957011...	白颜	63	21	2020-01-18	9	6
5203022001012...	白井波	19	341	2020-01-19	16	4
5111021995022...	白清占	25	141	2020-01-13	3	13
6106281953012...	白一帆	67	82	2020-01-13	8	7
6108302002032...	白云生	18	217	2020-01-19	16	13
1310261980043...	白祖武	40	133	2020-01-13	16	7
6107261981110...	百元明	39	218	2020-01-14	5	6
3304021992101...	柏萍	28	106	2020-01-14	12	13
6208211953121...	包敏	67	319	2020-01-12	NULL	NULL
4413021932112...	包蓉蕾	88	222	2020-01-11	NULL	NULL
4103061975052...	包鑫	45	315	2020-01-23	8	16
3408271977012...	包徐君	43	293	2020-01-24	9	12
6206021963090...	包妍	57	135	2020-01-15	7	15
4312271949071...	鲍莉娜	71	135	2020-01-15	NULL	NULL
4415271956008...	鲍永国	51	220	2020-01-24	10	0

图 3.20 视图创建结果

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”，注意其中：接触日期为乘车日期。

根据要求，设计 SQL 语句如下，定义名为 MyTrigger1 的触发器实现上述功能。思路是先使用查询语句查询该患者是否满足发病前 14 天内有乘车记录这一条件，若满足则根据患者的身份证号查询其乘车记录，将乘车记录中出现过的同排及前后排乘客信息查询出来，使用 INSERT 语句加入“乘客紧密接触者表”中。

具体 SQL 语句如下图 3.21 所示：

```

1 CREATE TRIGGER MyTrigger1
2 ON DiagnoseRecord
3 AFTER INSERT AS
4 IF( SELECT COUNT(*)
5     FROM inserted,TakeTrainRecord tr,TrainPass tp
6     WHERE tr.PCardID = inserted.PCardID
7     AND tr.TID = tp.TID
8     AND tr.SStationID = tp.SID
9     AND inserted.DStatus = 1
10    AND SStatus = 1
11    AND DATEDIFF(DD,tp.STime,FDay) <= 14 ) > 0
12 BEGIN
13 PRINT '插入一个新冠患者，同排及前后排加入乘客紧密接触者表'
14 INSERT INTO TrainContactor
15 SELECT SDate CDate,tr.PCardID CCardID,'3' DStatus,TEMP.PCardID PCardID
16 FROM TakeTrainRecord tr,Train t,
17     (SELECT tr1.TID,CarrigeID,SeatRow,inserted.PCardID
18     FROM inserted,TakeTrainRecord tr1,TrainPass tp1
19     WHERE tr1.PCardID = inserted.PCardID
20     AND tr1.TID = tp1.TID
21     AND tr1.SStationID = tp1.SID
22     AND SStatus = 1
23     AND DATEDIFF(DD,tp1.STime,FDay) <= 14 ) TEMP
24 WHERE tr.TID = t.TID
25 AND tr.TID = TEMP.TID
26 AND tr.PCardID <> TEMP.PCardID
27 AND SStatus = 1
28 AND tr.CarrigeID = TEMP.CarrigeID
29 AND ABS(tr.SeatRow-TEMP.SeatRow) <= 1
30 END

```

图 3.21 触发器 1 的 SQL 语句

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”。

根据要求，设计 SQL 语句如下，定义名为 MyTrigger2 的触发器实现上述功能。思路是先使用查询语句在乘客紧密接触者表中找到满足与新插入数据的身份证号相等的数据项，若数目大于 0 说明该患者为紧密接触者，然后根据患者的身份证号，使用 UPDATE 语句更新其状态为 1，表示确诊。

具体 SQL 语句如下图 3.22 所示：

```

1 CREATE TRIGGER MyTrigger2
2 ON DiagnoseRecord
3 AFTER INSERT
4 AS
5 IF(
6     SELECT COUNT(*)
7     FROM inserted,TrainContactor tc
8     WHERE inserted.PCardID = tc.CCardID
9     AND inserted.DStatus = 1
10    ) > 0
11 BEGIN
12 PRINT '在乘客紧密接触者表中修改其状态为1，确诊'
13 UPDATE TrainContactor
14 SET DStatus = 1
15 WHERE CCardID IN (SELECT PCardID FROM inserted)
16 END

```

图 3.22 触发器 2 的 SQL 语句

向乘客紧密接触者表中插入如下图所示的一条数据，该数据可以触发上述的触发器，插入数据的 SQL 语句以及触发器测试的结果如下图 3.23 所示：



图 3.23 触发器测试结果

3.2.3 查询

分别使用一条 SQL 语句完成下列查询要求：

1) 查询确诊者“张三”的在发病前 14 天内的乘车记录

代码和查询结果如下图 3.24 所示：

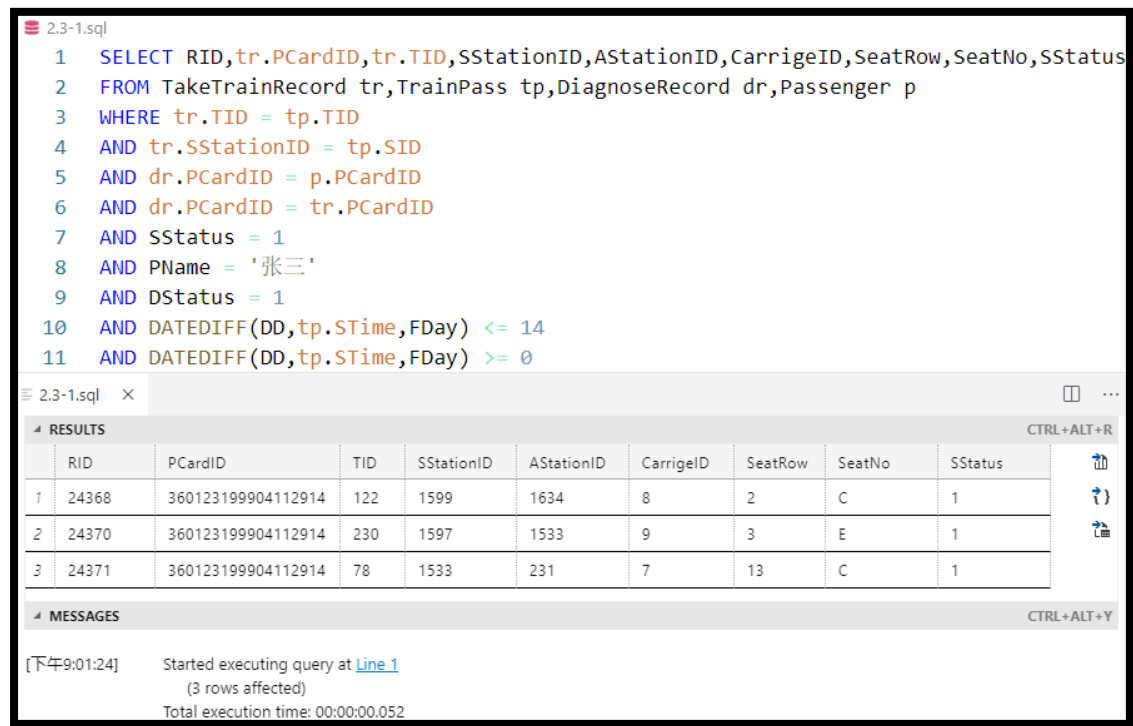


图 3.24 查询结果 1

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

代码和查询结果如下图 3.25 所示：

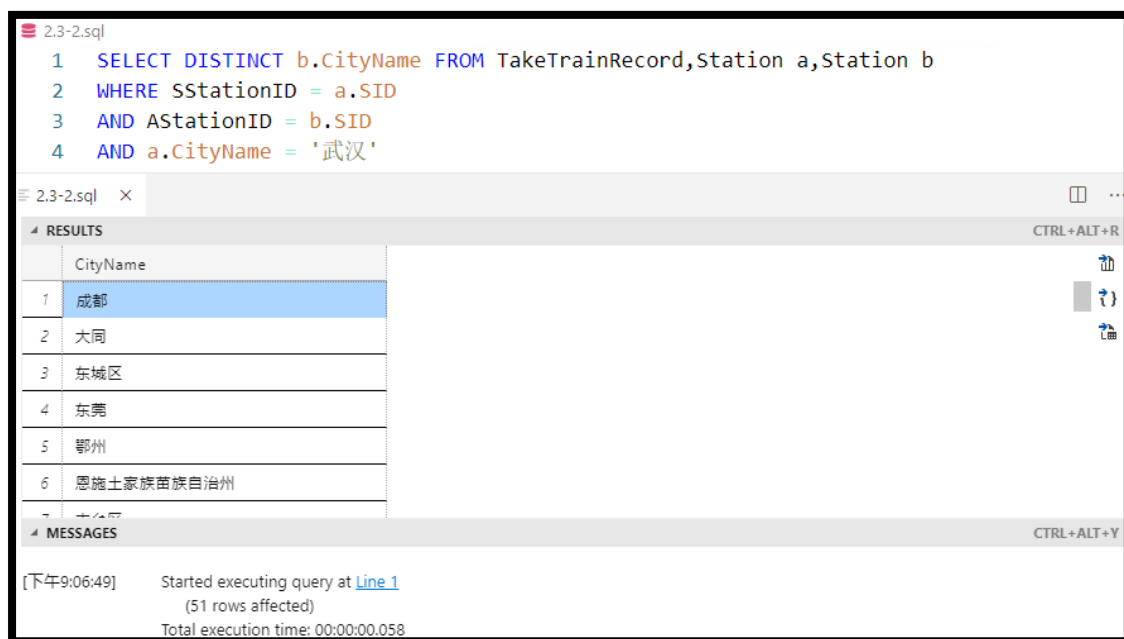


图 3.25 查询结果 2

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

代码和查询结果如下图 3.26 所示：



图 3.26 查询结果 3

4) 查询“2020-01-22”从“武汉”发出的所有列车

代码和查询结果如下图 3.27 所示：

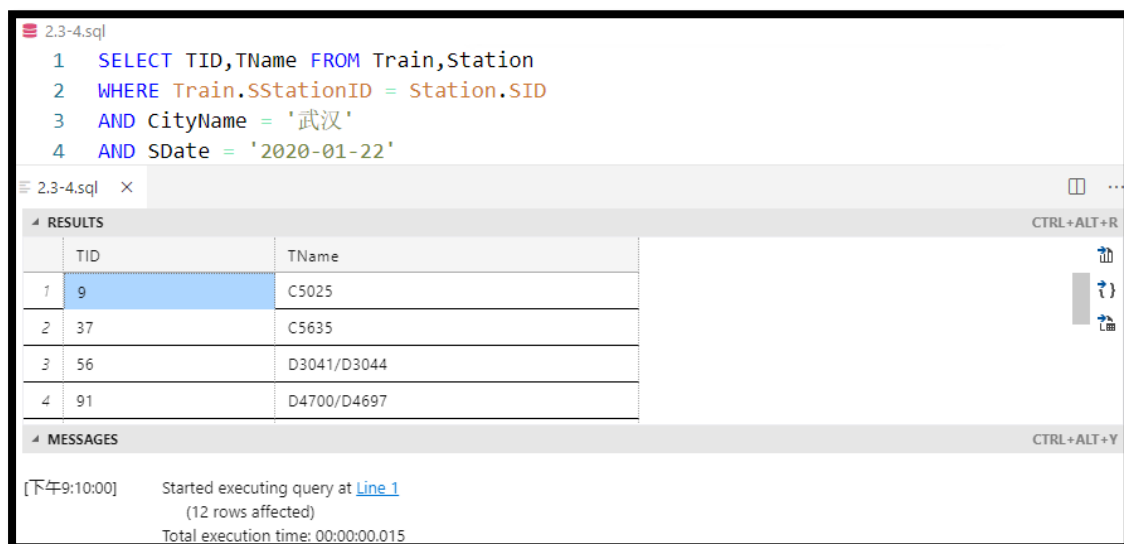


图 3.27 查询结果 4

5) 查询“2020-01-22”途经“武汉”的所有列车
代码和查询结果如下图 3.28 所示:

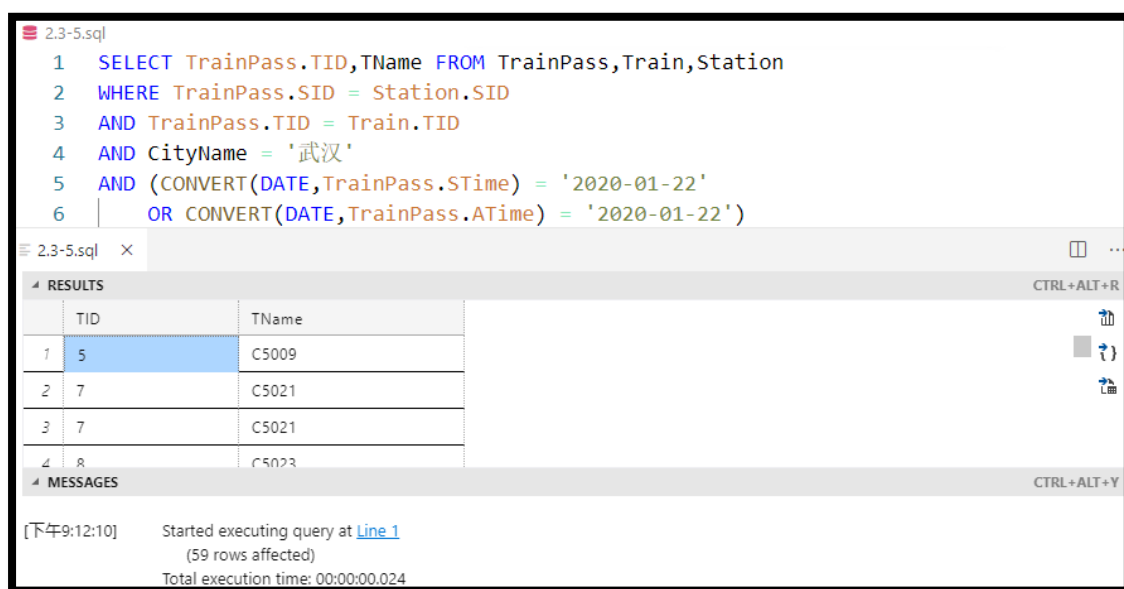


图 3.28 查询结果 5

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

代码和查询结果如下图 3.29 所示:

2.3-6.sql

```

1 SELECT PCardID,sb.CityName,CONVERT(DATE,tpb.ATime) ArriveDate
2 FROM TakeTrainRecord tr,TrainPass tpa,TrainPass tpb,Station sa,Station sb
3 WHERE tr.TID = tpa.TID
4 AND tr.SStationID = tpa.SID
5 AND tr.TID = tpb.TID
6 AND tr.AStationID = tpb.SID
7 AND tr.SStationID = sa.SID
8 AND tr.AStationID = sb.SID
9 AND SStatus = 1
10 AND sa.CityName = '武汉'
11 AND CONVERT(DATE,tpa.STime) = '2020-01-22'

```

RESULTS

	PCardID	CityName	ArriveDate
1	330402199505011250	咸宁	2020-01-10
2	510322201402213873	闵行区	2020-01-22
3	610830200203204137	广州	2020-01-20
4	131026198004303237	宜昌	2020-01-25

MESSAGES

[下午9:13:30] Started executing query at [Line 1](#)
(306 rows affected)
Total execution time: 00:00:00.063

图 3.29 查询结果 6

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

代码和查询结果如下图 3.30 所示:

2.3-7.sql

```

1 SELECT sb.CityName,COUNT(PCardID) Num
2 FROM TakeTrainRecord tr,TrainPass tp,Station sa,Station sb
3 WHERE tr.TID = tp.TID
4 AND tr.SStationID = tp.SID
5 AND tr.SStationID = sa.SID
6 AND tr.AStationID = sb.SID
7 AND SStatus = 1
8 AND sa.CityName = '武汉'
9 AND CONVERT(DATE,tp.STime) = '2020-01-22'
10 GROUP BY sb.CityName

```

RESULTS

	CityName	Num
1	成都	21
2	鄂州	1
3	福州	18
4	广州	48

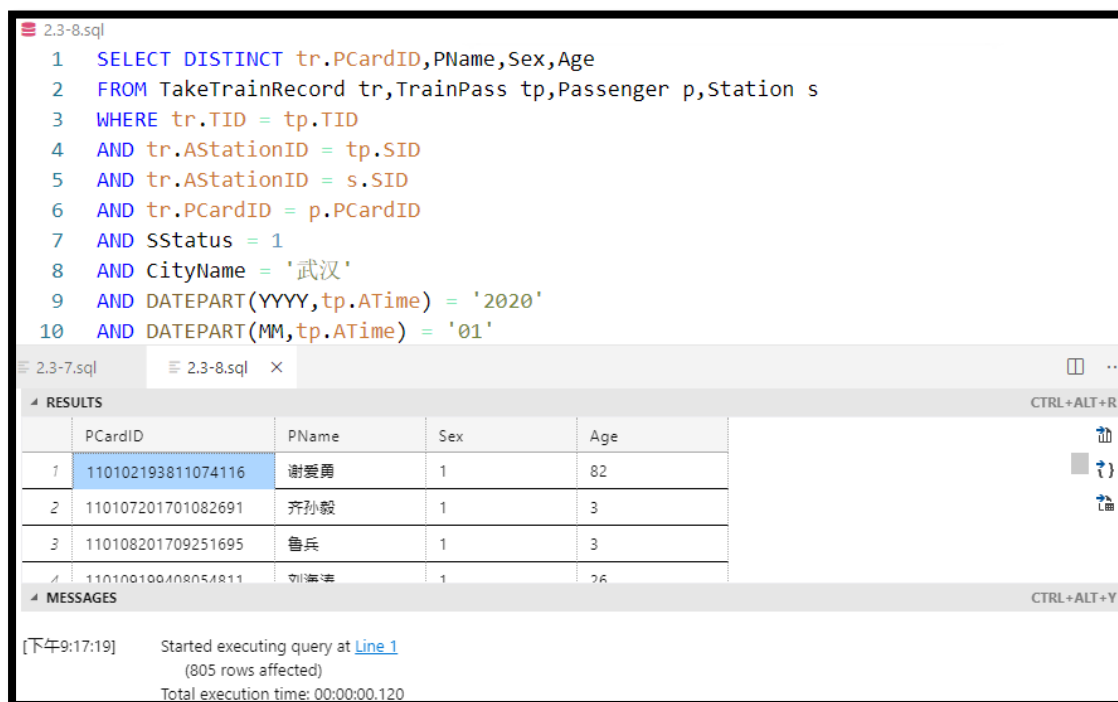
MESSAGES

[下午9:14:33] Started executing query at [Line 1](#)
(12 rows affected)
Total execution time: 00:00:00.038

图 3.30 查询结果 7

8) 查询 2020 年 1 月到达武汉的所有人员

代码和查询结果如下图 3.31 所示:



```
2.3-8.sql
1 SELECT DISTINCT tr.PCardID,PName,Sex,Age
2 FROM TakeTrainRecord tr,TrainPass tp,Passenger p,Station s
3 WHERE tr.TID = tp.TID
4 AND tr.AStationID = tp.SID
5 AND tr.AStationID = s.SID
6 AND tr.PCardID = p.PCardID
7 AND SStatus = 1
8 AND CityName = '武汉'
9 AND DATEPART(YYYY,tp.ATime) = '2020'
10 AND DATEPART(MM,tp.ATime) = '01'
```

	PCardID	PName	Sex	Age
1	110102193811074116	谢爱勇	1	82
2	110107201701082691	齐孙毅	1	3
3	110108201709251695	鲁兵	1	3
4	110100100408054811	刘海强	1	26

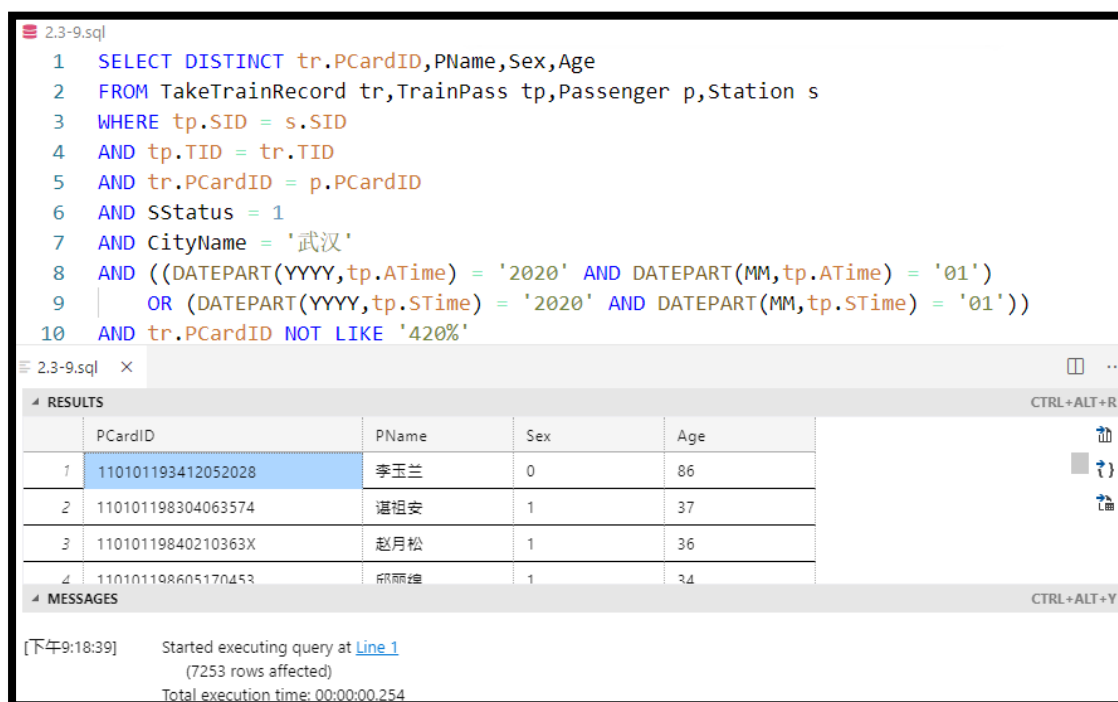
MESSAGES

[下午9:17:19] Started executing query at Line 1
(805 rows affected)
Total execution time: 00:00:00.120

图 3.31 查询结果 8

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)

代码和查询结果如下图 3.32 所示:



```
2.3-9.sql
1 SELECT DISTINCT tr.PCardID,PName,Sex,Age
2 FROM TakeTrainRecord tr,TrainPass tp,Passenger p,Station s
3 WHERE tp.SID = s.SID
4 AND tp.TID = tr.TID
5 AND tr.PCardID = p.PCardID
6 AND SStatus = 1
7 AND CityName = '武汉'
8 AND ((DATEPART(YYYY,tp.ATime) = '2020' AND DATEPART(MM,tp.ATime) = '01')
9 OR (DATEPART(YYYY,tp.STime) = '2020' AND DATEPART(MM,tp.STime) = '01'))
10 AND tr.PCardID NOT LIKE '420%'
```

	PCardID	PName	Sex	Age
1	110101193412052028	李玉兰	0	86
2	110101198304063574	湛祖安	1	37
3	11010119840210363X	赵月松	1	36
4	110101198605170453	郝丽维	1	34

MESSAGES

[下午9:18:39] Started executing query at Line 1
(7253 rows affected)
Total execution time: 00:00:00.254

图 3.32 查询结果 9

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

代码和查询结果如下图 3.33 所示：

```

1 SELECT TName, SDate, CarrigeID
2 FROM TakeTrainRecord tr, DiagnoseRecord dr, Train t
3 WHERE tr.PCardID = dr.PCardID
4       AND tr.TID = t.TID
5       AND DStatus = 1
6       AND SStatus = 1
7       AND CarrigeID IS NOT NULL
8 GROUP BY tr.TID, TName, SDate, CarrigeID
9 HAVING COUNT(RID) >= 3
  
```

	TName	SDate	CarrigeID
1	D5253	2020-01-20	1
2	G4678	2020-01-19	2
3	D5208/D5205	2020-01-18	3

MESSAGES: [下午9:34:04] Started executing query at Line 1 (22 rows affected) Total execution time: 00:00:00.061

图 3.33 查询结果 11

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
代码和查询结果如下图 3.34 所示：

```

1 SELECT tc.PCardID, PName, CDate
2 FROM TrainContactor tc, Passenger p
3 WHERE tc.PCardID = p.PCardID
4       AND tc.PCardID NOT IN (
5       SELECT PCardID
6       FROM TrainContactor
7       WHERE tc.DStatus <> 3)
  
```

	PCardID	PName	CDate
1	110102197108130057	付强大	2020-01-11
2	110102201307090958	张兆钢	2020-01-17
3	11010520110602032X	莫梦莹	2020-01-10
4	110106194312312933	郭宝	2020-01-27

MESSAGES: [下午9:35:33] Started executing query at Line 1 (2709 rows affected) Total execution time: 00:00:00.167

图 3.34 查询结果 12

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

代码和查询结果如下图 3.35 所示：

```
2.3-13.sql
1 SELECT DISTINCT TName, CityName
2 FROM Train, Station
3 WHERE Train.AStationID = Station.SID
4 AND CityName IN ('北京','上海','广州')
```

	TName	CityName
1	G1101	广州
2	G1103	广州

MESSAGES

[下午9:36:31] Started executing query at Line 1
(43 rows affected)
Total execution time: 00:00:00.025

图 3.35 查询结果 13

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

代码和查询结果如下图 3.36 所示：

```
2.3-14.sql
1 SELECT tr.PCardID, TEMP.TID
2 FROM TakeTrainRecord tr, TrainPass tp,
3 -- (SELECT PCardID,tr1.TID,STime,tr1.AStationID
4 (SELECT PCardID, tr1.TID, tr1.AStationID
5 FROM TakeTrainRecord tr1, TrainPass tp1, Station s1
6 WHERE tr1.TID = tp1.TID
7 AND tr1.SStationID = tp1.SID
8 AND tr1.SStationID = s1.SID
9 AND SStatus = 1
10 AND SName = '武汉'
11 AND CONVERT(DATE,STime) = '2020-01-22' ) TEMP
12 WHERE tr.PCardID = TEMP.PCardID
13 AND tr.SStationID = tp.SID
14 AND tr.TID = tp.TID
15 AND TEMP.AStationID = tr.SStationID -- 接续换乘
16 AND CONVERT(DATE,tp.STime) = '2020-01-22' -- 同一天
17 -- AND tp.STime > TEMP.STime -- 换乘时间在后
18 AND SStatus = 1
19 ORDER BY TEMP.TID DESC, tr.PCardID
```

	PCardID	TID
1	360123199904112914	230

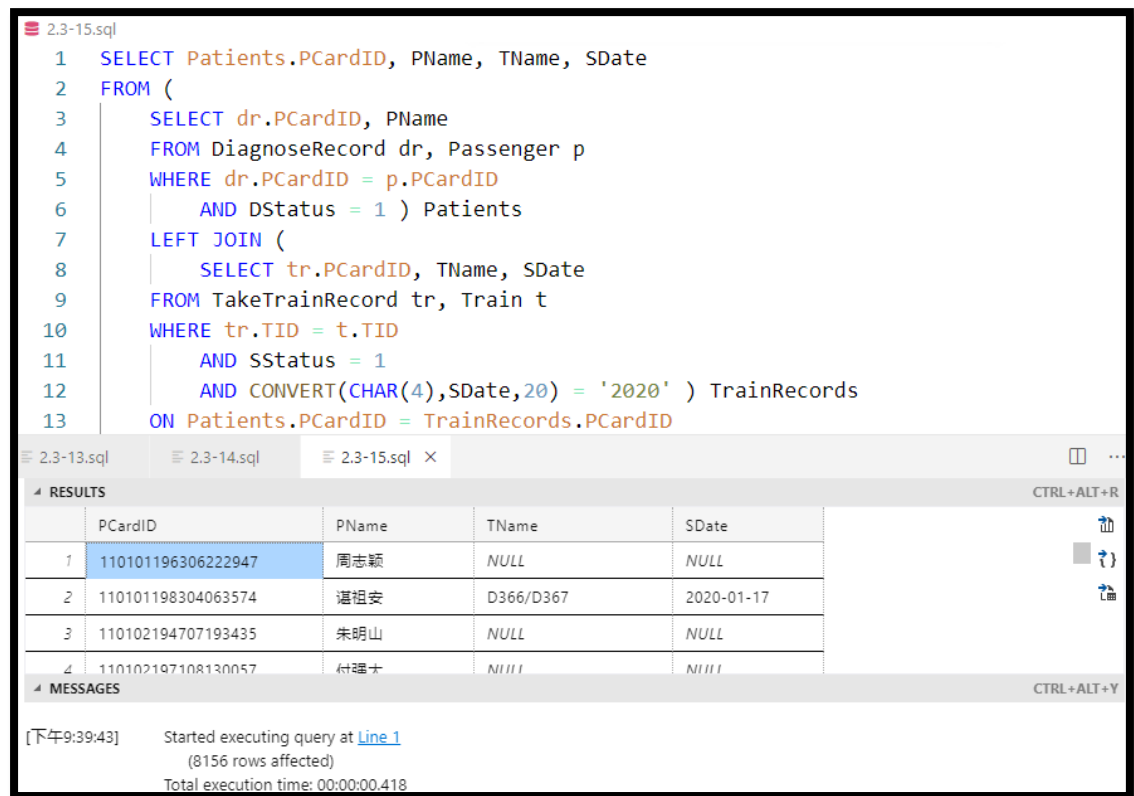
MESSAGES

[下午9:37:39] Started executing query at Line 1
(1 row affected)
Total execution time: 00:00:01.117

图 3.36 查询结果 14

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
代码和查询结果如下图 3.37 所示：



```
2.3-15.sql
1 SELECT Patients.PCardID, PName, TName, SDate
2 FROM (
3     SELECT dr.PCardID, PName
4     FROM DiagnoseRecord dr, Passenger p
5     WHERE dr.PCardID = p.PCardID
6           AND DStatus = 1 ) Patients
7 LEFT JOIN (
8     SELECT tr.PCardID, TName, SDate
9     FROM TakeTrainRecord tr, Train t
10    WHERE tr.TID = t.TID
11          AND SStatus = 1
12          AND CONVERT(CHAR(4),SDate,20) = '2020' ) TrainRecords
13 ON Patients.PCardID = TrainRecords.PCardID
```

	PCardID	PName	TName	SDate
1	110101196306222947	周志颖	NULL	NULL
2	110101198304063574	湛祖安	D366/D367	2020-01-17
3	110102194707193435	朱明山	NULL	NULL
4	110102197108130057	付强士	NULL	NULL

RESULTS

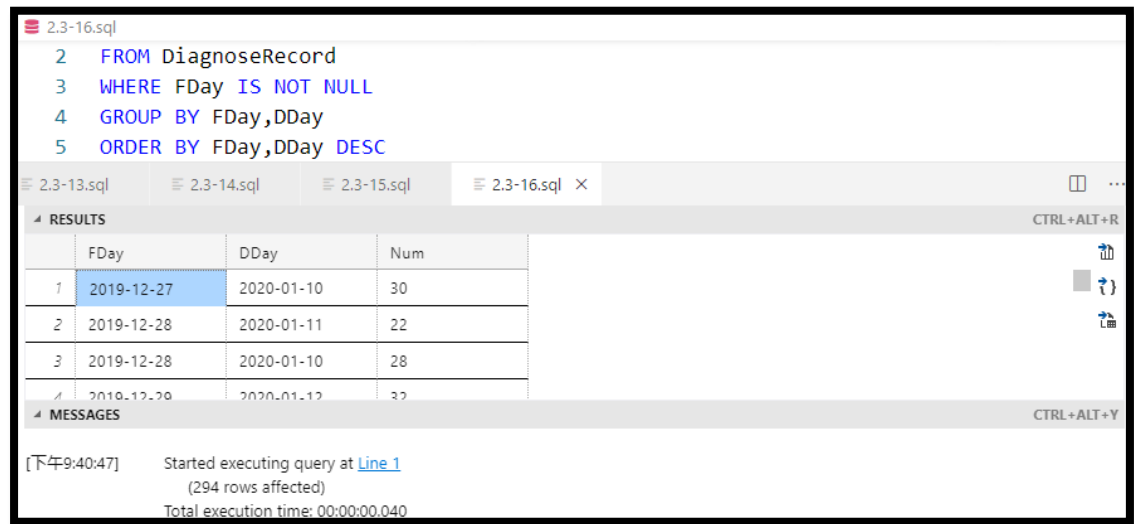
MESSAGES

[下午9:39:43] Started executing query at Line 1
(8156 rows affected)
Total execution time: 00:00:00.418

图 3.37 查询结果 15

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

代码和查询结果如下图 3.38 所示：



```
2.3-16.sql
2 FROM DiagnoseRecord
3 WHERE FDay IS NOT NULL
4 GROUP BY FDay,DDay
5 ORDER BY FDay,DDay DESC
```

	FDay	DDay	Num
1	2019-12-27	2020-01-10	30
2	2019-12-28	2020-01-11	22
3	2019-12-28	2020-01-10	28
4	2019-12-29	2020-01-12	32

RESULTS

MESSAGES

[下午9:40:47] Started executing query at Line 1
(294 rows affected)
Total execution time: 00:00:00.040

图 3.38 查询结果 16

3.3 任务总结

在本次实验中我完成了对数据库进行建表、数据更新以及查询这三个动作的任务，加强了我对 SQL 语句的熟练程度。

在建表过程中，我加深了对关系完整性的理解，设置主码的方式有属性级的（直接在属性后面加 **PRIMARY KEY**），也有表级的（在最后使用 **PRIMARY KEY(KEY1, KEY2, ...)**的形式来设置多属性构成的主码）。外码的设置方式与主码类似，在最后需要加上 **REFERENCES TableName(AttrName)**，此外根据观察性实验，还需要注意的是，外码必须参照的是被参照表的主属性，否则会出现错误。结合课堂知识，主码保证的是关系的实体完整性，外码则保证了关系之间的参照完整性。

在数据更新过程中，我明白了分别使用 **INSERT**、**DELETE** 和 **UPDATE** 语句对数据库实现增删改操作。根据观察性实验，还需要注意的是，如果一个关系在建表时未创建主码，则在插入多个相同数据条目后，无法使用可视化编辑器对表内的属性值进行修改，但删除操作还是仍然可以进行的。准备数据的过程还让我学习了使用 Excel 辅助进行批量数据导入的方法，使用 Excel 中功能强大的函数配合随机数，可以根据老师提供的原始数据生成大量符合数据库中表关系的数据，再使用 SQL Server Management Studio 提供的数据导入工具就可以实现批量数据导入数据库。需要注意的是由于实验中使用的是 SQL Server 2008 R2 版本，只能支持 97-2003 版本的 Excel（即.xls 格式）文件，不支持后续的.xlsx 格式 Excel 文件，因此需要使用兼容模式对 Excel 进行编辑，否则再导入数据时会出现错误。

在 SQL 查询语句的练习中，我学会了 **SELECT** 的各种用法，比如在需要用到多个表中内容时可以使用多表连接，再设置连接条件；在 **FROM** 字句后面除了使用数据库中已有表，还可以使用 **SELECT** 查询得到的新表；在需要根据查询结果进行分组统计时可以使用 **GROUP BY** 子句配合统计量来实现；若要对结果进行排序，可以使用 **ORDER BY** 子句，默认升序，加入 **DESC** 关键字可以降序；若要保持结果中的部分 **NULL** 值，可以使用左/右连接，例如本次实验第 15 个查询任务，就可以使用左连接 **LEFT JOIN** 进行连接查询。

总而言之，本次实验中的 SQL 练习部分充分检验了我使用 SQL 语句的熟练程度，锻炼了我使用 SQL 语句实现数据库操作的能力。

4 综合实践任务

4.1 系统设计目标

本次综合实践任务我选择的主题是实现一个“连锁品牌零售店库存管理系统”，这个想法来自于近期我购买乐高积木的一次经历。

当时我想要购买一款乐高积木，最近的一家连锁乐高积木零售店的店员告诉我，我要的那一款缺货了，但在他们的库存管理系统中可以查询到同城另一家连锁店还有库存。然而店员说他对另一家店的库存只能查询，并不能帮我调货。

根据上面的经历，我希望开发一个 C/S 架构的连锁品牌零售店库存管理系统，它除了能够实现店员身份认证、商品查询、进货、售货、销售统计等基础功能之外，还需要有能够保存缺货记录、在同品牌多个零售店间发出和处理调货申请的功能，可以很好地解决由于当前门店缺货而无法进行调货而造成的顾客流失问题。

4.2 需求分析

4.2.1 功能需求

（1）身份认证功能

要求系统通过验证用户名和密码的方式对身份进行认证，用户名采用员工的工号，店长和店员身份可以通过工号进行区分，只有店长或店员拥有工号，避免非本店人员进入本系统。

（2）店内人员管理功能

店长可以查看和管理本店人员身份信息，店员只能查看和修改自己的部分身份信息（如修改密码）。

（3）货品查询功能

要求提供可以输入货品各个属性的查询表单，并具有店内查询和全局查询两种模式，货品属性可以不填写完整，要求支持模糊查询。

（4）进货功能

要求提供可以输入进货货号、进货数量、货品各属性的进货单，以货号作为货品各属性的唯一标识。若库存中已存在该货号，则以库存中的货品属性为准，货品属性可以省略不填。

（5）售货功能

要求提供可以输入货品各个属性的售货表单，系统能够根据实际库存情况列出满足条件的本店可出售货品，或是列出可调货的其他店的同款货品。要求在店员选择出售时相应减少本店库存并生成售货记录，选择调货时自动生成调货申请，若无库存也无法调货，则自动生成缺货记录。

（6）销售记录统计功能

要求提供销售记录统计功能，并将统计结果用表格形式展示，仅店长有权限查看销售记录统计。

（7）调货申请处理功能

要求提供调货申请的查看和处理功能，其分为两方面。首先对于本店发出的申请，店员和店长都有权限查看和操作，对于被拒绝的申请要求提供删除操作，对于被接受的申请要求提供确认入库操作，对于处于等待的申请要求提供取消申请的操作。然后对于本店收到的申请，只有店长有权限操作，要求提供同意和拒绝两种操作。

4.2.2 性能需求

（1）数据量需求

要求数据库至少能够存储 200 家连锁店每家至少 1000 件货品的信息。

（2）并发能力需求

要求数据库至少支持 400 个用户的并发访问能力。

（3）响应速度要求

要求用户对数据库完成一次访问的响应时间不大于 5 秒。

4.2.3 数据完整性需求

（1）实体完整性

- ①要求零售店的门店号唯一且非空；
- ②要求员工的工号唯一且非空；
- ③要求乐高积木的货号唯一且非空；
- ④要求库存的货号与门店号的组合唯一且非空；
- ⑤要求缺货记录的缺货编号唯一且非空；
- ⑥要求销售记录的销售编号唯一且非空；
- ⑦要求调货记录的调货编号唯一且非空。

（2）参照完整性

- ①要求员工所属门店号必须存在；
- ②要求库存的货号和所属门店号必须存在；
- ③要求缺货记录的货号和操作员工的工号必须存在；
- ④要求销售记录的货号和操作员工的工号必须存在；
- ⑤要求调货记录的货号、申请店的店号以及被申请店的店号必须存在。

（3）用户定义的完整性

- ①要求员工类别取值范围为 0 或 1；
- ②要求乐高积木的适合年龄取值在 0-100 之间；
- ③要求门店名和门店所在地均不能为空；
- ④要求员工的类别和密码均不能为空；
- ⑤要求库存数量不能为空；
- ⑥要求乐高积木的品名、单价和折扣均不能为空；
- ⑦要求缺货记录的缺货数量不能为空；
- ⑧要求销售记录的售货数量、合计收款不能为空；
- ⑨要求调货记录的数量、状态不能为空；
- ⑩要求所有外码不能为空。

4.2.4 数据流图

数据流图如下图 4.1 所示：

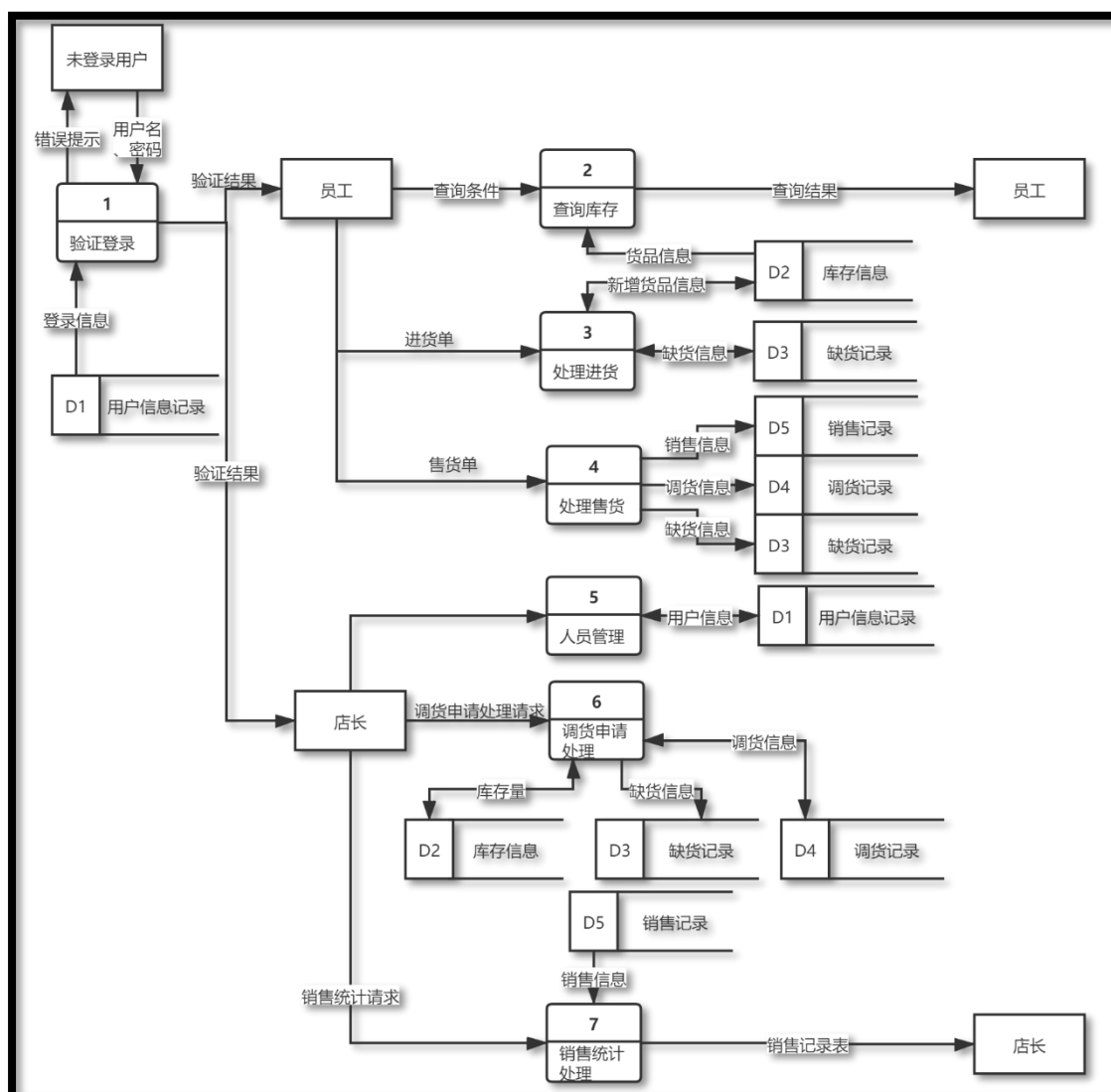


图 4.1 数据流图

4.2.5 数据字典

数据字典如下表 4.1 至 4.7 所示：

表 4.1 门店 Shop 表

名称	代码	数据类型	注释
门店号	SID	int	主键
门店名	SName	varchar(128)	
门店所在地	CityName	varchar(128)	

表 4.2 乐高积木 Juggle 表

名称	代码	数据类型	注释
货号	JID	int	主键
货品名称	JName	varchar(128)	
含积木块数	JNum	int	
适合年龄	JAge	smallint	
单价	Price	decimal(8,2)	
折扣	Discount	decimal(3,2)	

表 4.3 员工 Employee 表

名称	代码	数据类型	注释
工号	EID	int	主键
门店号	SID	int	外键
员工类别	EType	char(1)	
密码	PWD	varchar(32)	

表 4.4 售货记录 Market 表

名称	代码	数据类型	注释
售货编号	MID	int	主键
工号	EID	int	外键
货号	JID	int	外键
售货数量	MNum	int	
合计收款	TotalPrice	decimal(8,2)	
售货时间	MDate	datetime	

表 4.5 缺货记录 Lack 表

名称	代码	数据类型	注释
缺货编号	LID	int	主键
工号	EID	int	外键
货号	JID	int	外键
缺货数量	LNum	decimal(8,2)	
缺货时间	LDate	datetime	

表 4.6 调货记录 Transfer 表

名称	代码	数据类型	注释
调货编号	TID	int	主键
货号	JID	int	外键
申请店号	FSID	int	外键
被申请店号	TSID	int	外键
调货数量	TNum	int	
调货时间	TDate	datetime	
调货状态	TStatus	char(1)	

表 4.7 库存 Repo 表

名称	代码	数据类型	注释
货号	JID	int	主键，外键
门店号	SID	int	主键，外键
库存数量	RNum	int	

4.2.6 社会、健康、安全、法律及文化方面需求

本次实验中设计的连锁品牌零售店库存管理系统，在社会层面上要求能够充分发挥商品在各家门店之间的流动性，提高门店工作人员的工作效率，提升顾客购买的体验，能够尽量规避掉购买过程中因为门店库存周转导致的缺货情况。在健康层面上，要求本系统提供的内容不能包含任何不良因素，不能对用户的心理或身体健康造成任何负面影响。在安全层面上，作为一个与数据库联系紧密的系统，数据本身的安全性以及数据库的安全性是需要着重考虑的方向，本系统要能够提供用户身份验证、用户权限控制、数据库管理人员权限控制等控制手段，并引入视图机制和事务处理机制，保障数据库的安全性和事务的 ACID 特性。在法律层面上，要求本系统严格遵守法律法规，不提供违法违规的内容，不进行违法

违规的操作，不能为违法犯罪活动提供便利。在文化层面上，要求本系统能够一定程度上推动社会发展，为社会创造价值提供便利，从而促进文化发展。作为一名工程师，所作出的设计必须在社会、健康、安全、法律和文化等层面上做到对社会有益、对健康有益、安全有保障、遵守法律法规、促进文化发展，因为工程师的设计往往是面向大众投入使用的，因此必须担负起责任，对设计负责，对广大用户负责。

4.3 总体设计

4.3.1 系统 C/S 架构图

系统采用 C/S 架构，客户机上运行客户端程序，客户端程序向数据库服务器发送请求，等待数据库服务器的响应。当客户端程序接收到来自数据库服务器的响应后，对数据进行处理，然后通过图形界面将数据展示给用户。

架构示意图如下图 4.2 所示：



图 4.2 系统 C/S 架构图

系统的客户端运行在客户机上，由用户操作客户端向数据库服务器发送带参数的请求，数据库服务器响应请求并将响应结果发送给客户机，并由客户端将结果处理之后在客户端界面上展示给用户。

4.3.2 功能模块组成及说明

本次实验设计的系统主要含有七个模块，分别是登陆模块，人员管理模块，查询模块，进货模块，售货模块，销售统计模块和调货申请处理模块。

（1）登陆模块。在登录模块，前端页面要求职员输入账号和密码，然后客户端将账号和密码传给后台服务器，服务器通过查询数据库验证职员密码。如果密码正确，则进入本系统的操作界面。

（2）人员管理模块。对店长和店员采用不同权限进行管理。以店长身份登陆时，在这个模块中可以查看本店所有员工的身份信息，包括用户名、类别和密码，且能对员工进行增加和删除操作（比如新员工入职或员工离职），同时也能对员工进行修改密码的操作。以店员身份登录时，只能查看和修改自己的部分信息，不允许进行新增员工或删除员工的操作。

(3) 查询模块。在查询模块，当员工选择库存查询时，客户端要求员工输入查询条件，后端通过查询条件对数据库进行查询操作，将得到的结果表单以传递给前端，最后客户端把符合查询条件的服装信息通过表单向用户展示。查询单的项目包括货号、品名、积木数范围、适龄范围和单价范围，查询项目支持模糊查询，允许不完整填写查询单。

(4) 进货模块。在进货模块，当员工选择进货操作时，客户端要求员工填写进货单，后端通过进货单与数据库交互，实现写入库存记录，修改积木商品表相关的操作，并将结果传递给前端，客户端再将进货入库结果反馈给员工，并准备好下一次操作。此外，进货模块还具有进货推荐功能，该模块读取全局的缺货记录和本店的缺货记录，分别给出距离当前时间一周之内的稀缺商品排行以及本店缺货情况排行榜，将最稀缺的商品信息列出，便于员工有选择性地进行进货。

(5) 售货模块。在售货模块，当员工选择售货操作时，客户端要求员工输入售货信息，后台根据实际库存判断能否出售或能否从其他门店调货，最后将满足售货条件的条目交给客户端通过表单展示，要求员工选择执行出售或调货操作。若选择出售操作，后台对数据库的商品数量表单进行更新，若选择调货操作，则写入调货表单，供其他模块调用。

(6) 销售统计模块。只有店长能访问此模块，当店长选择销售统计功能时，后台会从数据库读取销售表单，客户端会将后台处理得到的本店职工的销售排名表（按销售额排序，包含职工信息、销售单数、销售商品数量、销售额）和各个门店的销售排名（按总销售额排序，包含门店信息、销售单数、销售商品数量、销售额）。分别以表单的形式展现给店长。

(7) 调货申请处理模块。在调货申请处理模块，当店长选择处理发出的调货申请时，客户端会将后台反馈的本店发出的申请信息以表单形式展现给店长，店长查看申请状态，并可确认已被同意的调货申请、可取消正在等待的调货申请、可删除已被拒绝的调货申请。同时这些操作也会通过后台更新数据库中的表内容。当店长选择处理收到的调货申请时，客户端会将后台反馈的本店收到的申请信息以表单形式展现给店长，店长可对收到的申请进行同意或拒绝操作。

4.3.3 设计特色亮点

设计中与传统的库存管理系统相比有以下亮点：

首先支持门店之间的调货功能，门店之间可以双向收发调货申请，调货申请需要双方同意才能生效。申请具有已同意、等待、未同意三种状态，发送方对已同意的申请可以进行入库确认，对等待中的申请可以撤销，对未同意的申请可以删除。申请的接收方可以对申请进行同意或拒绝，对于库存不满足的情况，系统会进行提示并拒绝。

然后支持一定程度的模糊查询功能，对于能够唯一标识商品信息的条件，进

行精准查询，对于不能唯一标识商品信息的条件，可以进行模糊查询，不完全填写表单也能够得到查询结果。

在进货页面会根据缺货情况对用户进行提示和引导，给出稀缺商品排名，让用户更明白有哪些商品是急需进货的。

对用户区分种类，不同种类的用户对系统拥有不同的访问和操作权限，增强系统安全性和隐私机密信息保护。

4.3.4 制约因素考量

在社会层面上，为了满足充分发挥商品流动性、提高工作效率、提升顾客购买体验的要求，本系统在库存管理的基础上开发了调货功能模块，该模块能够实现门店之间的调货申请和被申请的处理，在门店售货给顾客的过程中如果出现了门店库存不足的情况，门店可以在征得顾客同意的前提下对同一商品从其他门店进行调货，发出一个调货申请。系统将写入一条调货申请到数据库中，在被申请的门店能够查看到自己收到的这条调货申请并进行处理。若调货申请被同意，则货品将直接从被申请的门店转到申请门店的库存中，这样门店就可以通知顾客来取货，整个流程实际操作起来比重新进货要快很多。

在健康层面上，本系统各个模块在设计的过程中没有涉及任何影响健康的问题，本系统中使用的数据只和积木商品库存、销售和人员管理等信息相关，不包含也不提供不良信息。

在安全层面上，本系统在设计上考虑了登录用户权限分级的问题，将用户分为店主和店员两种类型，店主具有全部的查看和操作权限，店员仅具有部分权限。从数据库角度上，客户端的查询操作都设计为采用视图完成，一定程度上保证了数据的安全性，同时本系统使用了事务机制，保障了操作的逻辑正确性以及事务的原子性、一致性、隔离性和持久性。

在法律层面上，本系统在设计过程中严格遵守法律法规，没有任何违法的信息，也没有设计任何不合法的功能模块，全部功能模块的设计只围绕正常商品销售活动中的商品、门店和用户展开。

在文化层面上，本系统在设计之初就本着提高生产力，推动社会更好发展，为用户提供便利的目标。在查询、售货和进货的核心模块中集成了模糊查询功能和商品推荐功能，极大提高了系统的便利性，更好地为社会创造价值，从而一定程度上促进了文化发展。

4.4 数据库设计

4.4.1 ER 图设计及说明

数据库设计的 ER 图使用 Power Designer 进行绘制，如下图 4.3 所示：

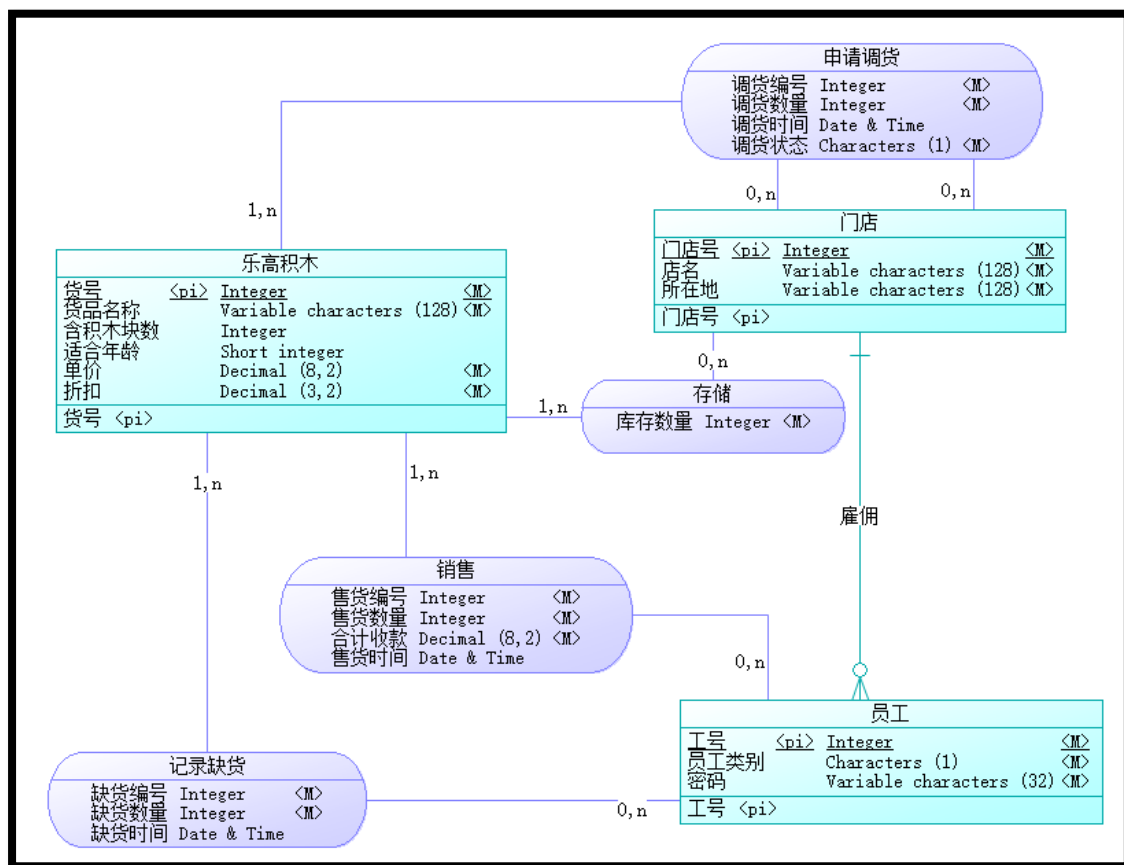


图 4.3 数据库 ER 图

可见上图中共有 3 个实体，5 个联系，其中有 4 个联系是多对多的，且关系带有属性，这样的 4 个多对多的联系将会在后续的分析中被抽象为数据库表，因此最终数据库将会有 7 张表。

如图所示，首先门店和员工之间具有雇佣关系，它是一个一对多的关系，门店是一端，员工是多端。门店由门店号唯一标识，还具有店名和所在地这两个属性。员工由工号唯一标识，还具有员工类别和密码这两个属性。

门店和乐高积木之间具有多对多的存储关系，这一联系具有库存数量的属性。乐高积木由货号唯一标识，还具有货品名称、含积木块数、适合年龄、单价和折扣这些属性。

员工和乐高积木之间具有多对多的销售关系，这一联系具有售货编号、售货数量、合计收款和售货时间这些属性。

员工和乐高积木之间还具有多对多的记录缺货关系，这一联系具有缺货编号、缺货数量、缺货时间这些属性。

4.4.2 数据库逻辑结构设计

使用 Power Designer 将上述概念模型（CDM）转为物理模型（PDM）之后，得到物理模型结构图，可以看到，图中将 4.4.1 小节中 CDM 的多对多联系转换

成表，且将表的主码、外码等特征进行标识。物理模型结构图如下图 4.4 所示：

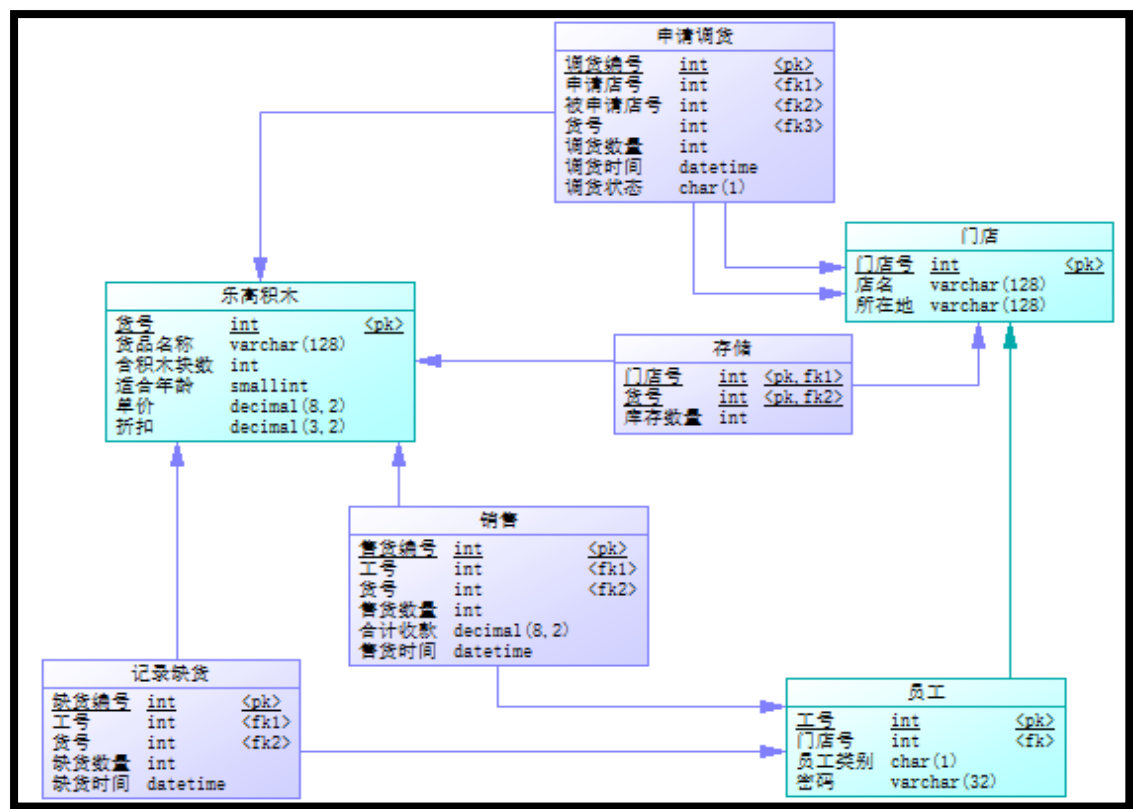


图 4.4 物理模型结构图

根据上图，可以清晰地查看各表的结构。进一步使用 Power Designer 生成满足 SQL Server 2008 R2 语法的 SQL 建表语句如下图所示：

```
D:\> Program Files (x86) > Sybase > PowerDesigner 16 > crebas.sql
1  /*=====*/
2  /* DBMS name:      Microsoft SQL Server 2008 */
3  /* Created on:     2020/5/18 12:54:37 */
4  /*=====*/
5
6
7  if exists (select 1
8  | from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
9  | where r.fkeyid = object_id('Employee') and o.name = 'FK_EMPLOYEE_HIRE_SHOP')
10 alter table Employee
11 | drop constraint FK_EMPLOYEE_HIRE_SHOP
12 go
13
14 if exists (select 1
15 | from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
16 | where r.fkeyid = object_id('Lack') and o.name = 'FK_LACK_LACK_EMPLOYEE')
17 alter table Lack
18 | drop constraint FK_LACK_LACK_EMPLOYEE
19 go
20
21 if exists (select 1
22 | from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
23 | where r.fkeyid = object_id('Lack') and o.name = 'FK_LACK_LACK2_JUGGLE')
24 alter table Lack
25 | drop constraint FK_LACK_LACK2_JUGGLE
26 go
```

图 4.5 Power Design

由于自动生成代码过长, 图中只展示了部分代码, 在 SQL Server Management Studio 中运行上图中 SQL 语句后得到如下结果:

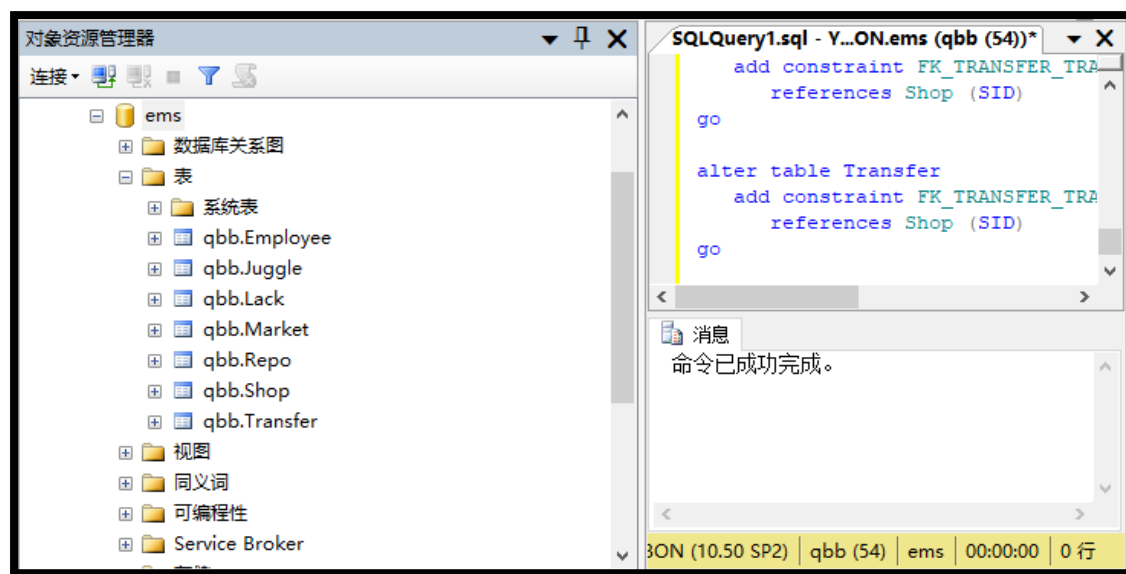


图 4.6 建表结果

由建表结果可知, 七张表均成功创建, 且与表 4.1 至表 4.7 中内容以及图 4.4 中物理模型结构图相符合。

4.5 详细设计与实现

4.5.1 平台说明

本系统在实现过程中使用的平台和工具如下:

- (1) 操作系统: Windows10 64 位
- (2) 数据库软件: SQL Server 2008 R2
- (3) 数据源: ODBC (开放数据互联)
- (4) 开发语言: C++
- (5) 图形库: QT 5.13.1
- (6) 编辑器: QT Creator

4.5.2 功能模块实现

本系统的实现依托于 QT 图形库编写的客户端与 SQL Server 数据库形成的连接, 在 QT 中与 SQL Server 建立连接需要先以 4.4.2 节中建立的数据库为基础, 建立 ODBC 数据源, 如图 4.7 所示:

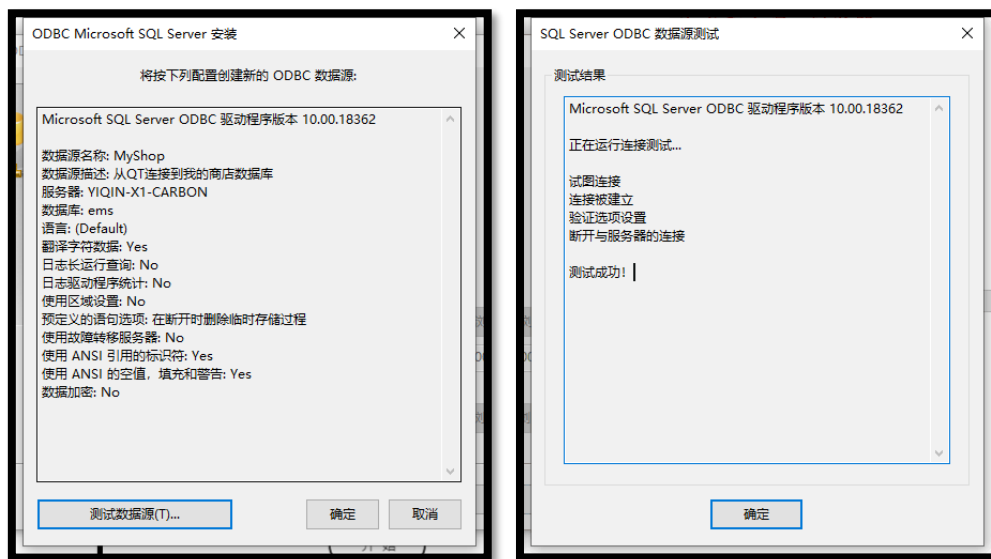


图 4.7 ODBC 数据源的建立

然后在 QT 中使用 ODBC 驱动连接到数据源，完成客户端和数据库服务器的连接过程，连接部分的代码如下：

```
bool OpenDatabase() {
    QSqlDatabase db =
        QSqlDatabase::addDatabase("QODBC"); //数据库驱动类型为SQL Server
    qDebug() << "ODBC driver?" << db.isValid();
    QString dsn = QString::fromLocal8Bit("MyShop"); //数据源名称
    db.setHostName("localhost"); //选择本地主机, 127.0.1.1
    db.setDatabaseName(dsn); //设置数据源名称
    db.setUserName("db_read_write"); //登录用户
    db.setPassword("123456"); //密码
    if (!db.open()) //打开数据库
    {
        qDebug() << db.lastError().text();
        QMessageBox::critical(nullptr, QObject::tr("Database error"),
                                db.lastError().text());
        return false; //打开失败
    } else
        qDebug() << "database open success!";
    return true;
}
```

图 4.8 在 QT 中连接 ODBC 数据源

如上图 4.8 所示,QT 连接到创建的 MyShop 数据源,其对应的数据库是 ems, 登录名为 db_read_write, 密码为 123456。以该登录名登录 ems 数据库的权限仅包括数据的读写,一定程度保护了数据库的安全。

下面介绍系统中的各个模块的具体实现情况：

(1) 登陆模块实现

登录模块的流程图如下图 4.7 所示：

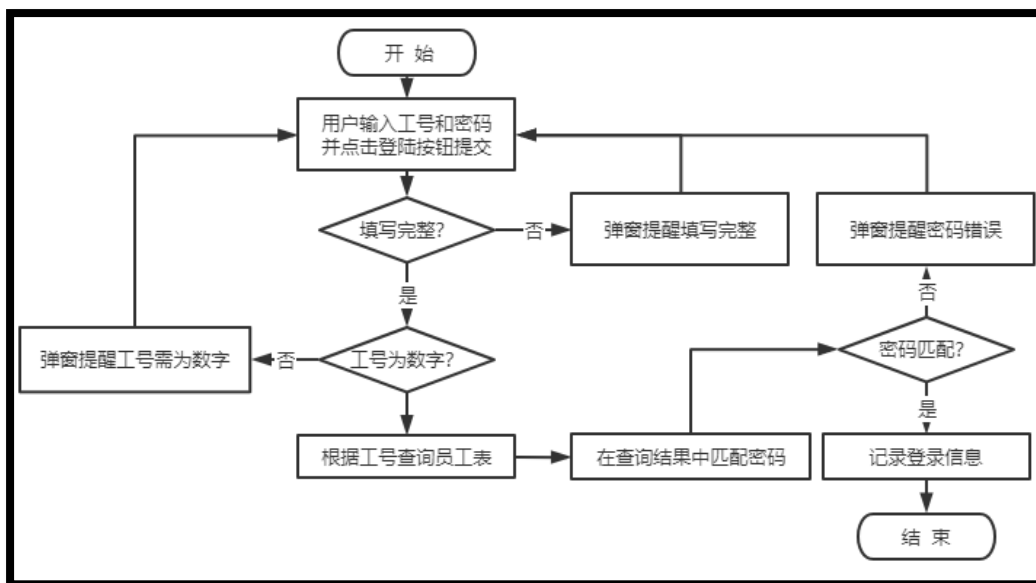


图 4.7 登录模块流程图

如上图所示，在本模块，登录时要求用户输入账号（工号）和密码，若工号或密码未填写完整，或者工号栏填写的不是数字则会弹窗提示。正确输入后客户端将账号和密码传给后台服务器，服务器通过工号使用 `select` 语句查询数据库中的员工表 `Employee`，若未查到结果则弹窗提示，若查到结果则将输入密码与查询结果的密码项进行对比，若比对失败则弹窗提示密码错误，否则将登录的用户信息进行记录，并注销当前窗口，启动主窗口的操作界面，完成登录。

（2）人员管理模块实现

人员管理模块的流程图如下图 4.8 所示：

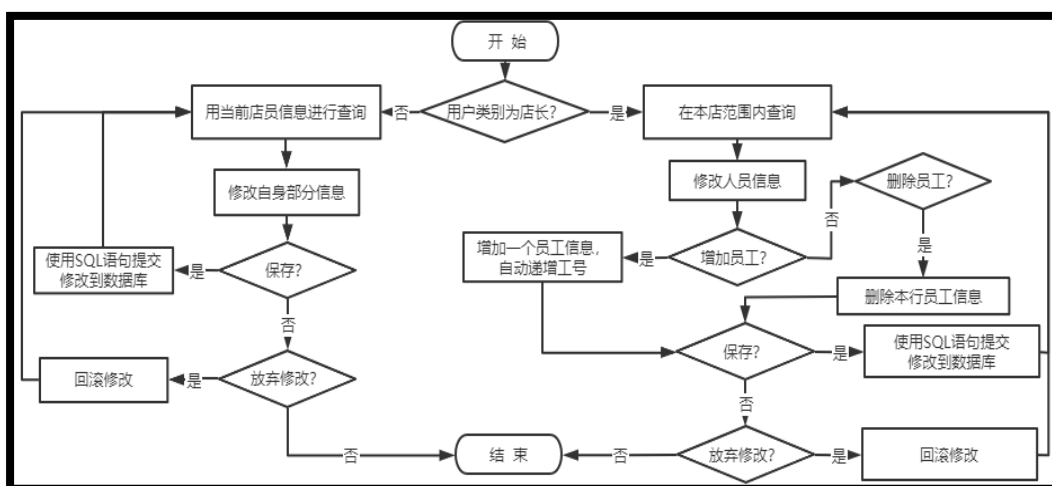


图 4.8 人员管理模块流程图

如上图所示，本模块对店长和店员采用不同权限进行管理。以店长身份登陆时，在这个模块中不仅可以查看并修改本店所有员工的身份信息，还能对员工进行增加和删除操作，若增加一个新员工则会自动递增其工号，避免工号重复破坏

了实体完整性。若点击删除员工，则会删除选中的那一行员工的信息。若以店员身份登陆系统，则只能对自己的部分信息进行修改（比如店员不能修改自己的员工类别为店长），且不能进行员工的增加或删除操作，保证了人员管理的安全性。注意本模块使用了事务机制，用户的所有操作都只是在前端完成，确认无误后点击保存按钮进行提交，将所有修改提交至数据库。若用户点击放弃修改按钮，则将前端界面还原为修改前的状态，数据库中的数据没有进行改变。

（3）查询模块实现

查询模块的流程图如下图 4.9 所示：

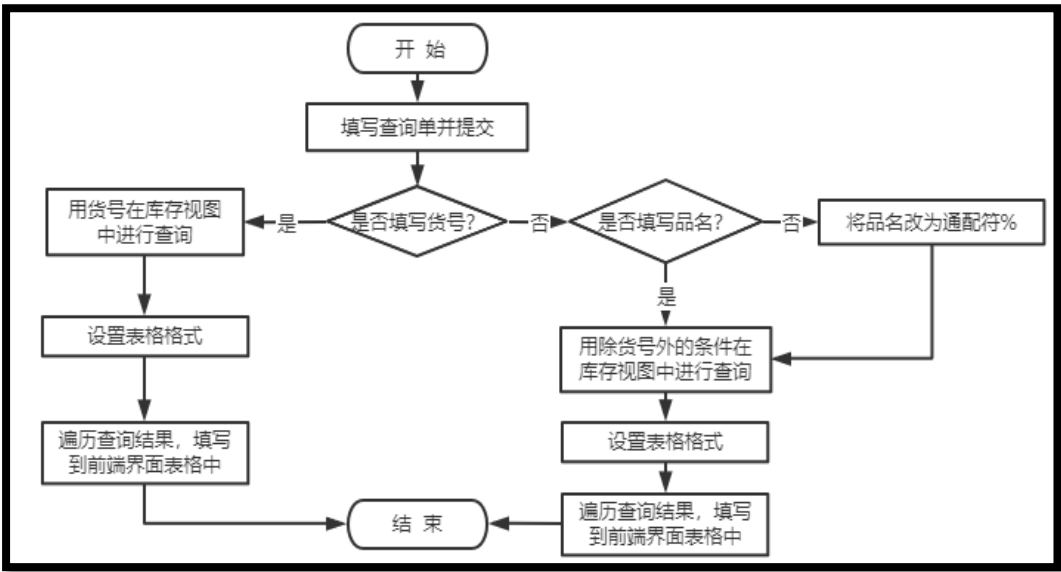


图 4.9 查询模块流程图

在查询模块，员工首先需要填写查询单，查询单的项目包括货号、品名、积木数范围、适龄范围和单价范围，查询项目支持模糊查询，允许不完整填写查询单。

当员工填写完查询单并点击提交按钮后，前端判断用户是否填写货号。若已填写货号，则可以以货号和当前门店号作为条件在库存视图中进行查找，因为货号和门店号构成库存表的主键，因此可以唯一确定一条库存信息。若未填写货号，则使用其他条件进行查询。首先判断是否填写品名，若未填写品名则将品名替换为通配符“%”，然后使用 like 关键字编写 SQL 查询语句，其他条件使用的则是范围查询。后端通过查询条件对数据库进行查询操作，将得到的结果传递给客户端界面按设定好的格式进行展示。

（4）进货模块实现

进货模块的流程图如下图 4.10 所示：

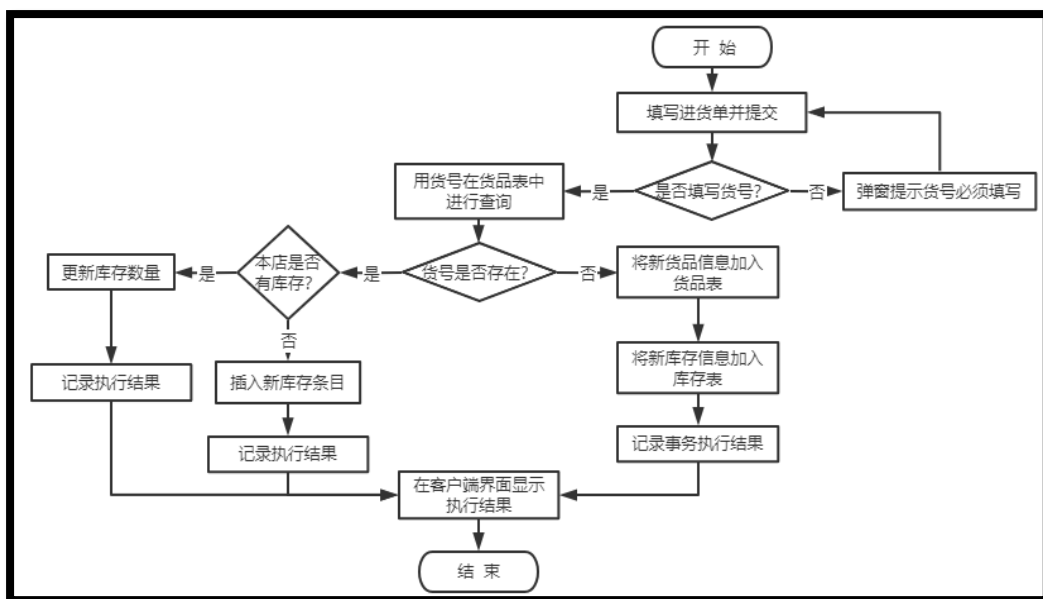


图 4.10 进货模块流程图

在进货模块，员工首先需要填写进货单，进货单的项目包括货号、品名、积木数、适龄、单价、折扣和进货数量，其中货号和进货数量为必填项，在货号确定的情况下，允许不将进货单填写完整。

当员工填写完进货单并点击提交按钮后，前端判断用户是否填写货号。若未填写货号则弹窗提示。

若已填写货号，则可以凭借货号作为条件在货品表 **Juggle** 中查找商品信息。若查询结果集为空说明货号不存在，则创建以下事务：将新商品的在进货单中填写的货品信息加入到货品表中，然后将新的库存信息加入到库存表中，最后记录事务的执行结果并在客户端界面进行显示。

若查询结果集不为空则说明货号已存在，接下来继续判断本店内是否有该货号商品的库存。

若本店内有该商品的库存，则可以直接使用 **update** 语句更新库存表中的商品数量，更新数量之后记录执行结果为消息字符串 **msg**，并在客户端界面进行显示。若本店内无该商品的库存，则需要使用 **insert** 语句插入一个新的库存条目，将其商品数设置为进货单上的进货数，然后记录执行结果为消息字符串 **msg**，并在客户端界面进行显示。客户端界面有一个 **list** 部件，能够将消息组织成一个链表的形式，并将执行结果的消息记录按照时间顺序打印在界面上，历史消息也能够予以展示。

(5) 售货模块实现

售货模块的流程图如下图 4.11 所示：

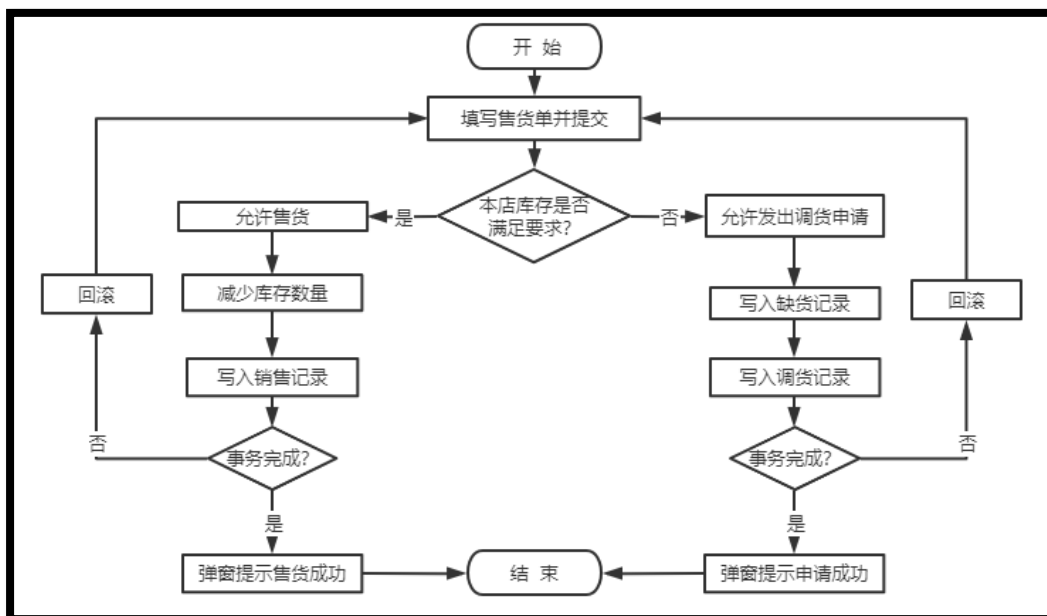


图 4.11 售货模块流程图

在售货模块，员工首先需要填写售货单，售货单的项目包括货号、品名、积木数范围、适龄范围、单价范围和售货数量，其中货号和售货数量为必填项，在货号确定的情况下，允许不将售货单填写完整。

当员工填写完进货单并点击提交按钮后，系统在数据库中按照要求进行查询，若本店库存满足要求则允许售货，否则对于其他门店满足要求的货品则允许发出调货申请。若员工选择售货，则进行如下事务：减少库存数量、写入销售记录，若事务未成功完成则回滚，否则提示售货成功。若员工选择调货，则进行如下事务：写入缺货记录、写入调货记录，若事务未成功则回滚，否则提示申请成功。

(6) 销售统计模块实现

销售统计模块的流程图如下图 4.12 所示：

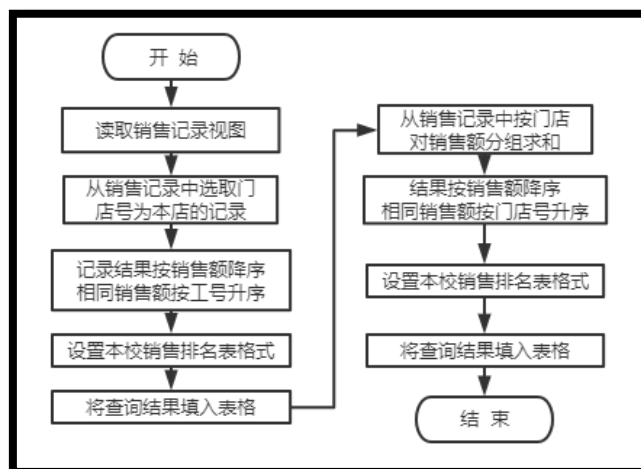


图 4.12 销售统计模块流程图

在销售统计模块，本系统提供两个统计表格的展示。一个是本店内员工的销

售情况排名,另一个表格则是所有门店的总销售额排名,只有店长能访问此模块。

当此模块界面加载时,首先在销售记录视图中进行查询操作,对于本店员工销售排名,以当前门店号为查询条件,使用“order by Amount desc, EID”条件对查询结果按照销售额 Amount 降序排序,销售额相同时按照工号 EID 升序排序。

然后对于各门店销售情况排名,将销售记录视图和门店表进行自然连接,对查询结果按门店总销售额降序排序,销售额相同时按照门店号升序排序。

得到查询结果后,设置两个表格的格式,将查询结果填入表格展示即可。

(7) 调货申请处理模块实现

调货申请处理模块的流程图如下图 4.13 和 4.14 所示:

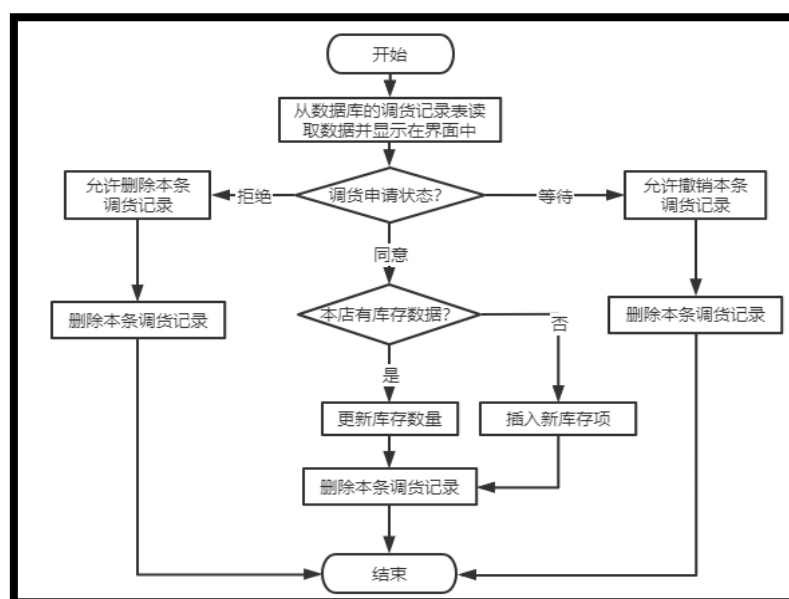


图 4.13 调货申请处理模块流程图——发出的申请

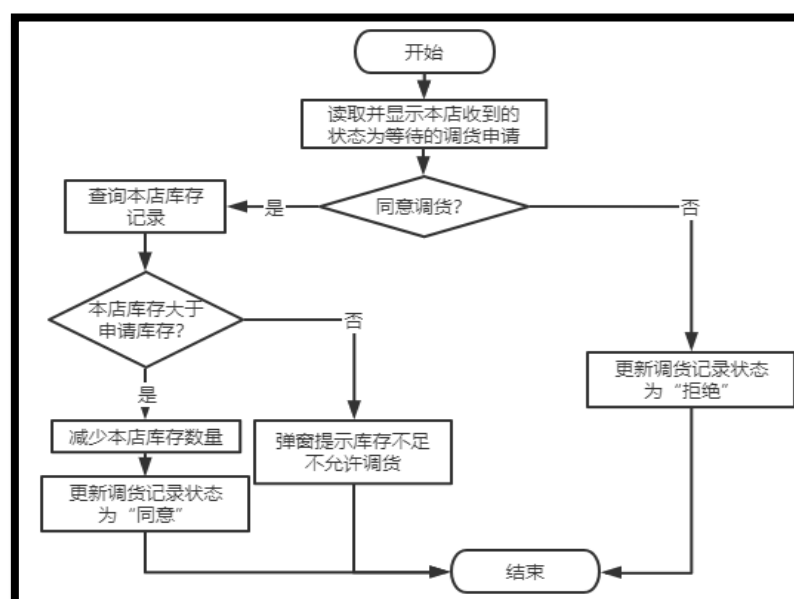


图 4.14 调货申请处理模块流程图——收到的申请

调货申请处理模块可以分为处理发出的申请和处理收到的申请两个子模块。

①处理发出的申请

如图 4.13 所示，首先读取本店发出的每条调货申请的状态，并显示在客户端界面上。

对于状态为“拒绝”或“等待”的调货申请，发出申请的门店可以将该调货申请删除或撤销，两种操作的效果是一样的，实质上都是在数据库中将对应的调货申请条目删除。

对于状态为“同意”的调货申请，门店可以进行确认入库操作。确认时先查询本店库存，若本店有库存数据则直接更新库存数量，若没有对应库存数据则插入一条新的库存条目，处理结束后删除该调货申请。

②处理收到的申请

本模块因为涉及出库操作，仅允许店主访问。如图 4.14 所示，首先读取并显示本店收到的状态为“等待”的调货申请。

若用户点击允许调货按钮，则先查询本店的库存记录，若本店当前库存记录大于被申请的商品数目则相应减少本店的库存数量，并更新调货记录的状态为“同意”，否则弹窗提示库存不足不允许调货。

若用户点击拒绝调货按钮，则直接更新调货记录的状态为“拒绝”。

4.5.3 数据库相关技术的使用与实现

（1）视图的定义与实现

本系统中共建立了如下图 4.15 所示的 5 个视图：



图 4.15 本系统中视图

①EmployeeSafe 视图

该视图建立在 Employee 表的基础上，删去了密码这一列，在与员工相关但不涉及到密码查询时使用 EmployeeSafe 视图可以防止含有密码的 Employee 表被直接读取，有利于提高系统的安全性。该视图建立时只需要对 Employee 表做投影即可。视图建立的 SQL 语句如下图 4.16 所示：

```
SELECT EID, EName, EType, SID
FROM qbb.Employee
```

图 4.16 建立 EmployeeSafe 视图

②RepoDetail 视图

该视图建立在 Repo 表和 Juggle 表的连接的基础上，是对库存记录 Repo 表的一个扩充，根据商品号加入了商品的各项信息，在查询库存同时需要查询商品信息时使用，可以省去一次连接操作，便于 SQL 语句的编写以及查询。该视图建立时只需要将 Repo 表和 Juggle 表做自然连接即可。视图建立的 SQL 语句如下图 4.17 所示：

```
SELECT qbb.Repo.JID, qbb.Juggle.JName, qbb.Juggle.JNum, qbb.Juggle.JAge,
qbb.Juggle.Price, qbb.Juggle.Discount, qbb.Repo.SID, qbb.Repo.RNum
FROM qbb.Repo INNER JOIN
qbb.Juggle ON qbb.Repo.JID = qbb.Juggle.JID|
```

图 4.17 建立 RepoDetail 视图

③SaleCount 视图

该视图建立在 EmployeeSafe 视图 Market 表的基础上，根据员工工号 EID 进行分组统计得到员工销售单数、销售商品数和销售总额，并将其作为一个视图。该视图在销售记录统计时使用，可以较便捷地获取销售统计所需要的信息，省去了对两张表多次进行连接和投影的操作，又可以防止客户端直接读取原始表，保障数据安全性。视图建立的 SQL 语句如下图 4.18 所示：

```
SELECT TOP (100) PERCENT dbo.EmployeeSafe.EID, dbo.EmployeeSafe.EName,
COUNT(*) AS Records, SUM(qbb.Market.MNum) AS Items, SUM(qbb.Market.TotalPrice) AS Amount,
dbo.EmployeeSafe.SID
FROM dbo.EmployeeSafe INNER JOIN
qbb.Market ON dbo.EmployeeSafe.EID = qbb.Market.EID
GROUP BY dbo.EmployeeSafe.EID, dbo.EmployeeSafe.EName, dbo.EmployeeSafe.SID
```

图 4.18 建立 SaleCount 视图

④TransferDetail 视图

该视图建立在 Transfer 表、Juggle 表以及两张 Shop 表的连接的基础上，是对调货记录 Transfer 表的一个扩充，根据商品号加入了商品的各项信息，根据申请店号和被申请店号加入了双方门店的各项信息。在处理调货申请时的展示界面使用该视图能够方便快捷地获取到需要展示的内容，不用编写冗长复杂的 SQL 语句，又能防止客户端直接读取原始表，保障数据安全性。该视图建立时只需要将 Transfer 表、Juggle 表以及两张 Shop 表分别根据 JID、FSID、TSID 这三个属性做自然连接即可，视图建立的 SQL 语句如下图 4.19 所示：

```
SELECT qbb.Transfer.TID, qbb.Juggle.JID, qbb.Juggle.JName, qbb.Juggle.Price, qbb.Transfer.FSID, s1.SName AS FSName,
s1.CityName AS FCityName, qbb.Transfer.TSID, s2.SName AS TSName,
s2.CityName AS TCityName, qbb.Transfer.TNum, qbb.Transfer.TDate, qbb.Transfer.TStatus
FROM qbb.Transfer INNER JOIN
qbb.Juggle ON qbb.Transfer.JID = qbb.Juggle.JID INNER JOIN
qbb.Shop AS s1 ON qbb.Transfer.FSID = s1.SID INNER JOIN
qbb.Shop AS s2 ON qbb.Transfer.TSID = s2.SID
```

图 4.19 建立 TransferDetail 视图

⑤LackCount 视图

该视图建立在缺货记录 Lack 表的基础上，对 Lack 表做了投影操作，删去了工号 EID 这一列，在于缺货记录相关的查询中使用该视图可以防止 Lack 表被直接读取，有利于提高系统的安全性。视图建立的 SQL 语句如下图 4.20 所示：

```
SELECT  LID, JID, LNum, LDate
FROM    qbb.Lack
```

图 4.20 建立 LackCount 视图

(2) 事务的定义与实现

在系统实现中有多处使用事务机制，使用方式为：在事务开始之前执行 SQL 语句“begin transaction”开始事务。在中间使用 SQL 语句描述事务动作。在最后判断事务中各个操作是否顺利完成，若顺利完成则使用“commit”语句提交修改。若未能顺利完成事务则使用“rollback”语句回滚到事务开始之前的状态。

查阅资料后发现 SQL Server 的单条 SQL 语句的执行会自动被事务机制保护，因此只需要考虑多条语句构成一个事务的情况。

以售货模块中本店成功售出货品这一事务为例，事务中需要进行的操作是：①减少本店库存数量、②写入一条新的销售记录。这两个操作需要事务机制进行保障，因为如果①操作成功完成而②操作未完成的话，就会出现库存减少但没有销售记录的情况，对于门店来说这种情况会导致账目信息错误。同理如果①失败而②成功了，则会出现一条没有实际销售的虚假销售记录。因此需要采用事务机制进行保护。实践中运用事务机制的代码样例如下图 4.21 所示：

```
//开始事务
query.exec("begin transaction");
//更新库存数量
bool repoOK = query.exec(
    QString("update qbb.Repo set RNum=RNum-%1 where JID=%2 and SID=%3")
    .arg(MNum)
    .arg(JID)
    .arg(SID));
//写入销售记录
bool marketOK = query.exec(
    QString("insert into qbb.Market(JID,MNum,TotalPrice,EID,MDate) "
        "values (%1,%2,%3,%4,'%5')")
    .arg(JID)
    .arg(MNum)
    .arg(TotalPrice)
    .arg(login::loggedEmp.EID)
    .arg(MDate));
if (repoOK && marketOK) {
    query.exec("commit");
    QMessageBox::information(nullptr, QStringLiteral("完成"),
        QStringLiteral("售货成功!"),
        QMessageBox::Ok, QMessageBox::Ok);
} else {
    query.exec("rollback");
    QMessageBox::warning(nullptr, QStringLiteral("警告"),
        QStringLiteral("出现错误，已回滚"),
        QMessageBox::Ok, QMessageBox::Ok);
}
//结束事务
```

图 4.21 使用事务机制的一个例子

（3）触发器的定义与实现

触发器的引入可以在数据表被修改时根据条件自动地完成一些动作，保证数据库的完整性。本系统中在库存表 Repo 上定义了一个触发器，名为 deleteWhenZero，该触发器的作用是在更新操作之后检查库存数量，如果库存数量小于等于 0 则从 Repo 表中删除掉这一库存项目。实现该触发器的 SQL 语句如下图所示 4.22 所示：

```
USE [ems]
GO
/***** Object: Trigger [qbb].[deleteWhenZero] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
--当库存为0时在Repo 表中删除库存信息
ALTER trigger [qbb].[deleteWhenZero] on [qbb].[Repo]
after update
as
if update(RNum)
begin
    declare @afterRNum int
    declare @afterJID int
    declare @afterSID int
    select @afterRNum=RNum,@afterJID=JID,@afterSID=SID from inserted
    if @afterRNum=0
    begin
        delete from Repo where JID=@afterJID and SID=@afterSID
    end
end
end
```

图 4.22 建立 deleteWhenZero 触发器

（4）数据库用户的权限管理

在本系统的数据库管理系统中我自定义了 4 个登录名，分别为 54397、qbb、db_read_write 和 db_backup。如下图 4.23 所示：

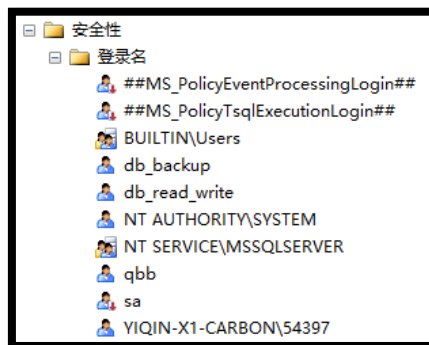


图 4.23 数据库登录名

其中 54397 为 Windows 验证登陆方式，作为整个数据库的管理员，具有一切权限。qbb 是本系统使用的数据库 ems 的所有者，对 ems 数据库具有全部权限。db_read_write 被赋予了 db_datareader 和 db_datawriter 角色，对 ems 数据库的数据具有读写权限。db_backup 被赋予了 db_backupoperator 角色，对 ems 数据库具有备份权限。

使用不同的登录名、用户对数据库进行权限分级的管理，有利于提高系统的安全性，角色（ROLE）的引入便于控制权限分配。

4.5.4 制约因素考量

在社会层面上，为了满足充分发挥商品流动性、提高工作效率、提升顾客购买体验的要求，本系统在货品查询模块、进货模块和售货模块的核心功能的设计上，都考虑了对于用户的人性化提示，在用户非法操作时会弹窗予以提示。并且本系统加入的调货申请处理模块在实现过程中利用“发出的申请”和“收到的申请”这两个子模块完成了 4.3 节中对调货申请模块的设计，不仅展示了调货申请的状态，也提供了按钮与对应槽函数连接，根据调货申请的内容完成对数据库的操作。

在健康层面上，本系统在实现过程中各个模块只通过 ODBC 数据源访问数据库，且数据库中的数据没有涉及任何影响健康的因素。

在安全层面上，本系统将登录用户进行权限分级，在人员管理模块的实现过程中，根据登录用户的类型提供不同的界面和操作。在销售统计模块和调货处理模块的实现过程中，对界面访问进行权限保护，店员无法访问销售统计模块和调货处理模块的“收到的申请”子模块。此外，本系统还使用了视图、事务、和数据库层面的权限管理等技术来保障系统的安全性，这部分的具体实现见 4.5.3 节的描述。

在法律层面上，本系统的各个模块在实现过程均考虑了合法合规的要求，不存在违法的功能。实现中使用的开发工具、数据库软件和第三方库等均为免费版本或者是开源版本，且在本次实验中用于学习使用，因此不存在法律层面的侵权问题。

在文化层面上，本系统的查询、售货和进货的核心模块在实现过程中使用数据库中的通配符查询来完成模糊查询，通过对缺货记录表进行查询和统计实现了进货商品推荐的功能，提高了系统的便利性，更好地为社会创造价值，从而一定程度上促进了文化发展。

4.6 系统测试

4.6.1 测试数据与计划

为了测试系统功能，使用 Excel 随机生成了一些数据写入到数据库的表中。由于篇幅限制，以 Transfer 表为例，使用函数随机生成的 Excel 表格内容如下图所示：

	A	B	C	D	E	F
1	FSID	TSID	JID	TNum	TDate	TStatus
2		1	2	10190	1 2020-04-07 10:38:55	0
3		1	3	10008	10 2020-05-10 05:31:23	1
4		2	3	10265	6 2020-05-07 05:56:44	0
5		1	2	10292	1 2020-05-26 03:16:57	1
6		1	2	10188	3 2020-05-28 21:30:39	2
7		1	2	10150	9 2020-04-12 09:07:35	1
8		2	3	10014	4 2020-03-16 01:13:32	2
9		1	3	10063	10 2020-05-04 21:17:24	1
10		2	1	10285	9 2020-03-26 15:12:14	0
11		2	3	10289	9 2020-03-23 06:17:10	1
12		3	2	10128	1 2020-06-19 10:26:44	1
13		1	2	10250	5 2020-04-14 16:55:45	0
14		4	2	10055	5 2020-05-01 08:05:22	1
15		2	3	10089	3 2020-06-04 10:23:56	1
16		3	4	10038	9 2020-05-12 07:46:55	2
17		2	4	10209	4 2020-06-05 06:57:13	2
18		3	1	10149	4 2020-06-08 05:14:42	0
19		2	1	10186	7 2020-06-15 01:52:52	0
20		2	1	10260	5 2020-04-07 04:35:11	0

图 4.24 Excel 生成的数据

其中 FSID 为申请店号，TSID 为被申请店号，JID 为商品编号，TNum 为调货数量，TDate 为申请时间，TStatus 为状态编号（取值范围为 0，1，2）。以申请时间为例，使用 Excel 的随机函数生成当前日期前 100 天以内的一个随机日期，其他数据项也使用类似操作生成随机数据。生成 Excel 表之后使用数据导入功能导入到数据库中即可。

测试计划如下表 4.8 所示：

表 4.8 测试计划表

模块名	测试点编号	测试说明
登录模块	1	输入错误的用户名和密码
登录模块	2	输入正确的用户名和密码
人员管理模块	3	新增一名员工
人员管理模块	4	删除一名员工
查询模块	5	使用模糊查询功能查询商品信息
进货模块	6	分别进货已存在的商品和不存在的商品
售货模块	7	售出本店商品
售货模块	8	向其他门店发出调货申请
销售统计模块	9	查看销售统计结果
调货申请处理模块	10	1 号店确认 4 号店同意的申请
调货申请处理模块	11	1 号店拒绝 3 号店发出的一个申请

下面的 4.6.2 节将按照上表中的测试计划对程序的功能进行测试，并给出测试过程与结果分析。

4.6.2 测试过程与结果分析

(1) 测试点 1

对登录模块进行测试，在登录界面输入错误的用户名和密码，结果如下图 4.25 所示：

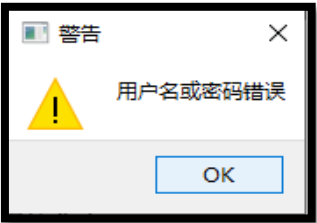


图 4.25 测试点 1 测试结果

该结果说明本系统能够识别用户名或密码错误的情况，并能够给出提示，阻止无法提供正确用户名和密码的用户登录到本系统中。测试结果与预期结果相符合。

(2) 测试点 2

对登录模块进行测试，在登录界面输入正确的用户名和密码，结果如下图 4.26 所示：



图 4.26 测试点 2 测试结果

测试过程以用户名（工号）411，密码 111 作为输入。由上图可知，输入正确的用户名和密码后能够自动注销登录窗口，并进入到本系统的主窗口。测试结果与预期结果相符合。

(3) 测试点 3

对人员管理模块进行测试，点击新增员工按钮新增一名员工，结果如下图 4.27 所示：



图 4.27 测试点 3 测试结果

如上图 4.27 所示，以店主身份登录之后，在店内人员管理页面点击新增员工按钮，系统会在末尾自动增加一个员工条目，并自动递增工号、自动填写所属门店号，由店主自主输入员工类型码、员工的登录密码以及员工名字。填写完成后点击 Save All 按钮，新增员工的信息便会提交到数据库。测试结果与预期结果相符合。

(4) 测试点 4

对人员管理模块进行测试，点击开除员工按钮删除一名员工，结果如下图 4.28 所示：



图 4.28 测试点 4 测试结果

如上图 4.27 所示，以店主身份登录系统之后，进入店内人员管理页面，在表格中选中待开除员工(选择测试点 3 中新增的员工)，然后点击开除员工按钮，系统弹窗询问是否删除该员工，点击确定后从数据库中删除该员工的信息条目。

可以看到图中“新增员工”的信息已经被删除，测试结果与预期结果相符合。

(5) 测试点 5

对查询模块进行测试，使用模糊查询功能查询商品信息，结果如下图 4.29 所示：

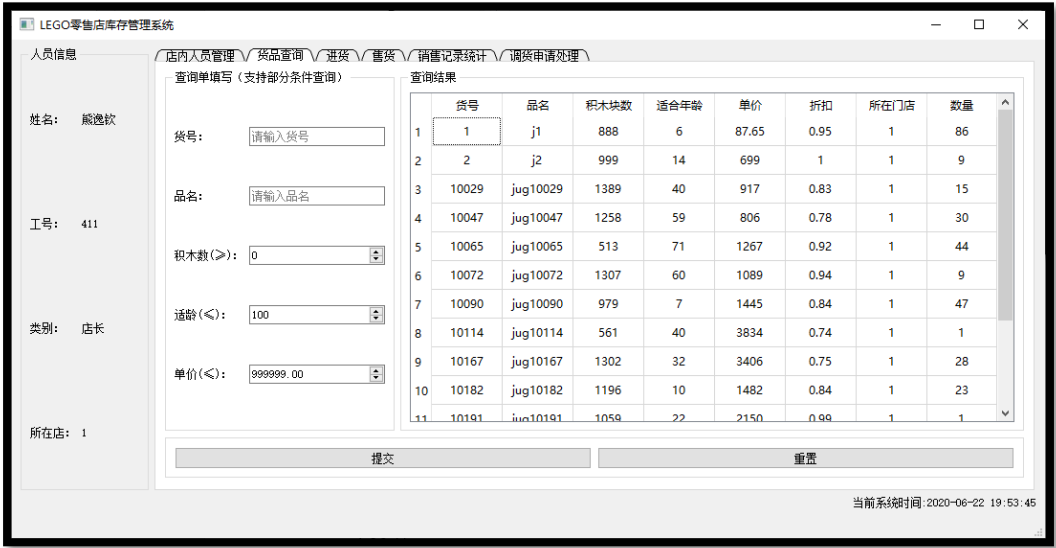


图 4.29 测试点 5 测试结果

登陆系统后进入货品查询界面，在查询单中既不输入货号也不输入品名，积木数、适龄和单价等属性采用默认值，使用模糊查询时，系统会使用通配符对未输入的查询条件进行替换。按照这个规则，这个查询单将能够查询出门店内全部商品。实际查询结果如图 4.29 所示，测试结果与预期结果相符合。

(6) 测试点 6

对进货模块进行测试，分别进货已存在的商品和不存在的商品，结果如下图 4.30 和 4.31 所示：



图 4.30 测试点 6 测试结果 1

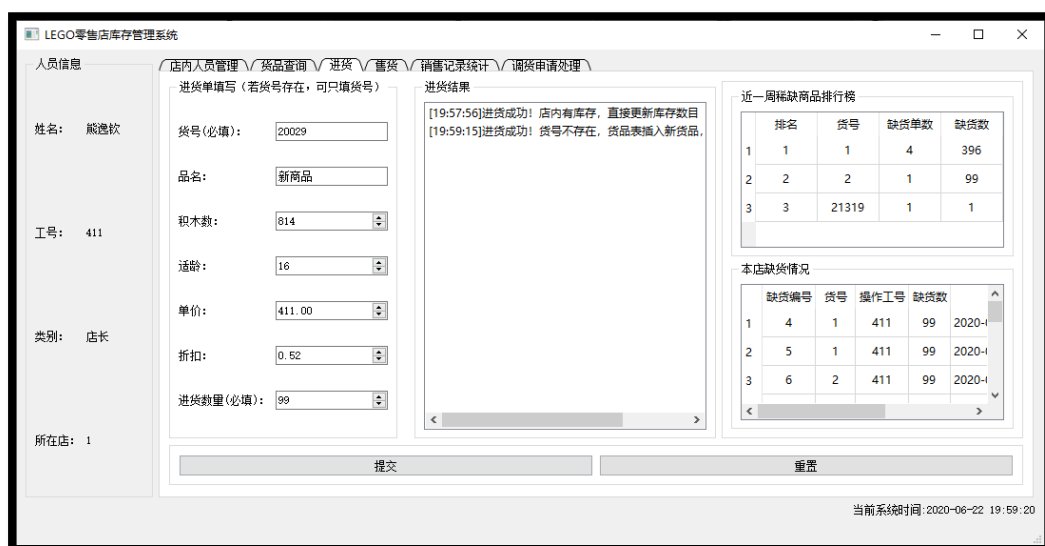


图 4.31 测试点 6 测试结果 2

图 4.30 测试了对一件库存中存在的商品（货号为 10029）进行进货操作，从图 4.30 中可以看到，进货操作的结果会显示在进货结果 groupBox 中。由进货结果可以看到，系统能成功识别店内有该商品的库存，并直接更新了库存数目，测试结果与预期结果相符合。

图 4.31 测试了对一件库存中不存在的商品（货号为 20029）进行进货操作，从图 4.31 中可以看到，进货操作的结果会显示在进货结果 groupBox 中。由进货结果可以看到，系统能成功识别该商品货号不存在，把进货单中填写的商品信息保存至商品表中，并新增一条库存项目，测试结果与预期结果相符合。

（7）测试点 7

对售货模块进行测试，售出本店商品，结果如下图 4.32 所示：



图 4.32 测试点 7 测试结果

这一测试点测试了对本店库存充足的商品进行售货操作。测试中限定了货号

为测试点 6 中新增的商品（货号 20029），该商品库存为 99 件，欲出售 10 件，库存满足条件。用户点击出售按钮后系统弹窗提示售货成功，意味着库存数量更新为 89，且写入了一条销售记录，测试结果与预期结果相符合。

(8) 测试点 8

对售货模块进行测试，向其他门店发出调货申请，结果如下图 4.32 所示：



图 4.32 测试点 8 测试结果

这一测试点依然测试了售货模块，不同的是这次使用模糊查询，并选择了一件其他门店的商品进行调货操作。当用户点击调货按钮时，会向数据库中的调货申请表写入一条记录，代表着本店向对应库存项目的所在店发出一条调货申请。由图 4.32 可见，系统能够成功处理发出调货申请的情况，并弹窗提示。

(9) 测试点 9

对销售统计模块进行测试，查看销售统计结果，结果如下图 4.33 所示：



图 4.33 测试点 9 测试结果

测试点 9 测试了销售记录统计界面的展示功能。由上图 4.33 可见，本系统能够正确统计本店职工的销售情况并按照 4.3 节中设计的顺序进行排序，同时也能够以门店为单位，正确统计各家门店的销售情况，并按照设计的顺序进行排序展示。实际使用过程中展示效果良好，测试结果与预期结果相符合。

(10) 测试点 10

对调货申请处理模块进行测试，1 号店确认 4 号店同意的申请，结果如下图 4.34 所示：



图 4.34 测试点 10 测试结果

测试点 10 测试了调货申请处理模块中“发出的申请”这一子模块。在这个界面可以查看本店发出的所有调货申请的状态，并对它们进行操作。在这个测试点中，用户确认了一个向 4 号店发出的，并且已经被 4 号店同意了调货申请。在用户点击确认按钮后，系统会检查本店库存中是否有该货品，若有则直接更新库存，否则写入新的库存项。由图 4.34 可知，系统检测出库存中无此商品，并新增了库存项目，完成了本次调货流程，测试结果与预期结果相符合。

(11) 测试点 11

对调货申请处理模块进行测试，1 号店拒绝 3 号店发出的一个申请，结果如下图 4.35 所示：

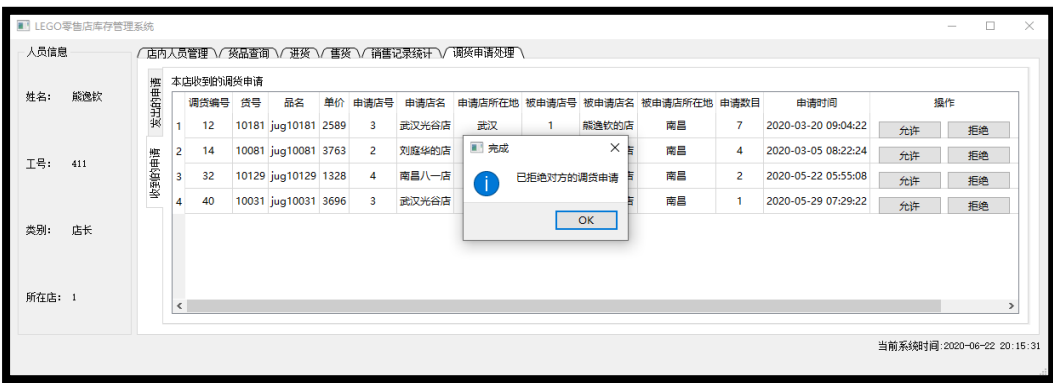


图 4.35 测试点 11 测试结果

测试点 11 测试了调货申请处理模块中“收到的申请”这一子模块。在这个界面可以查看本店收到的所有调货申请的状态，并对它们进行操作。在这个测试点中，1 号店（本店）拒绝了 3 号店发来的一个调货申请。当用户点击拒绝按钮时，系统定位到用户所操作的那一条调货申请，并将该条申请的状态由“等待”状态修改为“拒绝”状态，并将新状态更新到数据库中。由图 4.35 可见，系统成功完成了上述操作，拒绝了 3 号店发来的申请，测试结果与预期结果相符合。

综上所述，系统能正常通过测试计划中的全部测试点，各模块能够实现设计中所要求的功能。

4.7 系统设计与实现总结

在整个应用系统的设计和实现过程中，我所做的主要工作可以分为需求分析、总体设计、数据库设计与实现、客户端设计与实现和系统测试这几部分。

需求分析方面，我阐述了系统的应用背景以及系统对功能、性能以及完整性方面的需求，使用数据流图、数据字典等分析方法对系统所需要的数据处理方式进行了分析，并且分析了系统在社会、健康、安全、法律及文化方面的需求。可以说需求分析是应用系统设计过程中不可缺少的重要一步，它能够明确系统设计的目标。

然后在总体设计方面，我确定了系统使用的架构模式为 C/S 架构，并且明确了系统的七个功能模块，对需求分析中提出的需求进行一一对应的满足。此外我还总结描述了系统的亮点和特色。还针对社会、健康、安全、法律及文化方面的需求提出了制约因素的考量。总体设计过程能够让我初步把握应用系统的设计框架，并且把需求划分为一个个小的功能模块去对应实现。

在数据库的设计与实现部分，我使用 Power Designer 辅助设计工具绘制了系统的 E-R 图，并通过这一概念模型生成了物理模型，得到了数据库中的各张表。结合实际情况对 Power Designer 自动生成的数据表进行调整之后导出为 SQL Server 可以执行的 SQL 语句代码，并复制到 SQL Server 中运行，即可自动建立数据库中的各表。

在客户端的设计与实现部分，我使用 C++ 语言配合 QT 图形库编写了一个具有图形化界面的客户端程序，该程序能够将前端用户填写的表单转化为对应的 SQL 查询语句，并通过 ODBC 驱动将查询请求发送到数据库服务器进行查询，并将得到的查询结果处理后放入图形化的表格控件中进行显示。分工明确的功能模块以及设计完善的数据库使得客户端设计和实现的过程比较顺利，整体结构层次分明。

在系统测试部分，为了测试数据的完善，我仿照任务三的 SQL 练习部分，使用 Excel 提供的函数对每个表随机生成了一批数据。在这个过程中，需要考虑

的重点是随机生成过程中数据的关联性，比如说发送申请的门店号和接受申请的门店号不能是同一个，对于这个问题我使用“加 $n-1$ 取模”的方式解决，其中 n 为门店总数。有了较为全面的数据之后，系统测试部分就比较简单了，只需要针对每个模块的功能，选取较为关键的部分进行验证即可。

在社会、健康、安全、法律和文化方面，本系统也做出了努力。

在社会层面上，本系统实现了商品在各家门店间高效流通，一定程度上提高了工作效率，同时也减少了顾客等待的时间，提高了服务的质量，有利于社会进步和发展。

在健康层面上，经过严格的自我审查，本系统的设计以及实现中没有包含任何有害于身心健康的内容和功能。

在安全层面上，本系统在设计和实现过程中使用了用户身份验证、用户权限分级、视图、事务、触发器、数据库管理人员权限控制以及错误检查与提示机制，从客户端到数据库服务器端，对安全性实现了多方面、多层次的保护。

在法律层面上，本系统能够严格遵守法律法规，知法守法，在内容和功能上均做到合理合法，在设计和实现过程中使用的工具软件均为正版或开源免费版本，且本系统仅用作学习使用，不存在侵权行为。

在文化层面上，本系统能够一定程度上推动社会发展，为社会创造价值，传达科技改变生活的正能量，为人们的生活提供便利，从而促进文化发展。

我深知一名工程师在进行设计和实现设计的过程中需要在社会、健康、安全、法律和文化方面承担的重要责任，在实际工程中，这些方面都是需要仔细去考虑和斟酌的，往往会成为系统设计和实现过程中的制约因素，必须有所考量。

5 课程总结

本次课程实践过程中，我主要跟随任务书的描述完成了关系数据库管理系统软件功能的学习、SQL 语句练习以及一个综合实践应用系统的设计与实现。

在软件功能的学习部分，我选取 SQL Server 2008 R2 作为本次实践过程中使用的关系数据库软件，它配套的管理软件为 SQL Server Management Studio。在跟随指导书完成了数据库备份和增加用户以及配置权限等操作之后，我对 SQL Server Management Studio 的使用有了初步的理解，并且将备份和用户及权限配置等操作应用到了综合实践任务中，为后续的工作打下了基础。

在 SQL 练习部分，我以“疫情期间乘坐列车”为主题，从建表、数据导入、数据更新到最后的查询任务，逐个完成任务书中的要求。在这一过程中，我练习了使用 SQL 语句进行表的创建、删除、修改操作，以及对于特定数据表的查询、插入、删除、更新操作，还练习了触发器的编写，最后使用 SQL 完成的多个复杂查询任务充分锻炼了我使用 SQL 语句满足实际需求的能力，让我对 SQL 语句的运用更加熟悉。

在综合实践任务的设计和实现过程中，我从需求分析开始，历经总体设计、数据库设计与实现、客户端设计与实现以及最终测试环节，最终完成了一个以数据库服务器作为服务端的 C/S 架构的连锁品牌零售店库存管理系统。这一过程系统地锻炼了我分析问题和解决问题的能力。在这个过程中，我明白了，面对一个实际问题，可以通过系统的方式，一步步使用科学的分析方法和合理的工具进行设计和实现。整个任务的实现运用了数据库相关的多种技术，包括 SQL 语句、视图、事务、触发器、数据库用户管理和权限控制。运用这些技术，不仅能够使系统的实现过程更加简洁明了，还能显著提高系统的健壮性，保障数据以及数据库的安全性。通过本次实践，我还明白了一个好的工程不仅要在功能上满足要求，还要在社会、健康、安全、法律及文化方面达到要求。

总体而言，本次数据库课程实践始终围绕着数据库的原理展开应用和实践，让我对数据库的相关知识有了实践层面上的更深刻理解。这学期的课由于疫情只能线上开展，但老师依然能够在疫情期间为我们整理完备的学习资源、坚持用视频会议的形式讲课，还在课后 QQ 群里为我们答疑解惑，非常感谢老师的付出！

在本次课程中，我仍有待完善的工作，比如尚未对系统的查询性能作出分析，在综合实践任务中只进行了本地测试，没有考虑连接到互联网的情况等等。后续有时间希望能够对待完善的工作进行补足。

附录

附录中节选了一些关键代码，包括主函数入口、登录模块主要函数、QT 中对 Table View 和 Sql Table Model 的使用、QT 中 Table Widget 的使用、QT 在表格中添加按钮、QT 中实现 SQL 查询、QT 中实现事务机制。

(1) 主函数入口：

```
int main(int argc, char *argv[]) {
    QApplication a(argc, argv);
    //加载翻译文件
    QTranslator translator;
    translator.load("qt_zh_CN.qm");
    a.installTranslator(&translator);
    //连接数据库
    if (!OpenDatabase()) return 1;
    //创建窗口对象
    MainWindow w;
    login l;
    w.setWindowTitle(QStringLiteral("LEGO 零售店库存管理系统"));
    l.setWindowTitle(QStringLiteral("登录系统"));
    //显示 login 窗口
    l.show();
    //槽函数连接登录窗口和主窗口, login 界面登录成功后唤醒 mainwindow
    QObject::connect(&l, SIGNAL(logged()), &w, SLOT(on_logged()));
    return a.exec();
}
```

(2) 登录模块主要函数：

```
void login::on_buttonBox_accepted() {
    QString inputEID = ui->inputEID->text();
    QString inputPWD = ui->inputPWD->text();
    if (inputEID == "" || inputPWD == "") {
        QMessageBox::warning(nullptr, QStringLiteral("警告"),
                               QStringLiteral("工号和密码需填写完整"),
                               QMessageBox::Ok, QMessageBox::Ok);
        this->show();
    }
}
```

```

        return;
    }
    bool isIntEID;
    int intEID = inputEID.toInt(&isIntEID, 10);
    if (!isIntEID) {
        QMessageBox::warning(nullptr, QStringLiteral("警告"),
            QStringLiteral("工号需为数字"), QMessageBox::Ok,
                QMessageBox::Ok);

        this->show();
        return;
    }
    QSqlQuery query; //查询
    QString sqlStr =
        QString("select * from qbb.Employee where EID = %1").arg(intEID);
    query.exec(sqlStr);
    qDebug() << sqlStr;
    while (query.next()) {
        qDebug() << query.value(0);
        // 登陆成功
        if (inputPWD == query.value(3).toString()) {
            loggedEmp.EID = intEID;
            loggedEmp.PWD = inputPWD;
            loggedEmp.SID = query.value(1).toInt();
            loggedEmp.EType = query.value(2).toInt();
            loggedEmp.ENAME = query.value(4).toString();
            loggedEmp.logged = true;
            emit logged();
            return;
        }
    }
    // 登陆失败
    QMessageBox::warning(nullptr, QStringLiteral("警告"),
        QStringLiteral("用户名或密码错误"), QMessageBox::Ok,
            QMessageBox::Ok);

    this->show();

```



```

return;
}

```

(3) QT 中对 Table View 和 Sql Table Model 的使用:

```

void MainWindow::show_tab1() {
    qDebug() << QStringLiteral("切换到 tab1");
    ReadOnlyDelegate *readOnlyDelegate = new ReadOnlyDelegate();
    //仅店主可增删员工
    if (login::loggedEmp.EType != 1) {
        ui->groupBox_12->setVisible(false); //按钮不可见
    }
    if (model == nullptr) model = new QSqlTableModel(this);
    model->setTable("qbb.Employee");
    model->setEditStrategy(QSqlTableModel::OnManualSubmit);
    //    仅店主可查看所有信息
    if (login::loggedEmp.EType != 1)
        model->setFilter(QString("SID=%1 and EID=%2")
                           .arg(login::loggedEmp.SID)
                           .arg(login::loggedEmp.EID));
    else
        model->setFilter(QString("SID=%1").arg(login::loggedEmp.SID));
    model->setSort(0, Qt::AscendingOrder); //按工号升序
    model->select();
    model->setHeaderData(0, Qt::Horizontal, QStringLiteral("工号"));
    model->setHeaderData(1, Qt::Horizontal, QStringLiteral("所属门店号"));
    model->setHeaderData(2, Qt::Horizontal, QStringLiteral("员工类型码"));
    model->setHeaderData(3, Qt::Horizontal, QStringLiteral("密码"));
    model->setHeaderData(4, Qt::Horizontal, QStringLiteral("名字"));
    //    不可编辑列
    ui->tableView->setItemDelegateForColumn(0, readOnlyDelegate);
    ui->tableView->setItemDelegateForColumn(1, readOnlyDelegate);
    if (login::loggedEmp.EType != 1)
        ui->tableView->setItemDelegateForColumn(2, readOnlyDelegate);
    //    宽度自适应填满
    ui->tableView->horizontalHeader()->setSectionResizeMode(

```

```

        QHeaderView::Stretch);
//    颜色交错
ui->tableView->setAlternatingRowColors(true);
//    设置 model
ui->tableView->setModel(model);
ui->tableView->show();
}

```

(4) QT 中 Table Widget 的使用：

```

//设置表格
ui->table2->setColumnCount(8);
QStringList headers; //表头
headers << QStringLiteral("货号") << QStringLiteral("品名")
        << QStringLiteral("积木块数") << QStringLiteral("适合年龄")
        << QStringLiteral("单价") << QStringLiteral("折扣")
        << QStringLiteral("所在门店") << QStringLiteral("数量");
ui->table2->setHorizontalHeaderLabels(headers);
ui->table2->horizontalHeader()->setSectionResizeMode(
        QHeaderView::Stretch); //填满
//不可编辑
ui->table2->setEditTriggers(QAbstractItemView::NoEditTriggers);
//填写表格
int nRow, nCol;
query.last();
nRow = query.at() + 1;
query.first();
qDebug() << "nRow:" << nRow;
ui->table2->setRowCount(nRow);
nCol = ui->table2->columnCount();
for (int i = 0; i < nRow; i++) {
    qDebug() << query.value(0).toInt();
    for (int j = 0; j < nCol; j++) {
        ui->table2->setItem(
            i, j, new QTableWidgetItem(query.value(j).toString()));
        ui->table2->item(i, j)->setTextAlignment(Qt::AlignHCenter |

```

```

Qt::AlignVCenter);
    }
    query.next();
}

```

(5) QT 在表格中添加按钮:

//填写表格

```

    int nRow, nCol;
    query.last();
    nRow = query.at() + 1;
    query.first();
    qDebug() << "nRow:" << nRow;
    ui->table6_1->setRowCount(nRow);
    nCol = ui->table6_1->columnCount();
    for (int i = 0; i < nRow; i++) {
        for (int j = 0; j < nCol - 1; j++) {
            //倒数第三列为时间
            if (j == nCol - 3) {
                ui->table6_1->setItem( i, j,
new QTableWidgetItem(query.value(j).toDate().toString(
                "yyyy-MM-dd hh:mm:ss"))));
            }
            //倒数第二列添加状态
            else if (j == nCol - 2) {
                QString statusStr;
                int statusInt = query.value(j).toInt();
                if (statusInt == 0) {
                    statusStr = QStringLiteral("同意");
                } else if (statusInt == 1) {
                    statusStr = QStringLiteral("拒绝");
                } else {
                    statusStr = QStringLiteral("等待");
                }
                ui->table6_1->setItem(i, nCol - 2,

```

```

        new QTableWidgetItem(statusStr));
    }
    //其余列正常
    else {
        ui->table6_1->setItem(
            i, j, new QTableWidgetItem(query.value(j).toString()));
    }
    ui->table6_1->item(i, j)->setTextAlignment(Qt::AlignHCenter |
                                                Qt::AlignVCenter);
}
//最后一列添加按钮
QPushButton *pBtn = new QPushButton();
//分情况讨论按钮状态和行为
QString btnTextStr;
int statusInt = query.value(nCol - 2).toInt();
if (statusInt == 0) {
    btnTextStr = QStringLiteral("确认");
    connect(pBtn,          SIGNAL(clicked()),          this,
    SLOT(on_btn_6_1_ok()));
} else if (statusInt == 1) {
    btnTextStr = QStringLiteral("删除");
    connect(pBtn,          SIGNAL(clicked()),          this,
    SLOT(on_btn_6_1_delete()));
} else {
    btnTextStr = QStringLiteral("取消");
    connect(pBtn,          SIGNAL(clicked()),          this,
    SLOT(on_btn_6_1_cancel()));
}
pBtn->setText(btnTextStr);
ui->table6_1->setCellWidget(i, nCol - 1, pBtn);
//下一条 query 结果
query.next();
}

```

(6) QT 中实现 SQL 查询:

```

//进行查询
QSqlQuery query;      //查询
QSqlQuery queryRepo;  //查询库存
QString sqlStr;
sqlStr = QString(
    "select TID,JID,JName,Price,FSID,FSName,FCityName,
    TSID,TSName,TCityName,TNum,TDate ");
sqlStr.append("from TransferDetail ");
sqlStr.append(QString("where TSID=%1 ")
    .arg(login::loggedEmp.SID)); //收到申请的为本店
sqlStr.append(QString("and TStatus='%1'").arg('2')); //处于等待状态
qDebug() << sqlStr;
query.exec(sqlStr);

```

(7) QT 中实现事务机制:

```

//开始事务
query.exec("begin transaction");
bool ok1 =
query.exec(QString("select * from qbb.Repo where JID=%1 and SID=%2")
    .arg(JID).arg(SID));

//本店有库存
QString msg;
bool ok2;
if (query.first()) {
    qDebug()<<"Has Repo";
    ok2 = query.exec(
        QString("update qbb.Repo set RNum=RNum+%1 where JID=%2
        and SID=%3").arg(TNum).arg(JID).arg(SID));
    msg=QStringLiteral("调货成功! 店内有库存, 直接更新库存数目");
}
//本店无库存
else {
    qDebug()<<"No Repo";
    ok2 = query.exec(QString("insert into qbb.Repo values (%1,%2,%3)")
        .arg(SID).arg(JID).arg(TNum));
}

```

```

        msg = QStringLiteral("调货成功！店内无库存，插入新库存项");
    }
    //删除调货记录
    bool ok3 =
        query.exec(QString("delete      from      qbb.Transfer      where
TID=%1").arg(TID));
    if (ok1 && ok2 && ok3) {
        query.exec("commit");
        QMessageBox::information(nullptr, QStringLiteral("完成"), msg,
                                QMessageBox::Ok, QMessageBox::Ok);
    } else {
        query.exec("rollback");
        QMessageBox::warning(nullptr, QStringLiteral("警告"),
                             QStringLiteral("出现错误，已回滚"),
                             QMessageBox::Ok, QMessageBox::Ok);
    }
    //结束事务

```