

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 综合设计题目： | 仓储管理系统 |
| 姓 名： | 薛旭升 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1606 |
| 学 号： | U201614683 |
| 指导教师： | 潘鹏 |

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

2019 年5月21日

**任务书**

**1 Sql练习部分（必做题）**

**1.1 建表**

1）假设某微博平台的数据库中有下列关系，请在DBMS中创建这些关系（请先建立自己的数据库（数据库名字：学生姓＋学号最后3位），包括主码和外码的说明，并写出指定关系的建表SQL语句：

用户【用户ID，姓名，性别，出生年份，所在城市】记录所有注册用户的基本信息，英文表名和字段名如下：

*USER(UID 整型，NAME 字符串，SEX 一位汉字，BYEAR 整型，CITY 字符串 )；*

分类【分类ID，分类名称】记录微博平台中所有可能涉及的微博的类型，例如文学、艺术、军事等，英文表名和字段名如下：

*LABEL(LID 整型，LNAME 字符串)；*

博文【博文ID，标题，用户ID，年，月，日，正文】记录每一篇微博的基本信息，英文表名和字段名如下：

*MBLOG(BID 整型，TITLE 字符串，UID 整型，PYEAR 整型，PMONTH 整型， PDAY 整型，CONT 字符串)，*

博文标注【博文ID，分类ID】记录每一篇微博的作者给该微博贴上的分类标签，一篇微博可以涉及不止一种分类，英文表名和字段名如下：

*B\_L(BID 整型，LID 整型)；*

关注【用户ID，被关注用户ID】记录每位用户关注的其他用户，每位用户可关注多人，英文表名和字段名如下：

*FOLLOW(UID 整型，UIDFLED 整型)；*

好友【用户ID， 好友ID】记录每位用户的好友（可多个），英文表名和字段名如下：

*FRIENDS(UID 整型，FUID 整型);*

订阅【用户ID, 订阅分类ID】记录用户订阅的每一种分类，英文表名和字段名如下：

*SUB(UID 整型，LID 整型);*

点赞【用户ID， 博文ID】记录用户点赞的每一篇微博，英文表名和字段名如下：

*THUMB(UID 整型，BID 整型)，*

头条【年，月，日，博文ID，顺序号】记录每一天的热度排名前十的博文ID号以及该博文在热度前十名中的排名，英文表名和字段名如下：

*TOPDAY(TYEAR整型，TMONTH 整型，TDAY 整型，BID 整型，TNO 整型)。*

**[选做题]** 如果允许对一条博文进行多个回复（每个回复有发表作者、发表日期、内容），也允许对一条博文的回复进行回复，并允许对回复进行点赞，请问是否要增加表？若增加表，要增加几张？或者仅需要对原有的某个／些表进行修改？请给出相关表定义（表名和属性名自己定义）。

**2）观察性实验**

用户在订阅分类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数据更新**

**1）分别用一条sql语句完成对博文表基本的增、删、改的操作；**

**2）批处理操作**

将关注3号用户的用户信息插入到一个自定义的新表FANS\_3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1.1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。

在后续的上机实验环节，通过导入导出或者备份机制实现前次上机环节的数据恢复。

**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）触发器实验**

编写一个触发器，用于实现对点赞表的完整性控制规则：当点赞博文时，如果博文作者就是点赞者本人，则拒绝执行；而且一个人对一篇博文只能点赞一次。

**1.3查询**

请分别用一条SQL语句完成下列各个小题的需求：

1）查询“张三”用户关注的所有用户的ID号、姓名、性别、出生年份，所在城市，并且按照出生年份的降序排列，同一个年份的则按照用户ID号升序排列。

2）查找没有被任何人点赞的博文ID、标题以及发表者姓名，并将结果按照标题字符顺序排列。

3）查找2000年以后出生的武汉市用户发表的进入过头条的博文ID；

4）查找订阅了所有分类的用户ID；

5）查找出生年份小于1970年或者大于2010年的用户ID、出生年份、所在城市，要求where子句中只能有一个条件表达式；

6）统计每个城市的用户数；

7）统计每个城市的每个出生年份的用户数，并将结果按照城市的升序排列，同一个城市按照出生用户数的降序排列其相应的年份；

8）查找被点赞数超过10的博文ID号；

9）查找被2000年后出生的用户点赞数超过10的博文ID号；

10）查找被2000年后出生的用户点赞数超过10的每篇博文的进入头条的次数；

11）查找订阅了文学、艺术、哲学、音乐中至少一种分类的用户ID，要求不能使用嵌套查询，且where子句中最多只能包含两个条件；

12）查找标题中包含了“最多地铁站”和“\_华中科技大学”两个词的博文基本信息；

13）查找所有相互关注的用户对的两个ID号，要求不能使用嵌套查询；

14）查找好友圈包含了5号用户好友圈的用户ID；

15）查找2019年4月20日每一篇头条博文的ID号、标题以及该博文的每一个分类ID，要求即使该博文没有任何分类ID也要输出其ID号、标题；

16）查找至少有3名共同好友的所有用户对的两个ID号。

17）创建视图：查阅DBMS内部函数，创建一个显示当日热度排名前十的微博信息的视图，其中的属性包括：博文ID、博文标题、发表者ID、发表者姓名、被点赞数。

**[选做题]**

18）找出对用户“张三”撰写的每篇博文回复最多的用户ID，按回复次数的多少列出前3位（假设“张三”的博文不止3篇）；

19）找出对用户“张三”撰写的博文的回复中最具有热点的（点赞最多）的回复帖子，列出该博文ID，博文点赞数、最热回复帖的作者、该热帖内容 及 点赞数。

**1.4 了解系统的查询性能分析功能（选做）**

选择上述2.3任务中较为复杂的SQL语句（9，10，11，14，16），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。

**1.5 DBMS函数及存储过程和事务（选做）**

1）编写一个依据用户ID号计算其发表的博文进入头条的累计天数的DBMS自定义函数，并利用其查询2000年后出生的用户上述头条累计天数达到100天的所有用户ID。

2）建立关系“点赞排行榜【博文ID，当天点赞人数】”，里面存储系统当天点赞数前十名的博文ID及其点赞人数，尝试编写一个DBMS的存储过程，通过该存储过程更新该表。

3）尝试在DBMS的交互式界面中验证事务机制的执行效果。

**2 软件功能学习部分（必做题）**

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）

**2.1 备份功能**

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

**2.2 用户授权与验证**

在教材上的学生-选课数据库（包括：学生表、课程表、选课表）上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

3类用户：学生、教师和管理员。

-管理员：负责整个数据库的建表、管理和用户授权操作，并负责课程表的设置和学生的录入及修改；

-学生：被授权学生可以查看课程表和选课表，可以进行选课操作；

-教师：被授权教师可以查看学生表、课程表和选课表，可以修改选课表（只能输入/修改成绩）。

请给出相关的授权语句，并给出权限验证语句及验证授权是否有效截图。、

**3 数据库应用系统设计（必做）**

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。

**题目8：仓储管理系统**

实现一个仓库库存货品信息管理系统软件。仓库的日常工作包括货品的入库和出库。入库要由采购人员提供进货单，进货单经过审核人员审核验收后方能进行货品入库。出库要由销售人员提供出货单，经过审核人员审核批准后才能提货。当销售人员需要提货而货品的库存量不足时可先进行缺货登记，当有相应货品入库时，按缺货登记时间顺序处理出货请求。要求：

* 能实现库存货品信息的管理，货品的相关信息包括：货品号、货品名、存放地、货品库存量、生产厂家等。
* 实现进货单的填写、修改、审核和查询等功能。
* 实现出货单的填写、修改、审核和查询等功能。
* 实现缺货登记、查询和处理。
* 根据不同用户身份提供不同的操作权限和界面。

设计一个B/S或C/S模式的系统实现上述功能。

**目 录**

[1. 课程任务概述 1](#_Toc10741)

[2. Sql练习部分 2](#_Toc29548)

[2.1. 任务要求 2](#_Toc13635)

[2.2. 完成过程 5](#_Toc28083)

[2.3. 任务总结 15](#_Toc119)

[3. 软件功能学习部分 16](#_Toc12838)

[3.1. 任务要求 16](#_Toc17559)

[3.2. 完成过程 16](#_Toc9843)

[3.3. 任务总结 22](#_Toc17985)

[4. 综合实践任务 23](#_Toc9187)

[4.1. 系统设计目标 23](#_Toc27722)

[4.2. 需求分析 23](#_Toc26075)

[4.3. 总体设计 28](#_Toc23750)

[4.4. 数据库设计 28](#_Toc31070)

[4.5. 详细设计与实现 30](#_Toc23295)

[4.6. 系统测试 36](#_Toc7572)

[4.7. 系统设计与实现总结 45](#_Toc13913)

[5. 课程总结 46](#_Toc17835)

[附录一 建表语句 47](#_Toc22850)

[附录二 仓储管理系统主功能代码 50](#_Toc11910)

# 课程任务概述

本次实验共分为3个部分：Sql练习部分、软件功能学习部分、综合设计部分。

Sql练习部分主要用于帮助熟悉SQL语句，与书本内容结合，完成实验。该部分涉及到的SQL内容有：表的创建、表的插入、查询、更新、删除、创建视图、创建触发器、表的完整性。该部分实验要求所有内容使用SQL语句实现，不能使用软件自带的功能。

软件功能学习部分主要通过用户授权操作了解软件架构。该实验涉及到的知识有：数据的导入导出、脱机备份与恢复、登录名、用户名、角色名等。该实验只要求授权语句必须通过SQL语句实现。

综合实验部分主是对数据库的设计及应用的开发。根据选定的题目，学生需要设计处完整的数据库，需要对数据库进行需求分析，做出数据字典，再根据数据字典得到E-R图。除此以外，学生还需要进行应用开发，对题目进行功能需求分析，得到功能模块图，然后做出一个交互性较好的图形界面系统。

# Sql练习部分

## 任务要求

### 建表

1）假设某微博平台的数据库中有下列关系，请在DBMS中创建这些关系（请先建立自己的数据库（数据库名字：学生姓＋学号最后3位），包括主码和外码的说明，并写出指定关系的建表SQL语句：

用户【用户ID，姓名，性别，出生年份，所在城市】记录所有注册用户的基本信息，英文表名和字段名如下：

*USER(UID 整型，NAME 字符串，SEX 一位汉字，BYEAR 整型，CITY 字符串 )；*

**请给出此关系的建表SQL语句；**

分类【分类ID，分类名称】记录微博平台中所有可能涉及的微博的类型，例如文学、艺术、军事等，英文表名和字段名如下：

*LABEL(LID 整型，LNAME 字符串)；*

博文【博文ID，标题，用户ID，年，月，日，正文】记录每一篇微博的基本信息，英文表名和字段名如下：

*MBLOG(BID 整型，TITLE 字符串，UID 整型，PYEAR 整型，PMONTH 整型， PDAY 整型，CONT 字符串)，*

**请给出此关系的建表SQL语句；**

博文标注【博文ID，分类ID】记录每一篇微博的作者给该微博贴上的分类标签，一篇微博可以涉及不止一种分类，英文表名和字段名如下：

*B\_L(BID 整型，LID 整型)；*

关注【用户ID，被关注用户ID】记录每位用户关注的其他用户，每位用户可关注多人，英文表名和字段名如下：

*FOLLOW(UID 整型，UIDFLED 整型)；*

好友【用户ID， 好友ID】记录每位用户的好友（可多个），英文表名和字段名如下：

*FRIENDS(UID 整型，FUID 整型);*

订阅【用户ID, 订阅分类ID】记录用户订阅的每一种分类，英文表名和字段名如下：

*SUB(UID 整型，LID 整型);*

点赞【用户ID， 博文ID】记录用户点赞的每一篇微博，英文表名和字段名如下：

*THUMB(UID 整型，BID 整型)，*

头条【年，月，日，博文ID，顺序号】记录每一天的热度排名前十的博文ID号以及该博文在热度前十名中的排名，英文表名和字段名如下：

*TOPDAY(TYEAR整型，TMONTH 整型，TDAY 整型，BID 整型，TNO 整型)。*

**请给出此关系的建表SQL语句；**

**[选做题]** 如果允许对一条博文进行多个回复（每个回复有发表作者、发表日期、内容），也允许对一条博文的回复进行回复，并允许对回复进行点赞，请问是否要增加表？若增加表，要增加几张？或者仅需要对原有的某个／些表进行修改？**请给出相关表定义（表名和属性名自己定义）。**

**2）观察性实验**

用户在订阅分类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。

### 数据更新

1）分别用一条sql语句完成对博文表基本的增、删、改的操作；

2）批处理操作

将关注3号用户的用户信息插入到一个自定义的新表FANS\_3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1.1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。

在后续的上机实验环节，通过导入导出或者备份机制实现前次上机环节的数据恢复。

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）触发器实验

编写一个触发器，用于实现对点赞表的完整性控制规则：当点赞博文时，如果博文作者就是点赞者本人，则拒绝执行；而且一个人对一篇博文只能点赞一次。

### 查询

请分别用一条SQL语句完成下列各个小题的需求：

1）查询“张三”用户关注的所有用户的ID号、姓名、性别、出生年份，所在城市，并且按照出生年份的降序排列，同一个年份的则按照用户ID号升序排列。

2）查找没有被任何人点赞的博文ID、标题以及发表者姓名，并将结果按照标题字符顺序排列。

3）查找2000年以后出生的武汉市用户发表的进入过头条的博文ID；

4）查找订阅了所有分类的用户ID；

5）查找出生年份小于1970年或者大于2010年的用户ID、出生年份、所在城市，要求where子句中只能有一个条件表达式；

6）统计每个城市的用户数；

7）统计每个城市的每个出生年份的用户数，并将结果按照城市的升序排列，同一个城市按照出生用户数的降序排列其相应的年份；

8）查找被点赞数超过10的博文ID号；

9）查找被2000年后出生的用户点赞数超过10的博文ID号；

10）查找被2000年后出生的用户点赞数超过10的每篇博文的进入头条的次数；

11）查找订阅了文学、艺术、哲学、音乐中至少一种分类的用户ID，要求不能使用嵌套查询，且where子句中最多只能包含两个条件；

12）查找标题中包含了“最多地铁站”和“\_华中科技大学”两个词的博文基本信息；

13）查找所有相互关注的用户对的两个ID号，要求不能使用嵌套查询；

14）查找好友圈包含了5号用户好友圈的用户ID；

15）查找2019年4月20日每一篇头条博文的ID号、标题以及该博文的每一个分类ID，要求即使该博文没有任何分类ID也要输出其ID号、标题；

16）查找至少有3名共同好友的所有用户对的两个ID号。

17）创建视图：查阅DBMS内部函数，创建一个显示当日热度排名前十的微博信息的视图，其中的属性包括：博文ID、博文标题、发表者ID、发表者姓名、被点赞数。

**[选做题]**

18）找出对用户“张三”撰写的每篇博文回复最多的用户ID，按回复次数的多少列出前3位（假设“张三”的博文不止3篇）；

19）找出对用户“张三”撰写的博文的回复中最具有热点的（点赞最多）的回复帖子，列出该博文ID，博文点赞数、最热回复帖的作者、该热帖内容 及 点赞数。

### 了解系统的查询性能分析功能（选做）

选择上述2.3任务中较为复杂的SQL语句（9，10，11，14，16），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。

### DBMS函数及存储过程和事务（选做）

1）编写一个依据用户ID号计算其发表的博文进入头条的累计天数的DBMS自定义函数，并利用其查询2000年后出生的用户上述头条累计天数达到100天的所有用户ID。

2）建立关系“点赞排行榜【博文ID，当天点赞人数】”，里面存储系统当天点赞数前十名的博文ID及其点赞人数，尝试编写一个DBMS的存储过程，通过该存储过程更新该表。

3）尝试在DBMS的交互式界面中验证事务机制的执行效果。

## 完成过程

### 建表

1）首先通过Microsoft SQL Server Management Studio软件连接服务器，连接后建立一个数据库，建立数据库命令如下：

CREATE DATABASE [薛683];

2）接着在建立的数据库[薛683]中创建任务要求的表，建表SQL语句见附录一。创建表后结果如图 2.1所示。

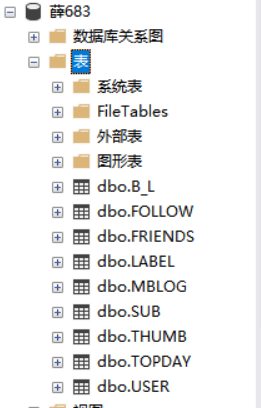


图 2.1 初步建立的数据库表项

1. 建表后可插入内容，执行插入语句后可以通过查询语句查看表内容，以USER表为例，其查询结果如图 2.2所示。



图 2.2 USER表部分内容

1. 测试观察性实验，该观察性实验是对参照完整性的测试，已知分类的范围是1~11，所以我们尝试订阅分类12，插入语句为：

INSERT [SUB] VALUES (1,12);

执行结果如图 2.3所示，由于违背了参照完整性，所以数据库拒绝插入。

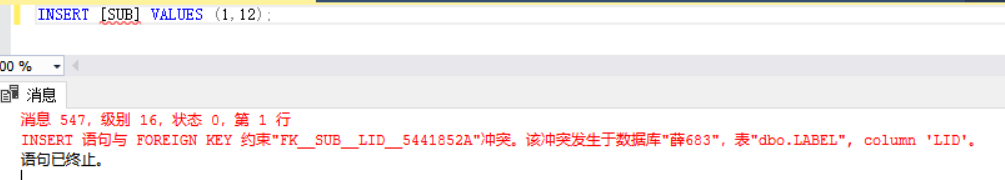


图 2.3 参考完整性测试

### 数据更新

1）首先创建一个新表FANS\_3，其SQL语句见附录一，接着将将关注3号用户的用户信息插入到新表FANS\_3，插入语句如下：

INSERT INTO [FAN\_3]

SELECT \* FROM [USER]

WHERE UID IN (

SELECT UID

FROM [FOLLOW]

WHERE UIDFLED=3);

该语句的子查询表示从关注表FOLLOW中筛选出3号用户的粉丝的UID，接着根据得到的UID从USER表中中调取3号用户的粉丝的信息插入到新表FAN\_3中。

执行结果如图 2.4所示，与关注表比较可知结果正确。



图 2.4 粉丝表插入结果

2）测试观察性实验，新建一个无主码的表TEST，其SQL语句为：

CREATE TABLE [TEST] (

NO INT,

NAME CHAR(6)

);

建表成功后直接通过图形化交互界面操作插入两个相同数据（1，李明），插入结果如图 2.5所示，从图中可以看出，虽然会警告，但数据插入成功，即无主码的表可以插入重复元组。接着尝试删除重复元组，删除结果如图 2.6所示，由于会删除多行，所以数据库拒绝删除。

从该观察性实验可以得出：每个表设置一个主码是很有必要的，否则影响使用时的功能实现。



图 2.5 表TEST插入结果

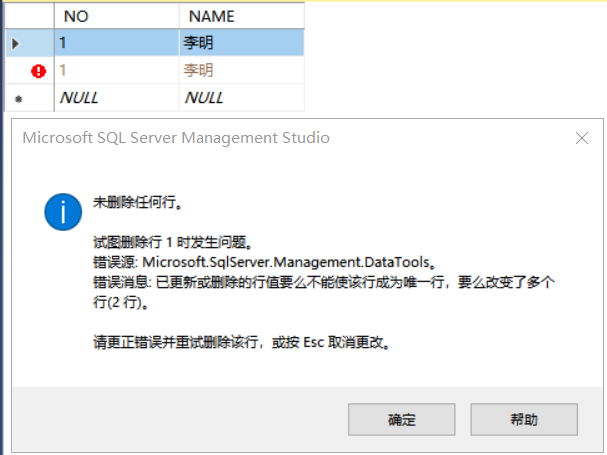


图 2.6 表TEST删除结果

3）编写触发器，触发器创建语句为：

CREATE TRIGGER [THUMB\_CHECK]

ON [THUMB]

AFTER INSERT,UPDATE

AS

BEGIN

DELETE FROM [THUMB]

WHERE EXISTS ( --点赞两次

SELECT \*

FROM inserted X

WHERE X.UID=UID AND X.BID=BID

)AND UID= ( --自己点赞自己

SELECT UID

FROM [MBLOG]

WHERE UID=[THUMB].UID AND BID=[THUMB].BID

)

END;

该触发器的作用是：每次向点赞表插入数据或更新点赞表时进行检测，若出现以下两种情况：

①插入的数据的点赞人是点赞博文作者，即自己点赞自己

②插入的数据已在点赞表中存在，即点赞两次

此时将刚才插入的数据删除，该触发器通过删除插入的数据达到拒绝执行的任务要求。

触发器创建成功后进行测试，测试语句如下：

INSERT INTO [THUMB] VALUES (1,18);

INSERT INTO [THUMB] VALUES (1,1);

对于第一个语句，由于表中已存在数据，所以数据库拒绝插入，未激活触发器；而由于1号博文的作者是1号用户，所以第二条语句激活了触发器，执行后点赞表未改变，表示触发器起作用了。

### 查询

1）查询“张三”用户关注的所有用户的ID号、姓名、性别、出生年份，所在城市，并且按照出生年份的降序排列，同一个年份的则按照用户ID号升序排列。

解决思路：该查询需要用到子查询，先从FOLLOW表中筛选出“张三”用户关注的所有用户的ID号，接着从USER表中查询出这些ID的信息，最后需要使用ORDER进行排序。

|  |
| --- |
| SELECT UID,NAME,SEX,BYEAR,CITY  FROM [USER]  WHERE UID IN  (SELECT UIDFLED  FROM [FOLLOW],[USER]  WHERE [FOLLOW].UID=[USER].UID  AND [USER].NAME = '张三')  ORDER BY BYEAR DESC,UID; |

2）查找没有被任何人点赞的博文ID、标题以及发表者姓名，并将结果按照标题字符顺序排列。

解决思路：没被人点赞表示点赞表THUMB中没有记录，所以从子查询中列出THUMB中所有出现的BID，接着在主查询中使用“NOT IN”将子查询结果从MBLOG表中排除，剩余的都是没有被任何人点赞的博文，博文作者信息可以使用连接从USER表中获取。

|  |
| --- |
| SELECT BID, TITLE, NAME  FROM [MBLOG],[USER]  WHERE [MBLOG].UID=[USER].UID  AND BID NOT IN  (SELECT DISTINCT BID  FROM [THUMB])  ORDER BY TITLE; |

3） 查找2000年以后出生的武汉市用户发表的进入过头条的博文ID；

解决思路：TOPDAY表中的博文都是进入过头条的博文，所以该查询可以从表TOPDAY入手，通过连接MBLOG表和USER表来筛选出满足条件的，需要注意的是MBLOG表和TOPDAY表中都有博文ID，所以需要注明博文ID的表名。

|  |
| --- |
| SELECT DISTINCT [MBLOG].BID  FROM [MBLOG],[USER],[TOPDAY]  WHERE [USER].UID=[MBLOG].UID AND [TOPDAY].BID=[MBLOG].BID  AND BYEAR>2000 AND CITY='武汉'; |

4）查找订阅了所有分类的用户ID；

解决思路：订阅所有分类可以理解为：该用户订阅的分类数等于分类总数，所以可以使用GROUP将订阅表SUB根据UID分组，分组后对其进行数量统计，若数量与分类表LABEL中分类数相同则表示该用户订阅所有分类。

|  |
| --- |
| SELECT UID  FROM [SUB]  GROUP BY UID  HAVING COUNT(UID)=  (SELECT COUNT(\*)  FROM [LABEL]); |

5）查找出生年份小于1970年或者大于2010年的用户ID、出生年份、所在城市，要求where子句中只能有一个条件表达式；

解决思路：该查询只要求年份，但WHERE子句只能有一个条件表达式，所以不能使用比较，需要使用NOT BETWEEN。

|  |
| --- |
| SELECT UID,BYEAR,CITY  FROM [USER]  WHERE BYEAR NOT BETWEEN 1970 AND 2010; |

6）统计每个城市的用户数；

解决思路：该查询只需要统计，所以可以使用聚集函数COUNT根据城市分组后可直接得到结果。

|  |
| --- |
| SELECT CITY,COUNT(\*) '用户数'  FROM [USER]  GROUP BY CITY; |

7）统计每个城市的每个出生年份的用户数，并将结果按照城市的升序排列，同一个城市按照出生用户数的降序排列其相应的年份；

解决思路：该查询与第6题相同，只是需要根据城市和年份两项分组，并且需要使用ORDER将结果排序。

|  |
| --- |
| SELECT CITY,BYEAR,COUNT(\*) '用户数'  FROM [USER]  GROUP BY CITY,BYEAR  ORDER BY CITY,BYEAR DESC; |

8）查找被点赞数超过10的博文ID号；

解决思路：该查询需要对统计结果进行比较，只是WHERE子句中不能使用聚集函数，所以需要使用HAVING子句进行筛选，只显示点赞数超过10的博文。

|  |
| --- |
| SELECT BID,COUNT(BID) '点赞数'  FROM [THUMB]  GROUP BY BID  HAVING COUNT(BID)>10; |

9）查找被2000年后出生的用户点赞数超过10的博文ID号；

解决思路：该查询时对第8题进行补充加强，所以只需要再第8题的基础上加一个对USER表的连接，并增加条件点赞用户出生年大于2000。

|  |
| --- |
| SELECT BID,COUNT(BID) '点赞数'  FROM [THUMB],[USER]  WHERE [THUMB].UID=[USER].UID  AND [USER].BYEAR>2000  GROUP BY BID  HAVING COUNT(BID)>10; |

10）查找被2000年后出生的用户点赞数超过10的每篇博文的进入头条的次数；

解决思路：该题是对第9题查询结果的应用，所以将第九题查询语句作为子查询，统计博文在头条表TOPDAY中出现的次数。

|  |
| --- |
| SELECT BID,COUNT(BID) '进入头条数'  FROM [TOPDAY]  WHERE BID IN (  SELECT BID  FROM [THUMB],[USER]  WHERE [THUMB].UID=[USER].UID  AND [USER].BYEAR>2000  GROUP BY BID  HAVING COUNT(BID)>10)  GROUP BY BID; |

11）查找订阅了文学、艺术、哲学、音乐中至少一种分类的用户ID，要求不能使用嵌套查询，且where子句中最多只能包含两个条件；

解决思路：该查询只需要判断分类，但有4种情况，而WHERE子句最多只能包含2个条件，所以可以使用确定集合对分类进行判断。

|  |
| --- |
| SELECT DISTINCT UID  FROM [SUB],[LABEL]  WHERE [SUB].LID=[LABEL].LID AND  LNAME IN ('文学','艺术','哲学','音乐'); |

12）查找标题中包含了“最多地铁站”和“\_华中科技大学”两个词的博文基本信息；

解决思路：该查询只涉及到博文表MBOG的查找，查询包含规定字符串的元组可以使用LIKE字符匹配完成，需要注意的是字符中包含“\_”，这是通配符，所以需要使用换码字符“\”。

|  |
| --- |
| SELECT \*  FROM [MBLOG]  WHERE TITLE LIKE '%最多地铁站%' AND TITLE LIKE '%\\_华中科技大学%'; |

13）查找所有相互关注的用户对的两个ID号，要求不能使用嵌套查询；

解决思路：相互关注信息都在关注表FOLLOW中，所以该查询只使用关注表即可，但是该查询需要对关注表中两个不同元组进行比较，所以需要使用自身连接来达到目的。

|  |
| --- |
| SELECT X.UID '用户1', Y.UID '用户2'  FROM [FOLLOW] X, [FOLLOW] Y  WHERE X.UID=Y.UIDFLED AND Y.UID=X.UIDFLED; |

14）查找好友圈包含了5号用户好友圈的用户ID；

解决思路：该查询可以用逻辑蕴涵来表达：不存在这样的用户y，他是5号用户的好友，却不是用户z的好友。该查询涉及到了双重否定，需要使用NOT EXIST实现。

|  |
| --- |
| SELECT DISTINCT UID  FROM [FRIENDS] X  WHERE NOT EXISTS (  SELECT \*  FROM [FRIENDS] Y  WHERE UID=5  AND NOT EXISTS (  SELECT \*  FROM [FRIENDS] Z  WHERE Z.FUID=Y.FUID  AND Z.UID=X.UID)); |

15）查找2019年4月20日每一篇头条博文的ID号、标题以及该博文的每一个分类ID，要求即使该博文没有任何分类ID也要输出其ID号、标题；

解决思路：头条博文通过头条表TOPDAY即可获得，但是分类ID组要用到连接，而且没有分类也需要输出博文信息，所以需要使用外连接。

|  |
| --- |
| SELECT DISTINCT [TOPDAY].BID 'BID', TITLE, LID  FROM [TOPDAY],[MBLOG] LEFT OUTER JOIN [B\_L] ON ([MBLOG].BID=[B\_L].BID)  WHERE [TOPDAY].BID=[MBLOG].BID  AND TYEAR=2019 AND TMONTH=4 AND TDAY=20; |

16）查找至少有3名共同好友的所有用户对的两个ID号。

解决思路：与相互关注相似，共同好友需要对好友表FRIEND两个不同元组进行比较，所以需要对好友表进行自身连接，自身连接后的结果是有共同好友的用户对，题目中3名共同好友的要求可以通过对结果使用聚集函数COUNT统计实现。

|  |
| --- |
| SELECT X.UID '用户1', Y.UID '用户2'  FROM [FRIENDS] X, [FRIENDS] Y  WHERE X.FUID=Y.FUID AND X.UID<>Y.UID  GROUP BY Y.UID,X.UID  HAVING COUNT(\*)>=3; |

### 视图创建

视图的创建与查询相似，题目要求显示当日二位年度前十的微博信息，所以可以从头条表TOOPDAY入手，博文标题、发表者ID可通过与博文表连接获取，被点赞数可通过与点赞表连接获取，发表者姓名可以通过将博文表与用户表连接获取。除此之外，由于需要显示当日的微博，所以需要获取当日日期，可以使用DBMS内部函数DATENAME获取年月日后与头条表的年月日比较。代码如下：

|  |
| --- |
| CREATE VIEW NOWTOP AS  SELECT [MBLOG].BID, TITLE, [USER].UID, NAME, COUNT(\*) '被点赞数'  FROM [TOPDAY],[USER],[THUMB],[MBLOG]  WHERE [TOPDAY].TYEAR=DATENAME(YEAR,GETDATE()) AND  [TOPDAY].TMONTH=DATENAME(MONTH,GETDATE()) AND  [TOPDAY].TDAY=DATENAME(DAY,GETDATE()) AND  [TOPDAY].BID=[MBLOG].BID AND  [MBLOG].UID=[USER].UID AND  [TOPDAY].BID=[THUMB].BID  GROUP BY [MBLOG].BID, TITLE, [USER].UID, NAME; |

视图创建后可以在数据库的视图中找到，右键查找选择显示前1000行观察执行结果，如图 2.7所示，接着查询头条表，如图 2.8所示，将视图查询结果与头条表比较，两者博文号相同，结果正确。



图 2.7 视图查询结果



图 2.8 头条表部分内容

### 选做题

1)表的创建。为了实现回复功能，需要创建4个表：回复表、回复回复表、博文回复表、回复点赞表。

回复表用于记录每条回复的基本信息，其创建SQl语句如下：

CREATE TABLE [REPLY] (

RID INT PRIMARY KEY, --回复号

UID INT, --回复人

RYEAR INT, --年

RMONTH INT, --月

RDAY INT, --日

CONT VARCHAR(100) --内容

);

回复回复表用于记录每条回复的回复，其创建SQl语句如下：

CREATE TABLE [RREPLY] (

RID INT PRIMARY KEY, --回复号

RRID INT, --回复的回复号

FOREIGN KEY(RID) REFERENCES [REPLY](RID)

ON DELETE CASCADE,

FOREIGN KEY(RRID) REFERENCES [REPLY](RID)

ON DELETE CASCADE

);

博文回复表用于记录每条博文的回复，其创建SQl语句如下：

CREATE TABLE [MREPLY] (

BID INT PRIMARY KEY, --博文号

RID INT, --回复号

FOREIGN KEY(BID) REFERENCES [MBLOG](BID)

ON DELETE CASCADE,

FOREIGN KEY(RRID) REFERENCES [REPLY](RID)

ON DELETE CASCADE

);

回复点赞表用于记录用户点赞的每一条回复，其创建SQl语句如下：

CREATE TABLE [RTHUMB] (

UID INT , --点赞人

RID INT, --回复号

PRIMARY KEY(UID,RID)

FOREIGN KEY(UID) REFERENCES [USER](UID)

ON DELETE CASCADE,

FOREIGN KEY(RID) REFERENCES [REPLY](RID)

ON DELETE CASCADE

);

2）找出对用户“张三”撰写的每篇博文回复最多的用户ID，按回复次数的多少列出前3位（假设“张三”的博文不止3篇）；

解决方案：该查询可以分为三部分：先查找“张三”的每篇博文，再根据博文号查找出对每篇博文回复最多的用户，再从这些用户中找出回复次数最多的3个，SQL语句如下：

|  |
| --- |
| SELECT TOP (3) [USER].UID  FROM [USER],[MBLOG] X,[MREPIY],[REPLY]  WHERE [USER].UID=X.UID AND [USER].NAME='张三'  AND [REPLY].RID=[MREPIY].RID AND X.BID=[MREPIY].BID  AND [USER].UID IN (  /\* 博文Y回复最多的用户ID \*/  SELECT TOP (1) [REPLY].UID  FROM [MREPIY] Y,[REPLY]  WHERE X.BID=Y.BID AND [REPLY].RID=Y.RID  GROUP BY [REPLY].UID,[MREPIY].BID  ORDER BY COUNT([REPLY].UID) DESC  )  GROUP BY [USER].UID  ORDER BY COUNT([USER].UID) DESC |

3）找出对用户“张三”撰写的博文的回复中最具有热点的（点赞最多）的回复帖子，列出该博文ID，博文点赞数、最热回复帖的作者、该热帖内容及点赞数。

解决方案，该查询分为两部分：博文的点赞、最热帖的点赞。所以该查询可以分为两个子查询，一个子查询查点赞数，一个子查询查最热帖内容，随后将两个内容结合起来。SQL语句如下：

|  |
| --- |
| SELECT S1.X "博文ID",S2.N "点赞数",[USER].NAME "最热贴作者",S1.Z "热帖内容",S1.L "点赞数"  FROM [USER],  (SELECT TOP(1) [MBLOG].BID X,[REPLY].UID Y,[REPLY].CONT Z,COUNT(\*) L  FROM [USER],[MBLOG] X,[MREPIY],[RTHUMB],[REPLY]  WHERE [USER].NAME='张三' AND [USER].UID=[MBLOG].UID  AND [MBLOG].BID=[MREPLY].BID AND [MREPLY].RID=[RTHUMB].RID  AND [REPLY].RID=[MREPLY].RID  GROUP BY [MBLOG].BID,[MREPLY].RID) S1,  (SELECT BID M,COUNT(BID) N  FROM [THUMB],  GROUP BY BID) S2  WHERE [USER].UID=S1.Y AND S1.X=S2.M; |

## 任务总结

本次实验任务量比较大，但难度较小，所以虽然完成的较慢，但遇到的问题并不多，最主要的问题就是对软件的使用，由于第一次使用，所以很多功能并不是很熟悉，包括对各个表的操作、数据的导出备份等，由于需要用到Microsoft SQL Server Management Studio的功能并不多，所以通过本次实验对该软件的基本功能都已经熟悉。除此以外，该实验中的触发器比较难写，尤其是SQL Server中定义触发器的语句与教材中不同，所以写触发器时一直提示有错，最后查询网上资料，发现网上该处的触发器定义语句与教材不同，而后参考同学的结果才学出触发器。除了以上两点，本次实验花费事件多一点的就是双重否定查询，这也时我当前接触的SQL语句中最难理解的语句，不过之前写作业时遇到过一次双重否定问题，所以此次实验中解决的较快，并未思考特别长时间。

# 软件功能学习部分

## 任务要求

### 备份功能

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

### 用户授权与验证

在教材上的学生-选课数据库（包括：学生表、课程表、选课表）上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

3类用户：学生、教师和管理员。

-管理员：负责整个数据库的建表、管理和用户授权操作，并负责课程表的设置和学生的录入及修改；

-学生：被授权学生可以查看课程表和选课表，可以进行选课操作；

-教师：被授权教师可以查看学生表、课程表和选课表，可以修改选课表（只能输入/修改成绩）。

请给出相关的授权语句，并给出权限验证语句及验证授权是否有效截图。

## 完成过程

### 备份功能

1）如图 3.1所示，鼠标选中的功能即为脱机备份功能，点击进入之后点击确认即可直接备份。脱机备份会把数据库所有信息都保存，脱机后该数据库不可再使用，只有联机后才可重新使用。

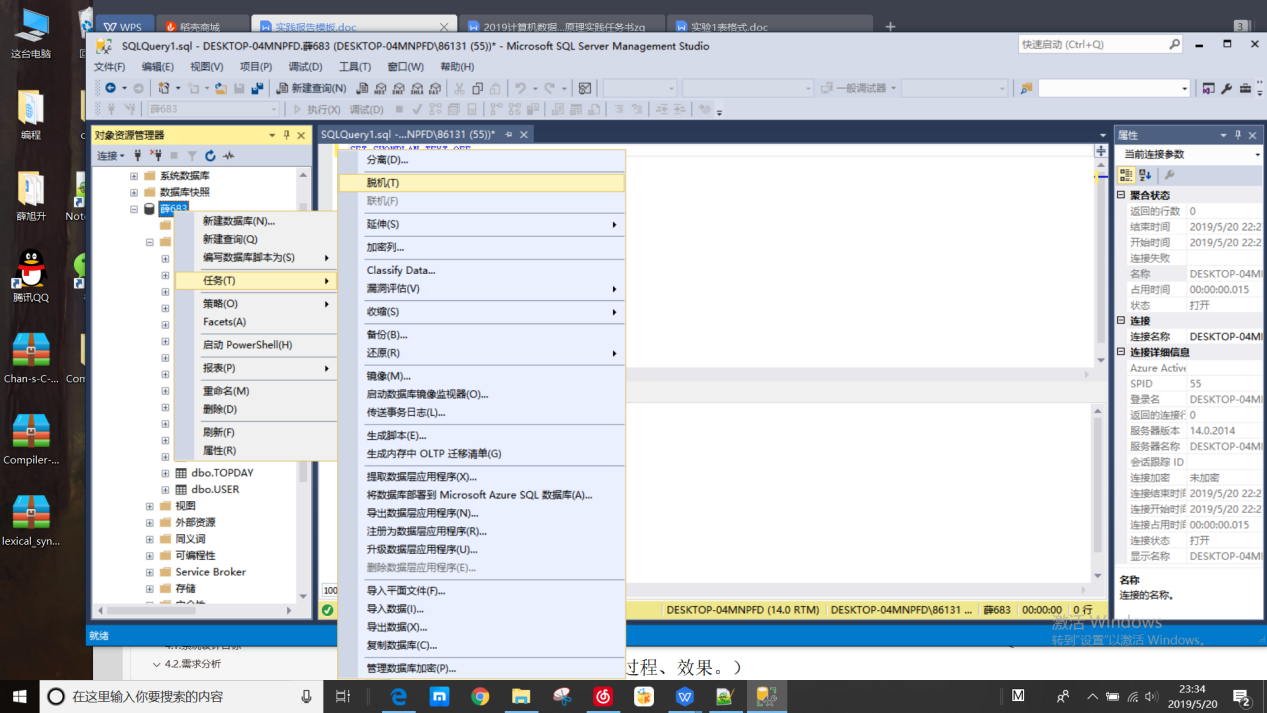


图 3.1 脱机备份

2）如图 3.2所示，鼠标选中的功能为数据的备份功能，选中后根据导出向导即可导出数据，导出数据有多种格式可以选择，备份功能相当与将数据复制，所以导出后数据库任可使用，并且可用导入数据功能重新载入数据。备份功能只会导出数据，各种配置不会备份，所以导入数据是必须保证各种配置相同，若角色、数据库名等不符就会导致导入失败。

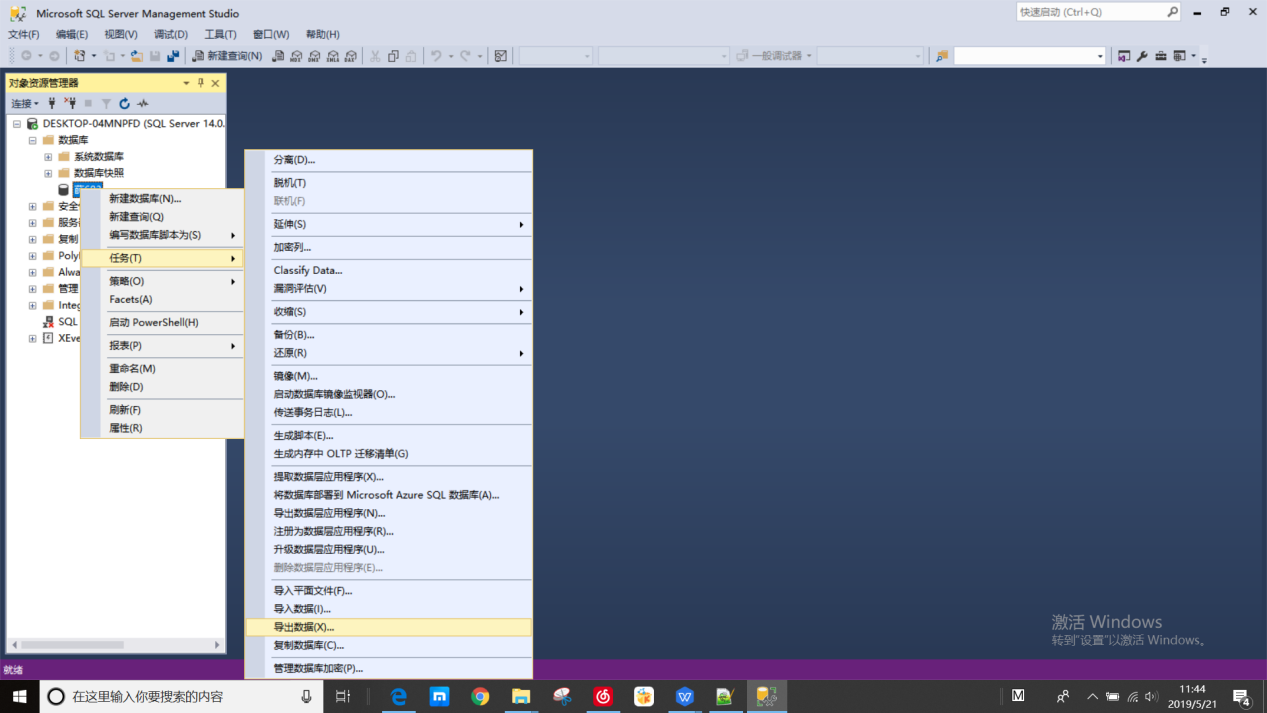


图 3.2 导出数据

### 用户授权与验证

1）首先根据教材新建三个表：学生表Student、课程表Course、选课表SC，并根据教材内容插入数据，插入数据可以直接使用图形界面完成。

2）由于题目要求创建3类用户，而非3个用户，所以需要创建3个角色：administrator、teacher、student，新建用户也可以直接使用图形界面完成，图 3.3为新建用户student截图。

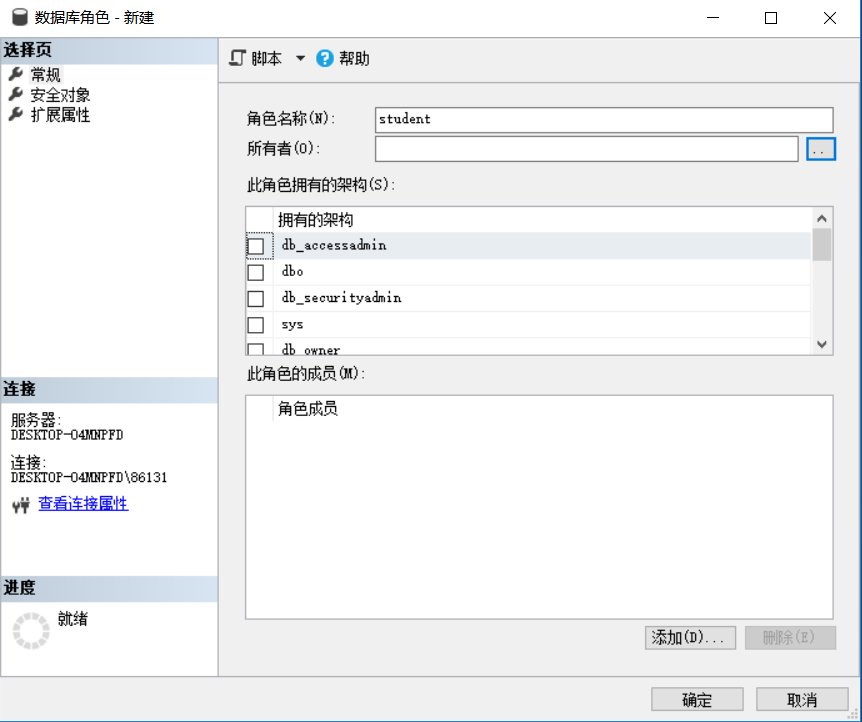


图 3.3 新建角色

3）为了测试授权效果，需要新建三个登录名，并新建三个用户：administrator1、teacher1、administrator、teacher、studentstudent1，用户可以绑定登录名，需要注意的是用户名不能和登录名重名，否则会出错。图 3.4为新建用户student截图，图 3.5为新建登录名student截图。

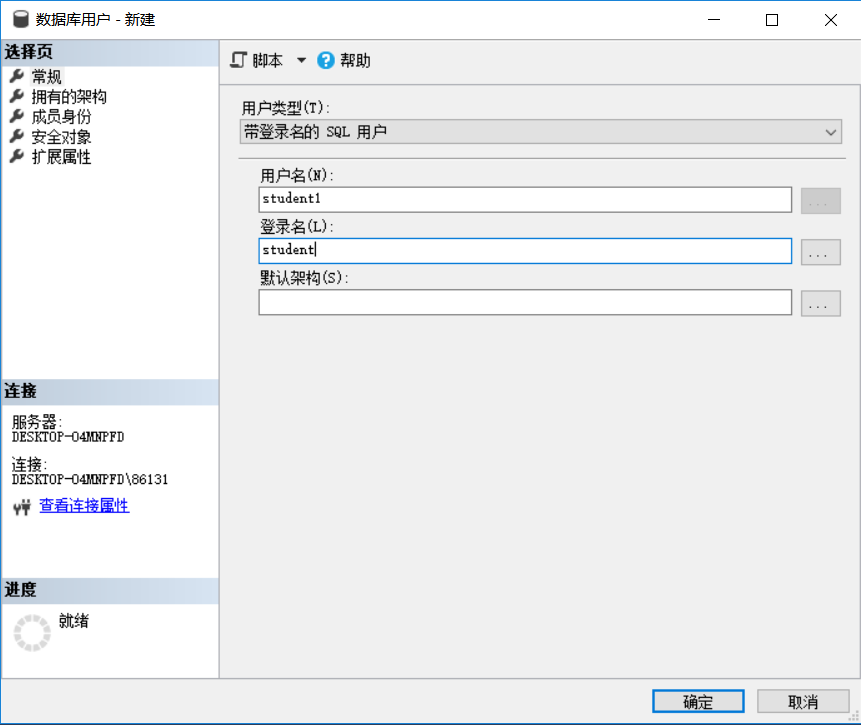


图 3.4 新建用户

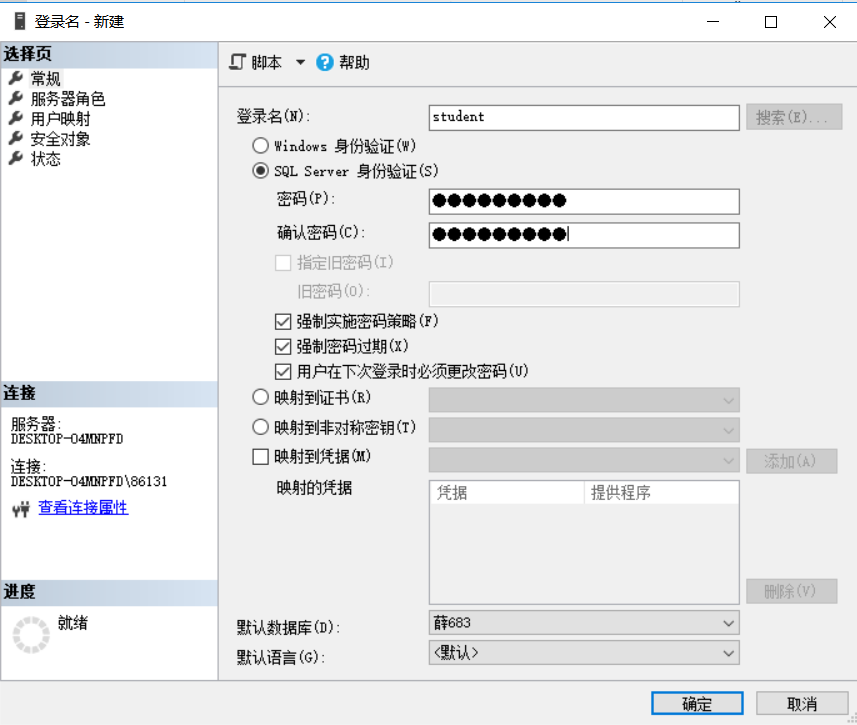


图 3.5 新建登录名

4）接着需要对三类角色授权。

管理员拥有所有权限并能将权限授予他人，其授权语句如下：

GRANT ALL PRIVILEGES ON Student TO administrator WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON Course TO administrator WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON SC TO administrator WITH GRANT OPTION;

学生可以查看课程表和选课表，可以进行选课操作，其授权语句如下：

GRANT SELECT ON Course TO student;

GRANT SELECT,INSERT ON SC TO student;

教师可以查看学生表、课程表和选课表，可以修改选课表，其授权语句如下：

GRANT SELECT ON Student TO teacher;

GRANT SELECT ON Course TO teacher;

GRANT SELECT,UPDATE(Grade) ON SC TO teacher;

5）将3个角色分别赋给3个用户。其SQL语句如下：

exec sp\_addrolemember 'student','student1';

exec sp\_addrolemember 'teacher','teacher1';

exec sp\_addrolemember 'administrator','administrator1';

6）开始测试。分别以用户名登录administrator、teacher、student，观察其数据库，如图 3.6~图 3.8所示。由于administrator、teacher对三个表都可以操作，故其表项有3个，而student只能操作课程表和选课表，所以只有两个表。

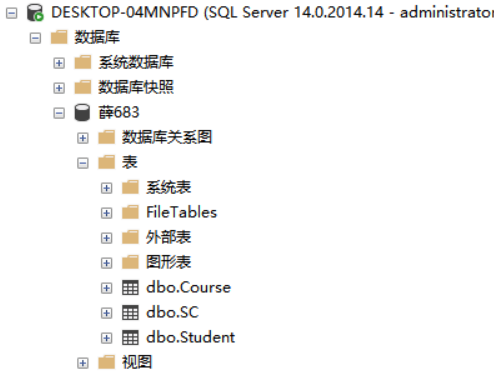


图 3.6 administrtor对应数据库表项

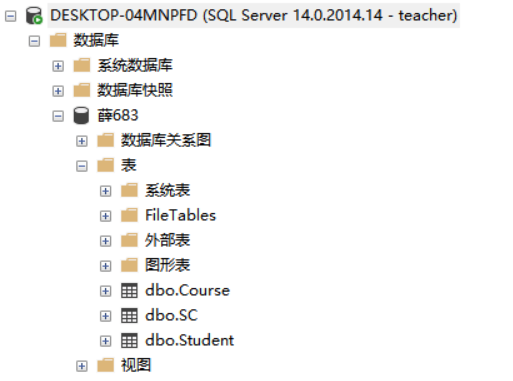


图 3.7 teacher对应数据库表项



图 3.8 student对应数据库表项

7）接着对三类用户功能进一步测试。其中管理员有所有权限，测试过程复杂，且建表时使用了管理员权限，所以再次不做测试。

对学生进行了三项测试：查询学生表、将数据插入学生表、将数据插入选课表。

查询学生表语句为：

select \* from [Student];

执行结果如图 3.9所示，学生无权限访问。

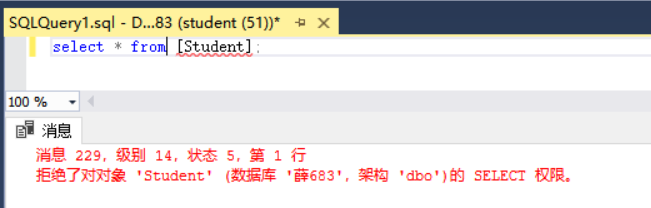


图 3.9 学生查询学生表

将数据插入学生表的语句为：

insert into [Student]

values ('201215121', '李勇', '男', 20, 'CS');

执行结果如图 3.10所示，学生无INSERT权限。

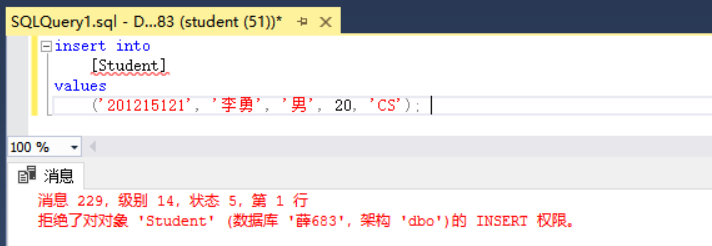


图 3.10 学生插入学生信息

将数据插入选课表的语句为：

insert into [SC]

values ('201215122', '1', null);

执行结果如图 3.11所示，插入成功，学生有选课权限。

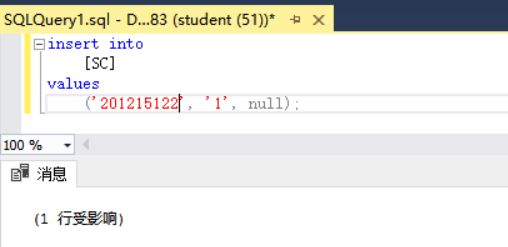


图 3.11 学生选课

对教师进行了四项测试：查询学生表、、将数据插入学生表、修改选课表上的学号、修改学号表上的成绩。

查询学生表的语句为：

select \* from [Student];

执行结果如图 3.12所示，教师对学生表有查询权限。

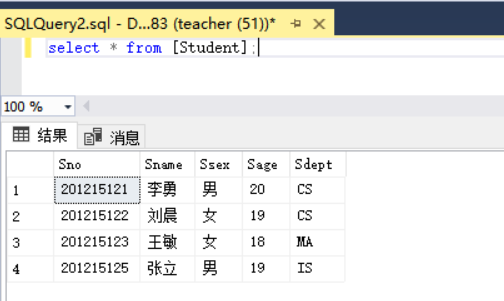


图 3.12 教师查询学生表

将数据插入学生表的语句为：

insert into [Student]

values ('201215121', '李勇', '男', 20, 'CS');

执行结果如图 3.13所示，教师对学生表无插入权限。

修改选课表上的课程号的语句为：

update [SC] set grade=90

where Sno='201215122' and Cno=1;

执行结果如图 3.14所示，教师对选课表的课程号无修改权限。

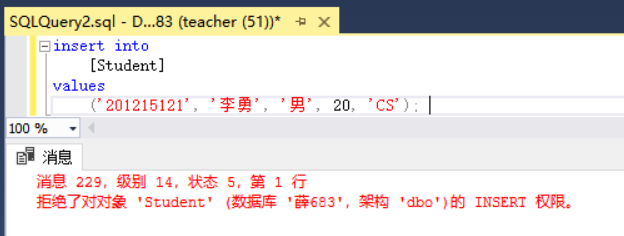


图 3.13 教师插入学生信息

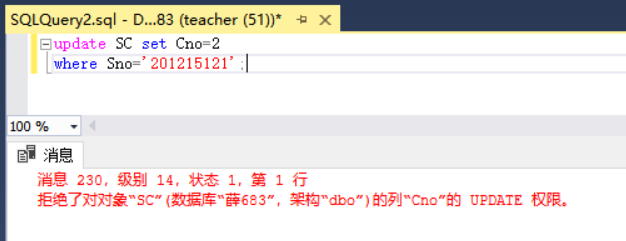


图 3.14 教师修改课程号

修改选课表上的成绩的语句为：

update SC set Cno=2

where Sno='201215121';

执行结果如图 3.15所示，教师可以修改选课表中的成绩。

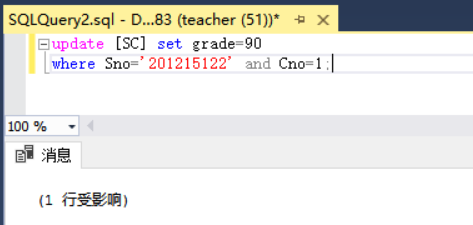


图 3.15 教师修改成绩

## 任务总结

1. 插入课程表时由于课程号是表的外码，所以插入先行课时会出错，最后解决方法是：先将先行可属性置为空，创建完课程表后再插入先行课。

2. 由于SQL Server的授权语句与教材中的SQL授权语句有差别，所以第二次实验时又出现了错误，不过由于之前写触发器时有了经验，所以马上想到了原因所在，去网上查找了SQL Server的授权语句改正了过来。

# 综合实践任务

## 系统设计目标

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

选择题目：仓储管理系统

## 需求分析

### 任务文档要求

实现一个仓库库存货品信息管理系统软件。仓库的日常工作包括货品的入库和出库。入库要由采购人员提供进货单，进货单经过审核人员审核验收后方能进行货品入库。出库要由销售人员提供出货单，经过审核人员审核批准后才能提货。当销售人员需要提货而货品的库存量不足时可先进行缺货登记，当有相应货品入库时，按缺货登记时间顺序处理出货请求。要求：

* 能实现库存货品信息的管理，货品的相关信息包括：货品号、货品名、存放地、货品库存量、生产厂家等。
* 实现进货单的填写、修改、审核和查询等功能。
* 实现出货单的填写、修改、审核和查询等功能。
* 实现缺货登记、查询和处理。
* 根据不同用户身份提供不同的操作权限和界面。

设计一个B/S或C/S模式的系统实现上述功能。

### 需求分析内容

概述：仓储管理系统是一个面向公司内部的管理系统，方便管理货物的进库、出库。

功能需求：

1. 作为采购人员，能够通过管理系统填写进货单并能查询、修改自己填写的进货单。

2. 作为销售人员，能够通过管理系统填写出货单并能查询、修改自己填写的出货单。

3. 作为审核人员，能够修改、审核、查询所有进货单和出货单，并能进行缺货登记和缺货处理，还可以查询所有货物信息。

5. 不同级别的管理系统账户有不同的修改和浏览权限（在GUI界面上实现阻止访问）。

6. 使用者无需在SQL级别上进行操作，均可通过管理系统的GUI界面满足需求。

展开需求分析，给出需求分析的主要内容的阐述，包括具体的功能需求、性能需求、数据完整性需求、数据流图、数据字典等。

### 数据字典

1）数据项

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项名 | 含义 | 数据类型 | 长度 | 取值范围 |
| 以下是用户表[User]的数据项 | | | | |
| 用户账号 | 用户登录号，唯一标识用户 | char | 6 |  |
| 用户密码 | 验证用户身份 | char | 6 |  |
| 用户名 | 用户的真实姓名 | char | 6 |  |
| 用户权限 | 用户身份标识 | char | 1 | 0~2（0：采购人员，1：销售人员，2：审核人员） |
| 以下是厂家表[Purchase]的数据项 | | | | |
| 厂家号 | 厂家的唯一标识 | int | 4 |  |
| 厂家名 | 厂家的真实名称 | char | 10 |  |
| 以下是进货单表[Purchase]的数据项 | | | | |
| 进货编号 | 进货单的唯一标识 | int | 4 |  |
| 进货数量 | 采购的货品数量 | int | 4 |  |
| 进货价 | 货品的采购总价 | dec(10,2) |  |  |
| 进货日期 | 提交进货单的日期 | timestamp |  |  |
| 进货状态 | 进货单当前状态 | int | 4 | 0~2（0：审核中，1：已通过，2：已驳回） |
| 进货备注 | 进货单的附加信息 | varchar | 100 |  |
| 以下是出货单表[Sale]的数据项 | | | | |
| 出货编号 | 出货单的唯一标识 | int | 4 |  |
| 出货数量 | 销售的商品数量 | int | 4 |  |
| 出货价 | 货品的销售总价 | dec(10,2) |  |  |
| 出货日期 | 提交出货单的日期 | timestamp |  |  |
| 出货状态 | 出货单当前状态 | int | 4 | 0~3（0：审核中，1：已通过，2：已驳回，3：缺货中） |
| 出货备注 | 出货单的附加信息 | varchar | 100 |  |
| 以下是缺货表[Stockout]的数据项 | | | | |
| 缺货状态 | 货品能否处理标志 | int | 4 | 0~1（0：缺货中，1：可处理） |
| 以下是货品表[Goods]的数据项 | | | | |
| 货品号 | 货品的唯一标识 | int | 4 |  |
| 货品名 | 货品的真实名称 | char | 10 |  |
| 以下是仓库表[warehouse]的数据项 | | | | |
| 仓库号 | 仓库的唯一标识 | int | 4 |  |
| 仓库名 | 仓库的真实名称 | char | 10 |  |
| 仓库备注 | 用于描述仓库额外信息 | varchar | 100 |  |
| 以下是库存表[Storage]的数据项 | | | | |
| 库存数量 | 某货物存储在某仓库的数量，依赖于货品号和仓库号 | int | 4 |  |

2）数据结构

{

数据结构名：用户表

含义说明：保存用户的基本信息。

组成：用户账号、用户密码、用户名、用户权限。

}

{

数据结构名：厂家表

含义说明：保存各厂家的基本信息。

组成：厂家号、厂家名

}

{

数据结构名：货品表

含义说明：保存所有货品的基本信息。

组成：货品号、货品名、厂家号

}

{

数据结构名：仓库表

含义说明：保存所有仓库的基本信息

组成：仓库号、仓库名、仓库备注

}

{

数据结构名：库存表

含义说明：保存各个仓库中货品数量信息。

组成：仓库号、货品号、库存数量

}

{

数据结构名：进货单表

含义说明：保存所有进货信息。

组成：进货编号、货品号、进货数量、用户号、进货价格、仓库号、进货日期、进货备注、进货状态。

}

{

数据结构名：出货单表

含义说明：保存所有出货信息。

组成：出货编号、货品名、出货数量、用户号、出货价格、厂家号、出货日期、出货备注、出货状态。

}

{

数据结构名：缺货表

含义说明：保存当前缺货信息。

组成：出货编号、货品名、出货数量、缺货状态。

}

3）数据流

{

数据流名：进货单

说明：记录进货信息

数据流来源：采购人员

数据流去向：提交

}

{

数据流名：出货单

说明：记录出货信息

数据流来源：销售人员

数据流去向：提交

}

{

数据流名：缺货单

说明：记录缺货信息

数据流来源：审核

数据流去向：缺货处理

}

4）数据流图

业务处理主要分为两部分：进货与出货。进货数据流图如图 4.1所示，采购人员填写产生进货单并提交，若输入不合格数据流返回给采购人员，否则存入数据库中的进货单表，经审核后进货单重新返回采购人员，并在此过程中产生库存信息。出货数据流图如图图 4.2所示，销售人员填写产生出货单并提交，若输入不合格数据流返回给销售人员，否则存入数据库中的出货单表，经审核后出货单和出货信息一起返回给销售人员，若缺货则产生缺货表，并经过缺货处理后再返回给销售人员，是否缺货需要借助库存表进行判断。

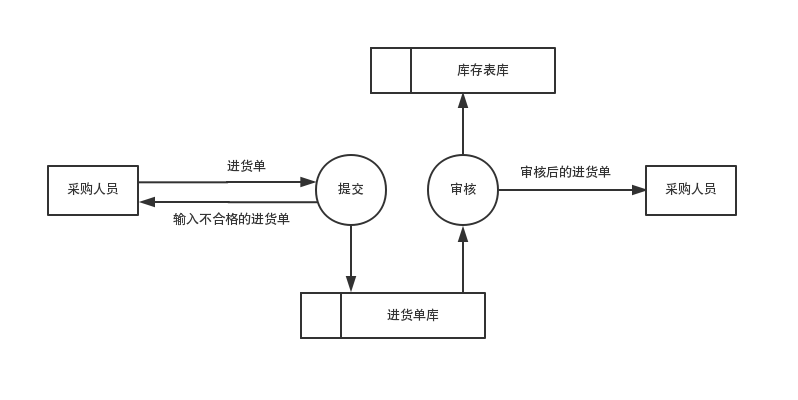


图 4.1 进货数据流图

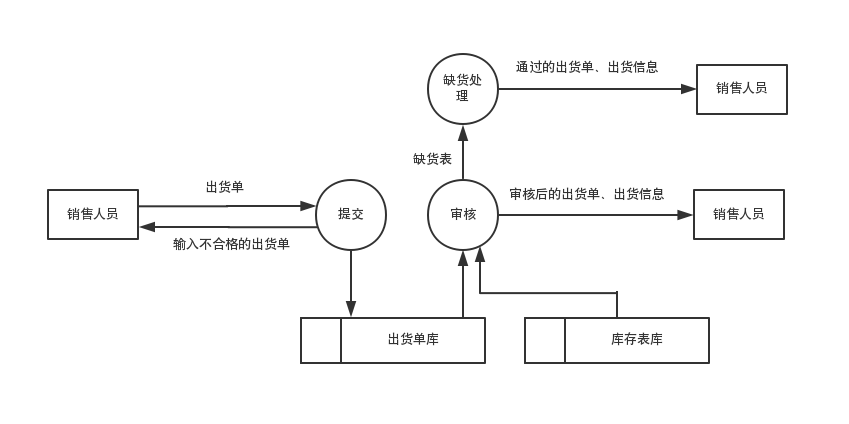


图 4.2 出货数据流图

## 总体设计

### 系统体系结构

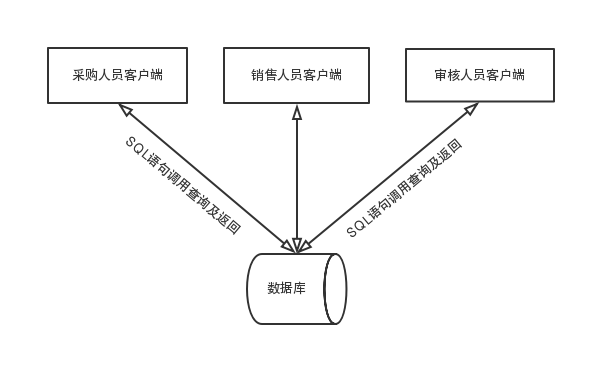


图 4.3 系统体系结构图

图 4.1为仓储管理系统的系统体系结构图。该系统中用户根据权限分为3种：采购人员、销售人员、审核人员，所以客户端也分为3种，各个客户端通过数据库进行数据传输，客户端访问数据库时通过SQL语句对其进行操作。

### 功能模块

如图 4.2所示，根据用户类型，仓储管理系统可以分为4个模块：登录模块、采购人员模块、销售人员模块、审核人员模块。后三种模块又可根据其权限进行模块细分。四个模块具体功能如下：

* 登录模块：用户进入后先从登录模块进行登录，该模块主要用于验证用户身份，并根据登录用户身份不同加载不同操作界面，用户权限的限制也是通过该模块实现的。
* 采购人员模块：采购人员需要填写进货单，进货单内容包括：货品名、货品数量、生产厂家、进货价。采购人员可以查询自己提交的所有进货单，查询后可以修改还未审核的进货单。
* 销售人员模块：销售人员需要填写出货单，出货单内容包括：货品名、货品数量、生产厂家、销售厂家、进货价。销售人员可以查询自己提交的所有出货单，查询后可以修改还未审核的出货单。
* 审核人员模块：审核人员可以管理货单：修改/审核货单，并对缺货的货单进行登记。审核人员模块还提供查询所有货品功能，货品信息包括：货品号、货品名、存放地、货品库存量、生产厂家，货品信息根据货品名、存放地、生产厂家分组。

## 数据库设计

* 图 4.3展示了主要管理数据的E-R图，数据库中的主要数据主要由7个实体构成：用户、厂家、进货单、出货单、缺货表、货品、仓库。用户分为：采

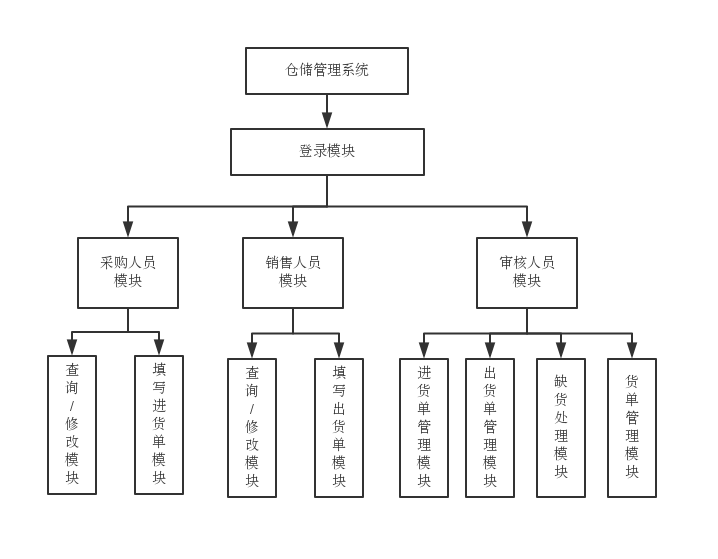


图 4.4 功能模块划分

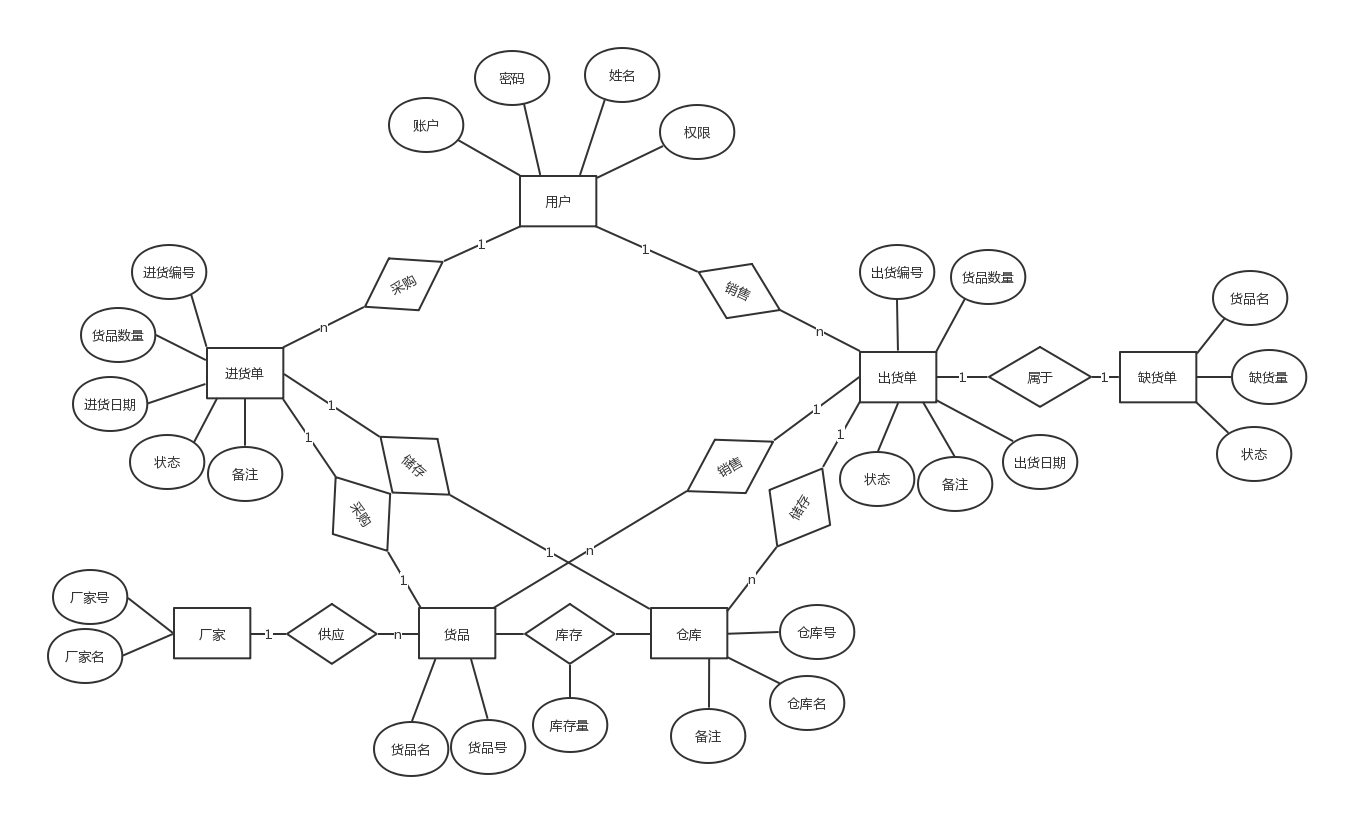


图 4.5数据库E-R图

购人员、销售人员、审核人员，采购人员从厂家采购货品从而生成进货单，销售人员向厂家销售货品从而生成出货单，都是一对多的关系。同一种货品厂家不同则货品号不同，所以一种货品对应一个厂家，一个厂家可以供应多种货品，所以是一对多的关系。一种货品可以存到多个仓库，一个仓库可以存多种货品，所以货品和仓库是多对多的关系。每个进货单只记录一种货品，该货品只存储到某一个仓库，所以都是一对一的关系。而出货单记录货品名相同的一类货品，该货品可能来自多个仓库，所以是一对多的关系。而缺货表属于出货单的一部分，用于缺货登记，只记录缺货信息，当缺货处理后，对应的缺货信息会删除，每个缺货表元组对应出货单一个元组，是一对一的关系。

## 详细设计与实现

### 数据库设计

1）视图

本次任务中查询需求较多，包括：进货单、出货单、货品列表，且这三个表都存在多个外码，查询语句复杂，所以可以定义为视图，每次查询时直接从视图中获取相应数据即可。

进货单的视图创建语句如下：

|  |
| --- |
| DROP VIEW IF EXISTS `PURCHASE\_SHOW`;  CREATE VIEW `PURCHASE\_SHOW` AS  SELECT number, gname, mname, gnumber, wname, time, price, status, `purchase`.uid 'uid', name  FROM `purchase`, `goods`, `manufacturer`, `warehouse`, `user`  WHERE `purchase`.gid=`goods`.gid and `goods`.mid=`manufacturer`.mid  and `purchase`.wid=`warehouse`.wid and `purchase`.uid=`user`.uid; |

该视图包含10个属性，分别是进货编号、货品名、进货厂家名、进货数量、仓库名、进货时间、进货价、进货单状态、进货人id、进货人姓名。其中进货人id、进货时间主要用于查询，而其他属性一般用于显示进货单基本信息。

出货单的视图创建语句如下：

|  |
| --- |
| DROP VIEW IF EXISTS `SALE\_SHOW`;  CREATE VIEW `SALE\_SHOW` AS  SELECT number, gname, mname, gnumber, price, time, status, `sale`.uid 'uid', name  FROM `sale`, `manufacturer`, `user`  WHERE `sale`.mid=`manufacturer`.mid and `sale`.uid=`user`.uid; |

该视图包含9个属性，分别是出货编号、货品名、出货厂家、出货数量、出货价、出货时间、出货单状态、出货人id、出货人姓名，与进货单类似，出货单中的出货人id和出货时间也主要用于查询，其他属性主要用于显示进货单基本信息。

货品信息表的视图创建语句如下：

|  |
| --- |
| DROP VIEW IF EXISTS `GOODS\_SHOW`;  CREATE VIEW `GOODS\_SHOW` AS  SELECT `goods`.gid 'gid', gname, mName, wName, number  FROM `storage`,`goods`,`warehouse`,`manufacturer`  WHERE `storage`.gid=`goods`.gid and `storage`.wid=`warehouse`.wid  and `goods`.mid=`manufacturer`.mid; |

该视图只有5个属性，分别是货品id，货品名、生产厂家、存储仓库名、库存量。

2）触发器

本次任务的数据库中共设置了两个触发器：当缺货表插入元组时，将出货表上与其对应的元组的状态更改为“缺货中”；当缺货表删除元组时，及那个出货表上与其对应的元组的状态更改为“已通过”。

通过创建这两个触发器，可以减少外部对缺货处理的步骤流程，使缺货处理只需操作缺货表，无需再考虑出货表，大大简化了外部操作。触发器的创建语句如下：

|  |
| --- |
| CREATE TRIGGER `STOCKOUT\_CHECK1`  BEFORE INSERT  ON `STOCKOUT` FOR EACH ROW  UPDATE `SALE` SET STATUS=3 WHERE NUMBER=NEW.NUMBER;  CREATE TRIGGER `STOCKOUT\_CHECK2`  AFTER DELETE  ON `STOCKOUT` FOR EACH ROW  UPDATE `SALE` SET STATUS=1 WHERE NUMBER=OLD.NUMBER; |

3）自定义函数

本次任务的数据库中之定义了一个自定义函数：isEnough，用于查看仓库中货品是否足够，在出货表审核及缺货处理时调用。该自定义函数需要传入两个参数：货品名和货品需求量，返回值是个布尔值。如果库存中货品数量达到需求量，则返回true，否则返回false。函数isEnough的创建语句如下：

|  |
| --- |
| /\* 自定义函数: isEnough  参数列表：gName：货品名，gNumber：货品需求量  返回值：仓库中对应货品是否足够  \*/  drop function if exists isEnough;  create function isEnough (name varchar(10), gNumber int)  returns boolean  begin  declare n int default 0; /\* 保存仓库中当前货品数量 \*/  select number from `storage`,`goods`  where `storage`.gid=`goods`.gid and gname=name  group by gname into n;  if n>=gNumber then return true; /\*仓库中货品数量大于需求量，返回true \*/  else return false;  end if;  end ; |

### 登录模块

登录模块整体流程较为简单，如图 4.4所示，输入账号、密码后，用户点击登录按钮即开始登录流程，程序先进行非空检查，若账号或密码为空，则会输出错误信息，否则程序会访问数据库，通过账号和密码获取权限，若获取失败，则说明账号或密码错误，无法登录，若获取成功，则根据权限进入不同界面，如果权限为0，进入审核模块界面，如果权限是1，进入采购模块界面，如果权限是

2，进入销售模块界面。

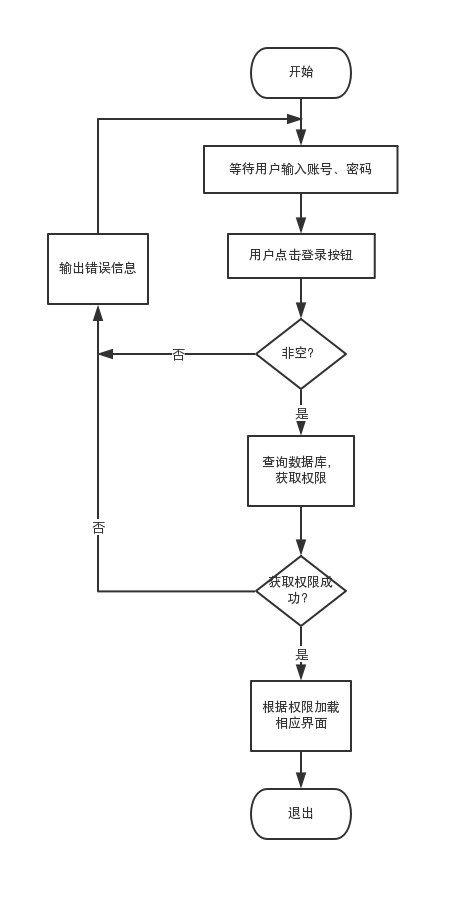


图 4.6 登录模块流程图

### 采购人员模块

采购人员模块主要分为两部分：填写进货单模块和查询模块。填写进货单模块只需要提交进货单信息，将进货单信息存入数据库即可，但是需要对用户的所有输入进行检错，保证存入正确信息。查询模块使用tableview即可展示，但需要根据需求制作单元格工厂。

填写进货单模块的整体流程如图 4.4所示。用户需要先填写进货单信息，包括：货品名、进货厂家、进货数量、进货价格、货品存储的仓库名，备注可选择性输入，用户点击提交按钮后开始进货单生成流程，先对以上几项进行非空检查，保证需求数据都已填写；然后通过货品名和生产厂家锁定货品号，若不存在说明两者不匹配，进行报错，仓库号与之类似；最后查询数据库获取进货编号及当前时间，若数据库正常，则可获取成功；最后将所有数据存入数据库即可。

查询模块无流程，点击按钮后之后查询即可。进货单的显示可以选用tableview实现，而使用tableview需要先建立一个类保存进货单的信息，该类中每个属性需要一个get方法和set方法，创建类后还需要将属性与tableview中的列绑定，以进货编号为例，其绑定语句为：pNumberInTable.setCellValueFactory(new PropertyValueFactory<>("pNumber"));

其中pNumberInTable是tableview中进货编号列名，pNumber是类中保存进货编号的变量名。

除此以外，进货查询还需要实现修改功能，及对于还未审核的进货单可以进行修改，凡是进货单填写的属性都应可以修改，包括：货品名、进货厂家、进货数量、进货价格、货品存储的仓库名，进货单修改可以新建一个界面，在新的界面中填写新的数据并提交。为了针对每个进货单分别修改，需要在tableview的每一行表项后加一个修改按钮，用于进入修改界面。为了连接两个界面的信息，可以设置一个全局变量保存进货编号，当点击修改按钮后，若对应进货单状态为“审核中”，将对应行的进货编号送入全局变量中保存，进入修改界面后可以通过该全局变量从数据库中获取进货单信息。而修改按钮的设置可以通过继承TableCell后自定义单元工厂实现，只需要在子类中重写uodateItem方法即可，在该方法中加入按钮，接着在子类的构造方法中为按钮添加事件驱动，当按钮接收到鼠标点击事件后进行响应。

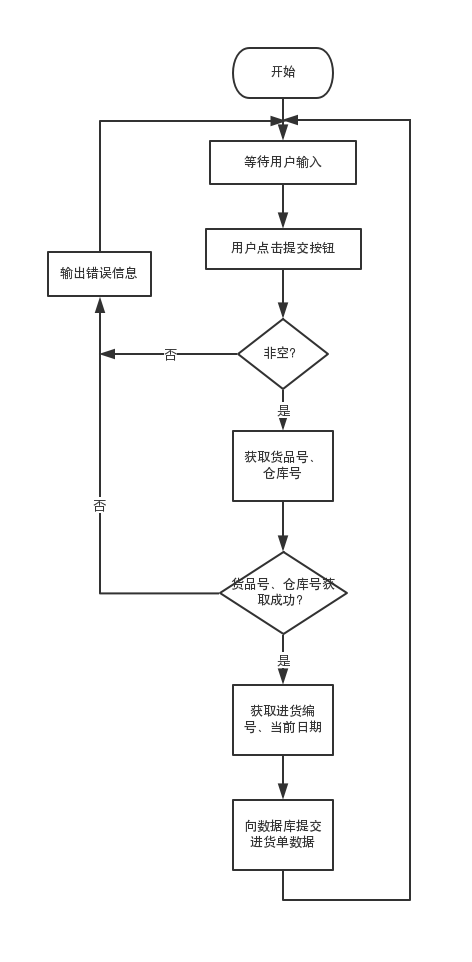


图 4.7 填写进货单流程图

而出货人员模块实现与进货人员设计相同，所以不予介绍。

### 审核人员模块

审核人员模块分为4部分：进货处理、出货处理、缺货处理、货品查询。这4个界面都涉及到查询，类似采购人员界面，这些查询结果都使用tableview展示，需要用到按钮的地方也通过继承TableCell实现，其中进货处理和出货处理中对于每个货单都有三种处理方式，所以通过ComboBox选择操作，而缺货处理中只有一种操作，所以可以设置为按钮。

进货处理界面是一个查询界面，可根据时间和货单状态查询所有进货单，查询后可对进货单进行操作，主要是对还未审核的进货单的操作，包括：通过进货单、驳回进货单、修改进货单，这三个操作通过ComboBox进行选择。驳回进货单较为简单，点击后直接通过数据库修改进货单状态即可；修改进货单可以借用采购人员模块中设计的修改界面实现，点击后直接进入修改界面即可；通过较为复杂，通过后先要将进货单状态更改为2（已通过），接着根据进货单信息更新库存表，最后还要对缺货表进行一次全表检查，根据当前进货单中货品名将缺货表中所有对应货品检查一遍，若货品足够，则将缺货表中对应元组状态更新为“可处理”，进货单通过的流程图如图 4.6所示，需要注意的是更改库存表时有可能仓库中已有该货品，此时无法插入新的信息，只能增加库存数量，所以更改库存时需要先查询数据库是否该库存是否已存在。除此以外，该过程中需要修改多个表，所以需要对整个事务进行失误处理，用于中间某一步出错时数据库恢复，具体步骤为：在所有事务前先使用setAutoCommit(false)阻止事务自动提交，在所有事务之后使用commit()一起提交，如果出现异常使用rollback()即可回滚恢复。

出货界面与进货界面相同，只是审核通过时有可能出现缺货的情况，所以审核通过流程不同，其具体流程为：点击通过按钮后，先检查库存中该货品是否足够，若不足则询问是否缺货登记，如果登记则向缺货表中插入缺货表信息，否则直接结束；如果货品充足直接进行出货流程，先将出货单状态更改为2（已通过），接着根据库存顺序开始出货，若当前仓库库存量不足，则全部出货，并循环至下一仓库，直至仓库库存足够，出货过程中需要将出货信息保存至出货表备注中用于显示。出货审核通过流程图如图 4.7所示。需要注意的是4.5.1中已提过缺货表插入时会触发触发器，所以缺货登记时无需在外部更改出货表，除此以外该部分也需要进行事务处理。

由于进货处理部分和出货部分处理已将出货处理的大部分事务已处理，所以出货处理较简单，出货处理部分主功能为根据查询条件将缺货表显示，处理缺货只需要将缺货表中对应元组删除，缺货表的插入由出货处理部分完成，缺货表的状态更新由进货处理部分完成，处理缺货时出货表的更新由触发器自动完成。

货品查询部分最为简单，只需要根据库存表进行查询并显示，无需进行操作。

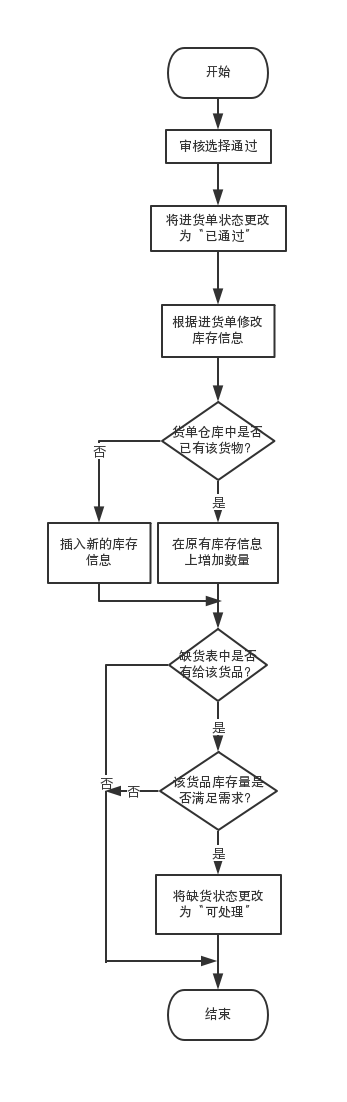


图 4.8 进货单通过流程图

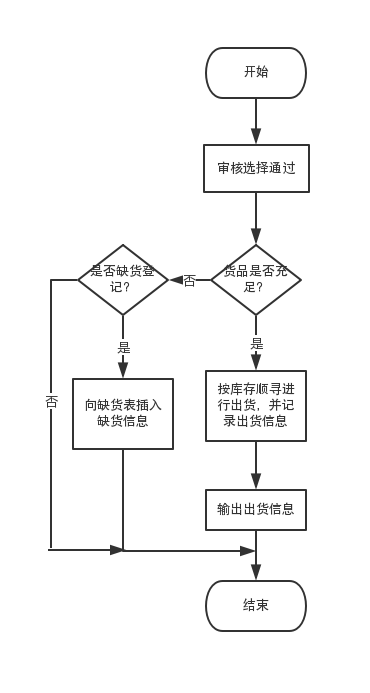


图 4.9 出货单通过流程图

## 系统测试

本次测试刚开始数据库中只有用户信息、货品信息、仓库信息、厂家信息，进货单、出货单、库存信息、缺货信息均从测试中产生。

### 登录模块测试

1.程序运行后即进入登录界面，如图 4.10所示，登录界面由一个文本框和一个密码框组成，确定按钮表示登录，清除按钮表示清除当前输入的信息。



图 4.10 登录界面

2.先进行非空测试，输入任意一个编号，并不输入密码，如图 4.11所示，输出错误提示信息“请输入密码”，符合功能要求。



图 4.11 登录界面非空检测

3.再进行错误测试，由于密码框不可看见，所以只测试账号，数据库用户账号只有000001、000002、000003三个，所以输入000004测试，如图 4.12所示，输出错误提示信息“编号或密码错误！”，符合功能要求。

### 采购人员模块测试

1.采购人员登录后直接进入进货单填写界面，如图 4.13所示，界面主要由9个文本框组成，其中进货员工编号、进货日期、进货编号是不可编辑的，进货员工编号在登录后自动生成，进货编号、进货日期在提交后生成。

2.测试非空检查，以商品名称为例，如图 4.14所示，货品名称为空时点击提交，错误提示“请输入货品名”，非空检查功能正常。



图 4.12 登录界面账号错误测试

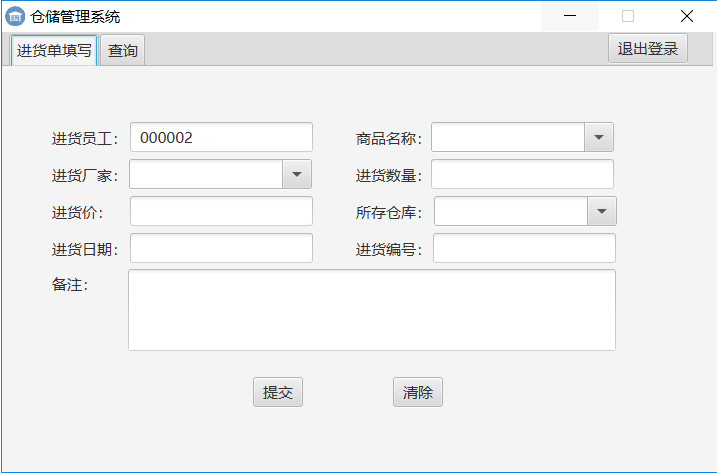


图 4.13 进货单填写界面



图 4.14 进货单填写非空检查

3.进行货品号错误测试，根据数据库，广东蔬菜公司不提供苹果，所以商品名称选择苹果，点击提交，结果如图 4.15，输出错误提示，货品号检测的功能正常。

4.输入正确信息，进行提交，结果如图 4.16所示，提交成功并返回进货编号1。为了后面的测试，还需要提交两个进货单：数量100的香蕉和葡萄。

5.测试查询模块，如图 4.17所示，查询条件只有时间，点击查询后，输出规定时间内的所有历史进货单，图中进货单正是之前提交的3个进货单，查询功能正常，并说明之前进货单提交成功。



图 4.15 进货单填写货品错误测试

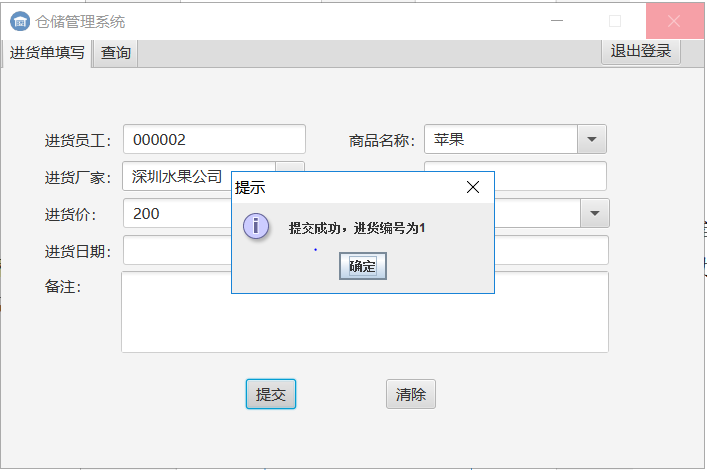


图 4.16 进货单填写成功提交

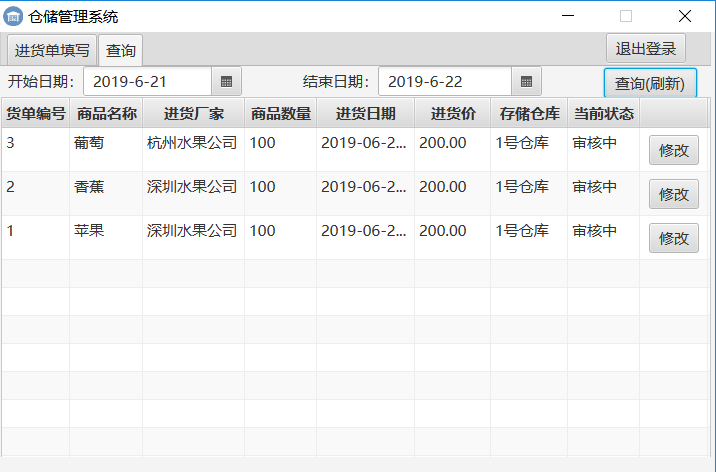


图 4.17 进货单查询

6.测试修改功能，点击编号3的进货单的修改按钮，进入修改界面，如图 4.18所示，将进货价改为300，并点击修改按钮，结果显示修改成功。返回查询界面，再次查询，结果如图 4.19，3号进货单中的进货价变为300，修改成功。

### 销售人员模块测试

1.从采购人员界面退出后登录进入销售人员界面，如图 4.20所示，销售界面由8个文本框组成，与进货界面相同，出货员工编号、出货日期、出货编号都是不可编辑的。

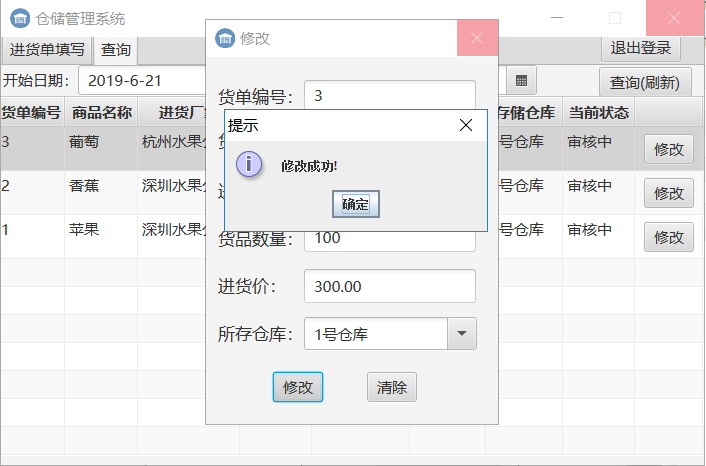


图 4.18 进货单修改



图 4.19 进货单修改结果

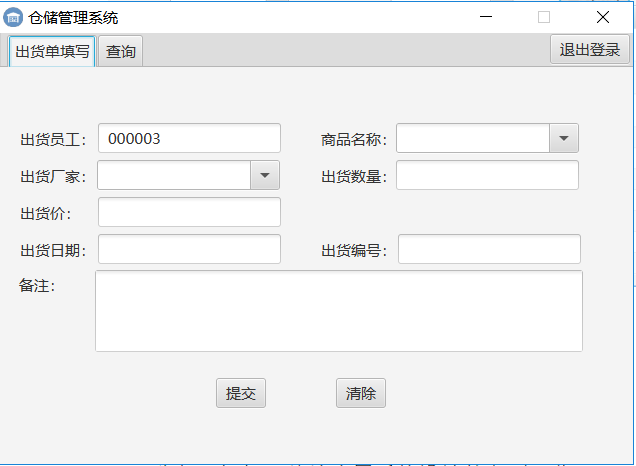


图 4.20 销售界面

2.测试非空检测，以商品名称为例，结果如图 4.21所示，弹出错误提示框，非空检测功能正常。

3..输入正确信息，进行提交，结果如图 4.22所示，提交成功并返回出货编号1。为了后面的测试，还需要提交两个出货单：数量100的香蕉和葡萄。



图 4.21 出货单非空检测

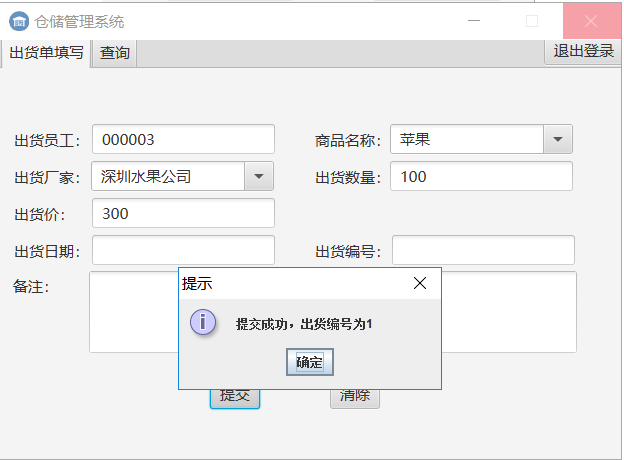


图 4.22 出货单成功提交

4.测试查询模块，如图 4.23所示，查询条件只有时间，点击查询后，输出规定时间内的所有历史进货单，图中出货单正是之前提交的3个进货单，查询功能正常，并说明之前出货单提交成功。



图 4.23 出货单查询

5.测试修改功能，点击编号3的出货单的修改按钮，进入修改界面，如图 4.24所示，将出货数量修改为50，并点击修改按钮，结果显示修改成功。返回查询界面，再次查询，结果如图 4.25，3号出货单中的出货数量变为50，修改成功。

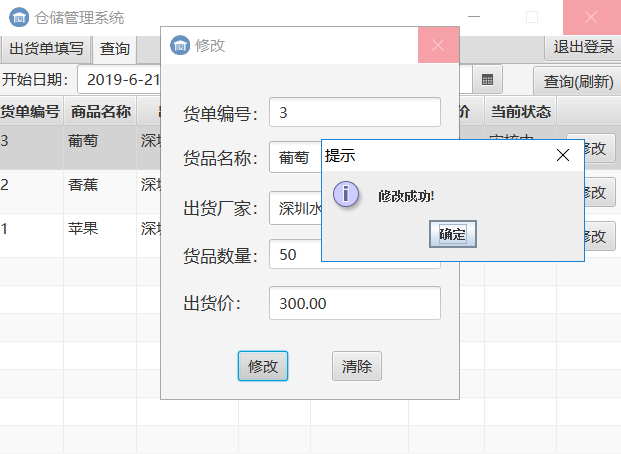


图 4.24 出货单修改



图 4.25 出货单修改结果

### 审核人员模块测试

1.退出销售界面进入蛇和审核界面，审核界面主要由进货单查询、出货单查询、缺货处理、货品查询4部分组成，与审核人员模块的4个功能模块对应。



图 4.26 审核界面

2.为测试缺货处理，先进行出货，当前还未进货，仓库中应无货品，进入“货品查询”界面查询仓库货品，如图 4.27所示，不输入条件，表示查询所有货品，结果为空。。



图 4.27 货品查询1

3.进行出货，当前审核中的出货单如图 4.28所示，只有之前提交的3个，由于仓库没有任何货物，所以出货不会成功，如图 4.29所示，选择通过后弹出提示框询问是否进行缺货登记，选择确认，进行登记，进入缺货查询界面查询，结果如图 4.30所示，缺货信息已登记，当前状态为已登记。

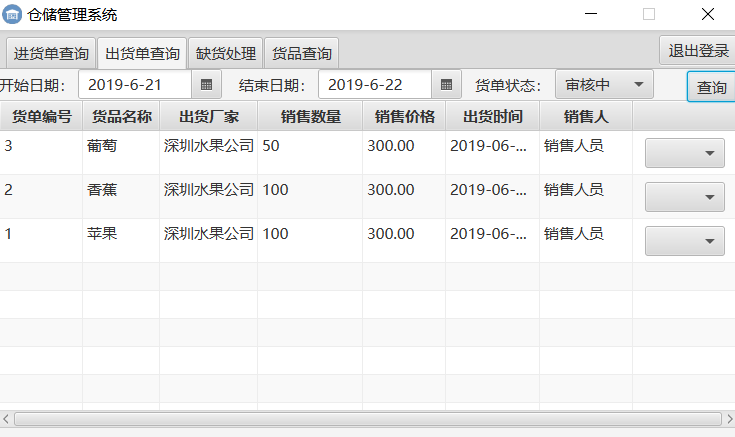


图 4.28 出货单状态

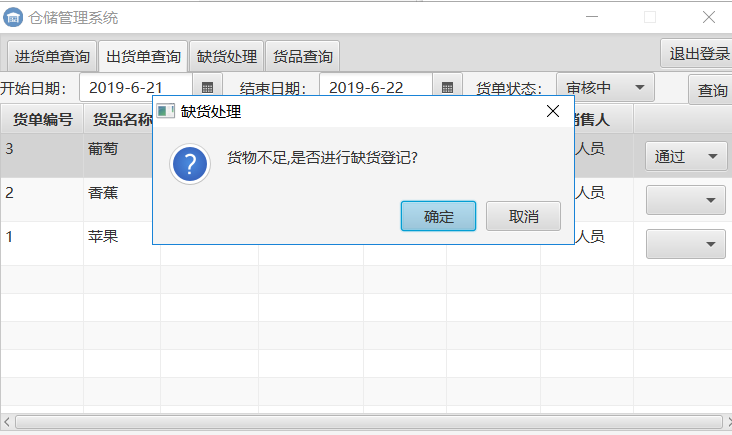


图 4.29 缺货登记询问

4.进行进货，当前审核中的进货单如图 4.31所示，只有之前提交的3个，将3号进货单审核通过，结果如图 4.32所示，提示框显示进货成功，再进入货品查询查看，如图 4.33所示，货品数据已存入数据库，再进入缺货处理界面查看，如图 4.34所示，缺货状态已转变”，说明进货处理正常。

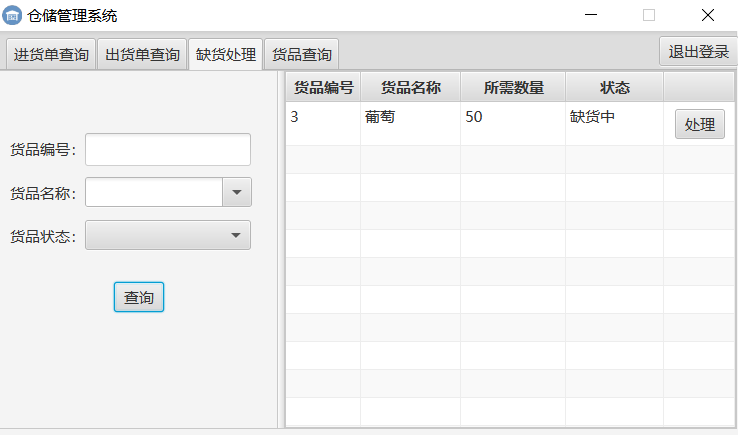


图 4.30 缺货查询1



图 4.31 进货单状态

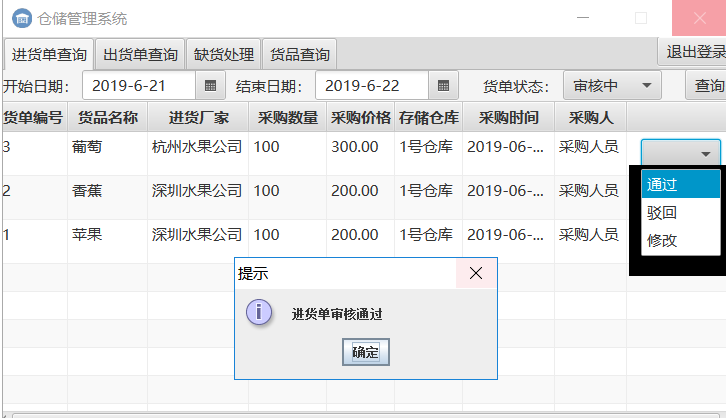


图 4.32 进货单通过

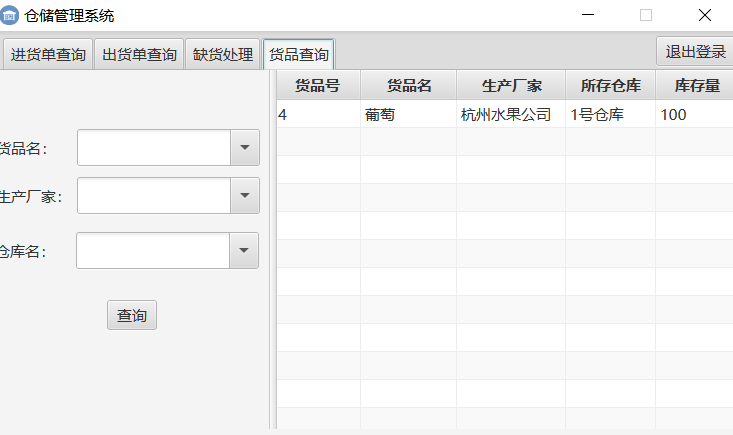


图 4.33 货品查询2

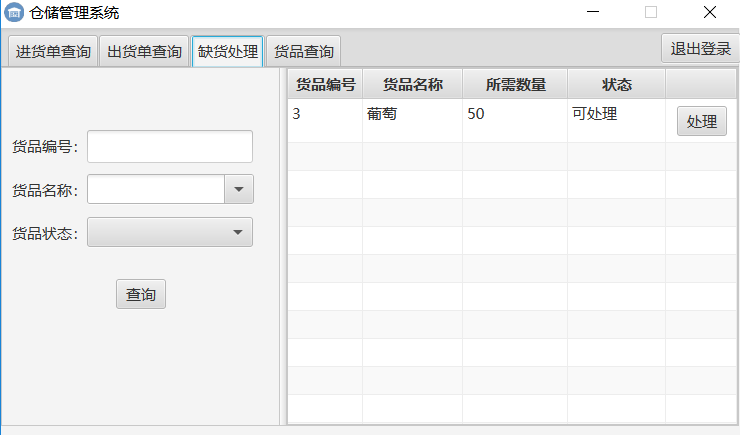


图 4.34 缺货查询2

5.开始测试缺货处理，点击处理按钮，结果如图 4.35所示，出货成功并输出了出货信息。

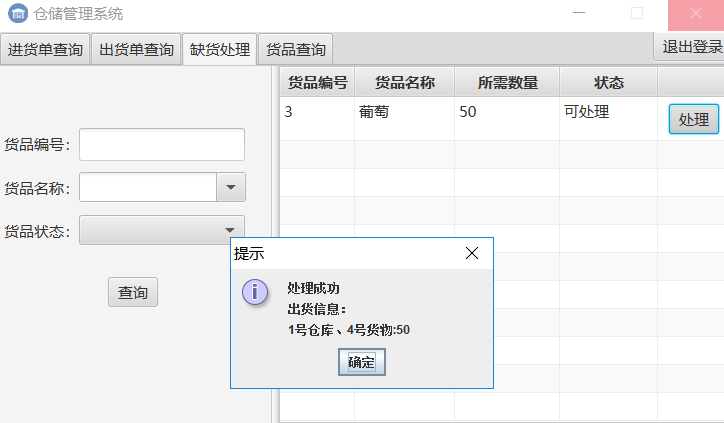


图 4.35 缺货处理

6.经过以上测试，进货、出货审核流程基本正常，货品查询也已实现，最后查询进货、出货后对采购人员、销售人员的反馈。

7.将2号进货单和2号出货单驳回，结果如图 4.36和图 4.37所示。

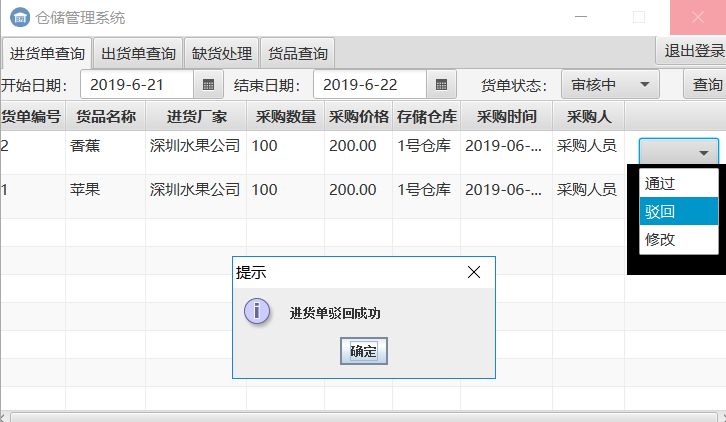


图 4.36 进货单驳回

8.退出登录进入采购界面的查询界面，如图 4.38所示，3号进货单显示已通过，2号进货单显示已驳回，状态正确，说明审核进货单处理成功。

9退出登录进入销售界面的查询界面，如图 4.39所示，3号出货单显示已通过，2号出货单显示已驳回，状态正确，说明审核出货单处理成功。

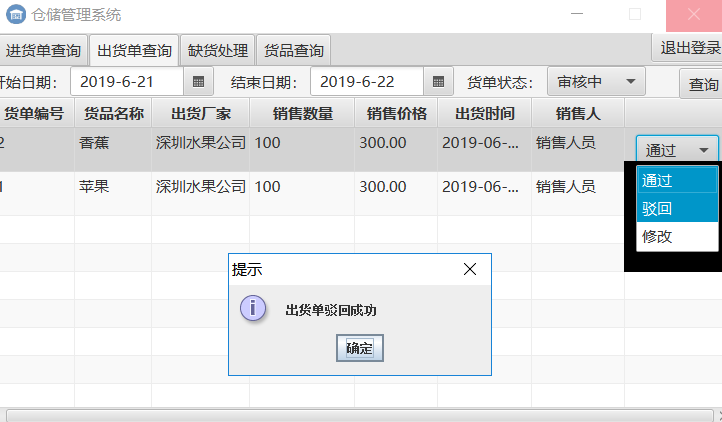


图 4.37 出货单驳回



图 4.38 进货单状态反馈



图 4.39 出货单状态反馈

## 系统设计与实现总结

（1）对系统进行功能需求分析。

（2）对系统进行数据库需求分析并做出数据字典。

（3）根据数据字典做出数据库的E-R图并建立响应数据库。

（4）对系统进行整体框架设计并补充数据库，根据需求加入视图、触发器等。

（5）进行详细设计，书写代码。

# 课程总结

本次实验我是使用java写的，所以功能实现与java类似，再代码实现时基本没遇到问题，主要时数据库的设计，但也通过亲自设计对数据看设计熟悉了许多。

首先是数据库的需求分析，虽然老师上课时讲过，但当亲自设计时对数据项、数据结构的定义还是不太懂，第一次尝试后才发现理解有错，重新写了一遍才设计正确，而数据流找的也很艰难，在写中期报告时并未找到，一直到写完整个实验才对数据由基本理解，才画出数据流图。

而E-R图也画的很艰难，需要根据功能判断各个实体之间的关系，那些属性作为外码也是画E-R图时就要考虑的问题，而之前设计的数据字典的一些不合理之处也是在画E-R图受阻时才发现的，即使如此，E-R图也不一定足够好，此时实验时由于时第一次设计数据库，所以有一些功能上的需求设计的数据库无法实现，直到写代码实现时才发现。

此次实验可以说收获最大的也是对数据库的设计，通过此次实验，我对数据库设计熟悉了许多，尤其是数据字典，不会像此次实验刚开始时直接不知道从何下手了。

# 附录一 建表语句

USE [薛683];

DROP TABLE IF EXISTS [B\_L];

DROP TABLE IF EXISTS [FOLLOW];

DROP TABLE IF EXISTS [FRIENDS];

DROP TABLE IF EXISTS [SUB];

DROP TABLE IF EXISTS [THUMB];

DROP TABLE IF EXISTS [TOPDAY];

DROP TABLE IF EXISTS [MBLOG];

DROP TABLE IF EXISTS [LABEL];

DROP TABLE IF EXISTS [USER];

DROP TABLE IF EXISTS [REPLY];

DROP TABLE IF EXISTS [RREPLY];

DROP TABLE IF EXISTS [MREPLY];

DROP TABLE IF EXISTS [RTHUMB];

--用户表USER (记录所有注册用户的基本信息)

CREATE TABLE [USER](

UID INT,

NAME CHAR(10),

SEX CHAR(2),

BYEAR INT,

CITY CHAR(10),

PRIMARY KEY(UID),

CHECK(SEX='男' OR SEX='女')

);

--分类表LABEL (记录微博平台中所有可能涉及的微博的类型)

CREATE TABLE [LABEL](

LID INT,

LNAME CHAR(6),

PRIMARY KEY(LID)

);

--博文表MBLOG (记录每一篇微博的基本信息)

CREATE TABLE [MBLOG](

BID INT,

TITLE VARCHAR(50),

UID INT,

PYEAR INT,

PMONTH INT,

PDAY INT,

CONT VARCHAR(100),

PRIMARY KEY(BID),

FOREIGN KEY(UID) REFERENCES [USER](UID)

ON DELETE CASCADE

);

--博文标注表B\_L (记录每一篇微博的作者给该微博贴上的分类标签)

CREATE TABLE [B\_L](

BID INT,

LID INT,

PRIMARY KEY(BID,LID),

FOREIGN KEY(BID) REFERENCES [MBLOG](BID)

ON DELETE CASCADE,

FOREIGN KEY(LID) REFERENCES LABEL(LID)

ON DELETE CASCADE

);

--关注表FOLLOW (记录每位用户关注的其他用户)

CREATE TABLE [FOLLOW](

UID INT,

UIDFLED INT,

PRIMARY KEY(UID,UIDFLED),

FOREIGN KEY(UID) REFERENCES [USER](UID)

ON DELETE CASCADE,

FOREIGN KEY(UIDFLED) REFERENCES [USER](UID)

ON DELETE CASCADE

);

--好友表FRIENDS(记录每位用户的好友(可多个))

CREATE TABLE [FRIENDS](

UID INT,

FUID INT,

PRIMARY KEY(UID,FUID),

FOREIGN KEY(UID) REFERENCES [USER](UID)

ON DELETE CASCADE,

FOREIGN KEY(FUID) REFERENCES [USER](UID)

ON DELETE CASCADE

);

--订阅表SUB(记录用户订阅的每一种分类)

CREATE TABLE [SUB](

UID INT,

LID INT,

PRIMARY KEY(UID,LID),

FOREIGN KEY(UID) REFERENCES [USER](UID)

ON DELETE CASCADE,

FOREIGN KEY(LID) REFERENCES LABEL(LID)

ON DELETE CASCADE

);

--点赞表THUMB(记录用户点赞的每一篇微博)

CREATE TABLE [THUMB](

UID INT,

BID INT,

PRIMARY KEY(UID,BID),

FOREIGN KEY(UID) REFERENCES [USER](UID)

ON DELETE CASCADE,

FOREIGN KEY(BID) REFERENCES MBLOG(BID)

ON DELETE CASCADE

);

--头条表TOPDAY(记录每一天的热度排名前十的博文ID号以及该博文在热度前十名中的排名)

CREATE TABLE [TOPDAY](

TYEAR INT,

TMONTH INT,

TDAY INT,

BID INT,

TNO INT,

PRIMARY KEY(TYEAR,TMONTH,TDAY,BID)

ON DELETE CASCADE,

FOREIGN KEY(BID) REFERENCES [MBLOG](BID)

ON DELETE CASCADE

);

# 附录二 仓储管理系统主功能代码

// 进货单审核通过

public static boolean passPurchase(String pNumber) {

final String purchasePass = "update `purchase` set status=1 where number=" + pNumber;

final String infoQuery = "select gid, wid, gnumber from `purchase` where number=" + pNumber;

try {

Data.connection.setAutoCommit(false);

if(!Data.update(purchasePass)) return false; //更改进货单状态

if(!Data.query(infoQuery)) return false; //查询进货货品、进货数量

ResultSet InfoGet = Data.resultSet;

if(InfoGet.next()) {

String gid = InfoGet.getString("gid");

String wid = InfoGet.getString("wid");

String gnumber = InfoGet.getString("gnumber");

if(!updateStorage(gid, wid, gnumber)) { //更新库存

return false;

}

}

Data.connection.commit();

Data.connection.setAutoCommit(true);

} catch (SQLException e) {

try {

Data.connection.rollback();

}

catch (SQLException e1) {

Global.messageDialogShow("回滚失败，数据库可能出错！");

}

System.out.println("更新进货单失败!");

return false;

}

return true;

}

// 出货单通过

public static boolean passSale(String sNumber) throws Exception {

final String infoQuery = "select gname,gnumber from `sale` where number=" + sNumber;

final String salePass = "update `sale` set status=1 where number=" + sNumber;

try {

Data.connection.setAutoCommit(false);

if(Data.query(infoQuery)) {

ResultSet infoGet = Data.resultSet;

if(!infoGet.next()) return false;

String gName = infoGet.getString("gname");

int gNumber = infoGet.getInt("gnumber");

if(isEnough(gName, gNumber)) {

// 货品充足，出货处理

if(!Data.update(salePass)) return false; //更改进货单状态

if(!updateStorage(sNumber, gName, gNumber)) return false; // 更新库存

Data.connection.commit();

Data.connection.setAutoCommit(true);

return true;

}

else {

// 缺货处理

Alert alert = new Alert(AlertType.CONFIRMATION);

alert.setTitle("缺货处理");

alert.setHeaderText(null);

alert.setContentText("货物不足,是否进行缺货登记?");

Optional<ButtonType> result = alert.showAndWait();

if (result.get() == ButtonType.OK){

if(!AuditQuery.insertStockout(sNumber, gName, gNumber,0)) { //插入缺货信息

Global.messageDialogShow("登记失败!");

}

}

Data.connection.commit();

Data.connection.setAutoCommit(true);

throw new Exception();

}

}

}

catch (SQLException e) {

try {

Data.connection.rollback();

}

catch (SQLException e1) {

Global.messageDialogShow("回滚失败，数据库可能出错！");

return false;

}

System.out.println("出货审核失败");

return false;

}

return false;

}

// 缺货处理

public static boolean disposeStockout(String sNumber, String gName, int gNumber){

final String stockoutDelete = "delete from `stockout` where number=" + sNumber;

if(!SaleQuery.isEnough(gName, gNumber)) {

Global.messageDialogShow("缺货中，无法受理！");

return false;

}

try {

Data.connection.setAutoCommit(false);

if(!Data.update(stockoutDelete)) { //删除缺货信息

Global.messageDialogShow("处理失败，请稍后重试！");

return false;

}

if(!SaleQuery.updateStorage(sNumber, gName, gNumber)) { //出货，更新库存

insertStockout(sNumber, gName, gNumber, 1);

Global.messageDialogShow("处理失败，请稍后重试！");

return false;

}

Data.connection.commit();

Data.connection.setAutoCommit(true);

} catch (SQLException e) {

try {

Data.connection.rollback();

}

catch (SQLException e1) {

Global.messageDialogShow("回滚失败，数据库可能出错！");

return false;

}

Global.messageDialogShow("处理失败，请稍后重试！");

return false;

}

return true;

}