



华中科技大学

数据库系统原理实践报告

姓 名: Dracula

学 院:

专 业:

班 级:

学 号:

指导教师:

分数	
教师签名	

2017 年 6 月 16 日

目 录

1 课程任务概述	1
2 实验任务一	2
2.1 任务要求.....	2
2.2 完成过程.....	2
2.2.1 数据定义.....	2
2.2.2 数据更新.....	2
2.2.3 数据查询.....	4
2.2.4 提高题（日期类型数据、查询结果转化、特殊查询）	8
2.2.5 选作题（事务隔离级）	9
2.3 任务总结.....	10
3 实验任务二	11
3.1 任务要求.....	11
3.2 完成过程.....	11
3.3.1 权限控制.....	11
3.3.2 使用 SQL 对数据进行完整性控制.....	14
3.3.3 存储过程及存储函数.....	17
3.3 任务总结.....	18
4 实验任务三	19
4.1 任务要求.....	19
4.2 完成过程.....	19
4.3 任务总结.....	24
5 综合实践任务	26
5.1 系统设计目标.....	26
5.2 需求分析.....	26
5.3 总体设计.....	30
5.4 数据库设计.....	31
5.5 详细设计与实现.....	35
5.6 系统测试.....	38
5.7 系统设计与实现总结.....	47
6 课程总结	48
附录	49

1 课程任务概述

本次数据库系统原理课程的实验任务共有 4 个,其中第 4 个也是最后 1 个任务是一个综合实践。4 个任务的完成难度逐层递增,从最简单的 SQL 语句的使用到最终的基于 CS 模型和 SQL 数据库的图书馆管理系统。

实验任务一:要求熟悉一种 DBMS 软件的安装及使用,并且熟悉并掌握使用 SQL 语言进行表的创建、增加、删除、修改、查询等操作。

实验任务二:在实验任务一的基础上,使用 DBMS 进行权限控制及完整性控制,以及选做的存储过程及存储函数。

实验任务三:了解基本的数据库应用编程方法与技术,为后续的数据库应用系统设计与实现打基础。实现一个出版物管理系统,系统应提供界面友好的出版物信息维护功能和查询功能。

实验任务四(综合实践任务):在课程设计指导老师的指导下,选定一个数据库应用系统的题目,完成需求分析、数据库设计和应用程序设计,并提交相应文档。系统采用客户/服务器(C/S)结构或浏览器/服务器(B/S)结构实现。

2 实验任务一

2.1 任务要求

1. 熟悉一种 DBMS 软件（Microsoft SQL Server / DM）的安装及使用；
2. 熟悉并掌握使用 SQL 语言进行表的创建、增加、删除、修改、查询等操作

2.2 完成过程

2.2.1 数据定义

参照下面的说明创建三个基表：

- (1) GOODS：商品表（商品名称，商品类型）

主码为商品名称；商品类型为电器、文具、服装……

- (2) PLAZA：商场表（商场名称，所在地区）

主码为商场名称；所在地区为汉口、汉阳、武昌……

- (3) SALE：销售价格表（商品名称，商场名称，价格，促销类型）

主码为（商品名称、商场名称）；促销类型为送券、打折，也可为空值，表示当前未举办任何活动；表中记录如（‘哈森皮靴’，‘大洋百货’，300，‘打折’），同一商场针对不同的商品可能采取不同的促销活动。

建表的 SQL 代码如下：

```
CREATE TABLE GOODS (GName CHAR(16) PRIMARY KEY,
                     GType CHAR(6));
CREATE TABLE PLAZA (PName CHAR(16) PRIMARY KEY,
                     PRegion CHAR(6));
CREATE TABLE SALE (GName CHAR(16),
                    PName CHAR(16),
                    Price FLOAT,
                    Discount CHAR(6),
                    PRIMARY KEY (GName, PName),
                    FOREIGN KEY (GName) REFERENCES GOODS (GName),
                    FOREIGN KEY (PName) REFERENCES PLAZA (PName));
```

2.2.2 数据更新

- (1) 用 SQL 语句完成以上三个关系的数据插入；

插入数据的 SQL 语句因篇幅原因详细给出，插入后，三个基表的内容分别如图 2.1、图 2.2 和图 2.3 所示。

- (2) 将 SALE 表中活动类型为打折的记录插入到新表 SALE_CHEAP 中；

创建表 SALE_CHEAP 的 SQL 语句与 SALE 表相同，只需将 TABLE 表明改为 SALE_CHEAP 即可。将 SALE 表中活动类型为打折的记录插入到表 SALE_CHEAP 中的 SQL 语句如下：

```
INSERT INTO SALE_CHEAP
SELECT * FROM SALE where Discount == '打折';
```

插入后，SALE_CHEAP 表的内容如图 2.4 所示。

SELECT * FROM GOODS;

	GNAME	GTYPE
1	Lee*牛仔裤	服装
2	奥利奥*饼干	食品
3	晨光*中性笔	文具
4	创维*电视	电器
5	得力*笔记本	文具
6	格力*空调	电器
7	哈森*皮鞋	服装
8	海尔*冰箱	电器
9	海天*酱油	食品
10	老干妈*辣酱	食品
11	老人头*T恤	服装
12	七匹狼*夹克	服装
13	小天鹅*洗衣机	电器

图 2.1 GOODS 表插入结果

SELECT * FROM PLAZA;

	PNAME	PREGION
1	大洋百货	武昌
2	汉商购物中心	汉阳
3	群光广场	武昌
4	王府井百货	汉口
5	校园超市	武昌
6	新世界百货	汉口

图 2.2 PLAZA 表插入结果

SELECT * FROM SALE;

	GNAME	PNAME	PRICE	DISCOUNT
1	Lee*牛仔裤	大洋百货	389.0	打折
2	Lee*牛仔裤	汉商购物中心	500.0	送券
3	奥利奥*饼干	汉商购物中心	8.0	NULL
4	奥利奥*饼干	校园超市	8.5	NULL
5	晨光*中性笔	汉商购物中心	1.5	NULL
6	晨光*中性笔	校园超市	1.0	NULL
7	创维*电视	群光广场	3188.0	打折
8	得力*笔记本	汉商购物中心	5.0	NULL
9	得力*笔记本	校园超市	4.5	NULL
10	格力*空调	大洋百货	4388.0	打折
11	格力*空调	群光广场	4588.0	打折
12	哈森*皮鞋	大洋百货	300.0	打折
13	海尔*冰箱	汉商购物中心	3888.0	NULL
14	老干妈*辣酱	大洋百货	9.0	NULL
15	老人头*T恤	大洋百货	1200.0	送券
16	七匹狼*夹克	新世界百货	800.0	打折
17	小天鹅*洗衣机	大洋百货	2288.0	NULL

图 2.3 SALE 表插入结果

SELECT * FROM SALE_CHEAP;

	GNAME	PNAME	PRICE	DISCOUNT
1	Lee*牛仔裤	大洋百货	389.0	打折
2	创维*电视	群光广场	3188.0	打折
3	格力*空调	大洋百货	4388.0	打折
4	格力*空调	群光广场	4588.0	打折
5	哈森*皮鞋	大洋百货	300.0	打折
6	七匹狼*夹克	新世界百货	800.0	打折

图 2.4 SALE_CHEAP 表插入结果

- (3) 群光广场的创维电视停止打折活动，价格恢复为 3988，请对 SALE 表和 SALE_CHEAP 表做相应的修改；

更新 SALE(UPDATE)和 SALE_CHEAP(DELETE)执行的 SQL 代码如下：

```
UPDATE SALE
SET Price = 3998, Discount = NULL
WHERE PName = '群光广场' AND
      GName = '创维*电视';
DELETE FROM SALE_CHEAP
WHERE PName = '群光广场' AND
      GName = '创维*电视';
```

- (4) 基于 SALE_CHEAP 表创建一个统计每个打折商品平均价格的视图。

建立视图执行的 SQL 代码如下：

```
CREATE VIEW SALE_CHEAP_AVG (AVGPrice)
AS
SELECT AVG(Price)
FROM SALE_CHEAP;
```

2.2.3 数据查询

- (1) 查询所有没有任何促销活动的商品及其所在的商场，结果按照商品名排序；
此查询所用的 SQL 语句及执行结果如图 2.5 所示。

```
SELECT GName, PName
FROM SALE
WHERE Discount IS NULL
ORDER BY GName;
```



	GNAME	PNAME
1	奥利奥*饼干	汉商购物中心
2	奥利奥*饼干	校园超市
3	晨光*中性笔	汉商购物中心
4	晨光*中性笔	校园超市
5	创维*电视	群光广场
6	得力*笔记本	汉商购物中心
7	得力*笔记本	校园超市
8	海尔*冰箱	汉商购物中心
9	老干妈*辣酱	大洋百货
10	小天鹅*洗衣机	大洋百货

图 2.5 数据查询（1）的 SQL 代码和执行结果

- (2) 查询价格在 200~500 元之间的商品名称、所在的商场名称、价格，结果按照商场名称排序；

此查询所用的 SQL 语句及执行结果如图 2.6 所示。

<pre> SELECT GName, PName, Price FROM SALE WHERE Price<=500 AND Price>=200 ORDER BY PName; </pre>																			
<div>消息 结果集 x</div> <table> <tr> <th></th><th>GNAME</th><th>PNAME</th><th>PRICE</th></tr> <tr> <td>1</td><td>Lee*牛仔裤</td><td>大洋百货</td><td>389</td></tr> <tr> <td>2</td><td>哈森*皮鞋</td><td>大洋百货</td><td>300</td></tr> <tr> <td>3</td><td>Lee*牛仔裤</td><td>汉商购物中心</td><td>500</td></tr> </table>					GNAME	PNAME	PRICE	1	Lee*牛仔裤	大洋百货	389	2	哈森*皮鞋	大洋百货	300	3	Lee*牛仔裤	汉商购物中心	500
	GNAME	PNAME	PRICE																
1	Lee*牛仔裤	大洋百货	389																
2	哈森*皮鞋	大洋百货	300																
3	Lee*牛仔裤	汉商购物中心	500																

图 2.6 数据查询（2）的 SQL 代码和执行结果

- (3) 查询每种商品的商品名称、最低售价、最高售价；

此查询所用的 SQL 语句及执行结果如图 2.7 所示。

<pre> SELECT GName, MIN(Price) MinPrice, MAX(Price) MaxPrice FROM SALE GROUP BY GName; </pre>																																																							
<div>消息 结果集 x</div> <table> <tr> <th></th><th>GNAME</th><th>MINPRICE</th><th>MAXPRICE</th></tr> <tr> <td>1</td><td>Lee*牛仔裤</td><td>389</td><td>500</td></tr> <tr> <td>2</td><td>奥利奥*饼干</td><td>8</td><td>9</td></tr> <tr> <td>3</td><td>晨光*中性笔</td><td>1</td><td>2</td></tr> <tr> <td>4</td><td>创维*电视</td><td>3988</td><td>3988</td></tr> <tr> <td>5</td><td>得力*笔记本</td><td>5</td><td>5</td></tr> <tr> <td>6</td><td>格力*空调</td><td>4388</td><td>4588</td></tr> <tr> <td>7</td><td>哈森*皮鞋</td><td>300</td><td>300</td></tr> <tr> <td>8</td><td>海尔*冰箱</td><td>3888</td><td>3888</td></tr> <tr> <td>9</td><td>老干妈*辣酱</td><td>9</td><td>9</td></tr> <tr> <td>10</td><td>老人头*T恤</td><td>1200</td><td>1200</td></tr> <tr> <td>11</td><td>七匹狼*夹克</td><td>800</td><td>800</td></tr> <tr> <td>12</td><td>小天鹅*洗衣机</td><td>2288</td><td>2288</td></tr> </table>					GNAME	MINPRICE	MAXPRICE	1	Lee*牛仔裤	389	500	2	奥利奥*饼干	8	9	3	晨光*中性笔	1	2	4	创维*电视	3988	3988	5	得力*笔记本	5	5	6	格力*空调	4388	4588	7	哈森*皮鞋	300	300	8	海尔*冰箱	3888	3888	9	老干妈*辣酱	9	9	10	老人头*T恤	1200	1200	11	七匹狼*夹克	800	800	12	小天鹅*洗衣机	2288	2288
	GNAME	MINPRICE	MAXPRICE																																																				
1	Lee*牛仔裤	389	500																																																				
2	奥利奥*饼干	8	9																																																				
3	晨光*中性笔	1	2																																																				
4	创维*电视	3988	3988																																																				
5	得力*笔记本	5	5																																																				
6	格力*空调	4388	4588																																																				
7	哈森*皮鞋	300	300																																																				
8	海尔*冰箱	3888	3888																																																				
9	老干妈*辣酱	9	9																																																				
10	老人头*T恤	1200	1200																																																				
11	七匹狼*夹克	800	800																																																				
12	小天鹅*洗衣机	2288	2288																																																				

图 2.7 数据查询（3）的 SQL 代码和执行结果

- (4) 查询以“打折”方式销售的商品总数超过 2 种的商场名称及其所在地区；

此查询所用的 SQL 语句及执行结果如图 2.8 所示。

```
SELECT PName, PRegion
FROM PLAZA, (SELECT PName, COUNT(Discount) DcNum
FROM SALE
WHERE Discount='打折'
GROUP BY PName) AS DcCount (DcPName, DcNum)
WHERE PName=DcPName AND
DcNum>2;
```

消息 结果集

	PNAME	PREGION
1	大洋百货	武昌

图 2.8 数据查询（4）的 SQL 代码和执行结果

- (5) 查询以“老”字开头的所有商品的名称；
此查询所用的 SQL 语句及执行结果如图 2.9 所示。

```
SELECT GName
FROM GOODS
WHERE GName like '老%';
```

消息		结果集
		GNAME
1		老干妈*辣酱
2		老人头*T恤

图 2.9 数据查询 (5) 的 SQL 代码和执行结果

- (6) 查询同时销售“晨光*中性笔”和“得力*笔记本”的商场名称；
此查询所用的 SQL 语句及执行结果如图 2.10 所示。

```
SELECT PName
FROM PLAZA
WHERE 2=(SELECT COUNT(GName)
FROM SALE
WHERE SALE.PName=PLAZA.PName AND
(SALE.GName='晨光*中性笔' OR
SALE.GName='得力*笔记本'));
```

消息		结果集
		PNAME
1		校园超市
2		汉商购物中心

图 2.10 数据查询 (6) 的 SQL 代码和执行结果

- (7) 查询未举办任何活动的商场；
此查询所用的 SQL 语句及执行结果如图 2.11 所示。

```
SELECT DISTINCT PName
FROM SALE SALE1
WHERE 0=(SELECT COUNT(Discount)
FROM SALE SALE2
WHERE SALE2.PName=SALE1.PName);
```

消息		结果集
		PNAME
1		校园超市

图 2.11 数据查询 (7) 的 SQL 代码和执行结果

- (8) 查询出售商品种类最多的商场名称；
此查询所用的 SQL 语句及执行结果如图 2.12 所示。

<pre> SELECT PName FROM PLAZA WHERE 0=(SELECT COUNT(Discount) FROM SALE WHERE SALE.PName=PLAZA.PName); </pre>	
消息	结果集
	PNAME
1	王府井百货
2	校园超市

图 2.12 数据查询（8）的 SQL 代码和执行结果

(9) 查询出售商品类型最多的商场名称；

此查询所用的 SQL 语句及执行结果如图 2.13 所示。

<pre> SELECT PName FROM SALE GROUP BY PName HAVING COUNT(GName) >= ALL(SELECT COUNT(GName) FROM SALE GROUP BY PName); </pre>	
消息	结果集
	PNAME
1	大洋百货

图 2.13 数据查询（9）的 SQL 代码和执行结果

(10) 查询所销售的商品包含了“校园超市”所销售的所有商品的商场名称。

此查询所用的 SQL 语句及执行结果如图 2.14 所示。

<pre> SELECT PName FROM PLAZA WHERE NOT EXISTS (SELECT * FROM GOODS WHERE GName IN (SELECT GName FROM SALE WHERE PName = '校园超市') AND NOT EXISTS (SELECT * FROM SALE WHERE SALE.PName = PLAZA.PName AND SALE.GName = GOODS.GName)) </pre>	
消息	结果集
	PNAME
1	校园超市
2	汉商购物中心

图 2.14 数据查询（10）的 SQL 代码和执行结果

2.2.4 提高题（日期类型数据、查询结果转化、特殊查询）

(1) 解决以下有关日期时间类型的问题：

- 在 SALE 表中增加一活动截止时间列，查出所有在整点时刻截止活动的商品；添加时间列所用的 SQL 语句如下：

```
ALTER TABLE SALE
ADD Deadline TIMESTAMP;
```

- 查询活动截止时间在本月最后一天的 SALE 记录；此查询所用的 SQL 语句如下：

```
SELECT *
FROM SALE
WHERE DATEDIFF(DY, Deadline, LAST_DAY(CURRENT_TIMESTAMP())) = 0;
```

- 查询截止时间在 2016 年的 SALE 记录；此查询所用的 SQL 语句如下：

```
SELECT *
FROM SALE
WHERE YEAR(Deadline) = 2016;
```

- 查询本月的最后一天；

此查询所用的 SQL 语句如下：

```
SELECT LAST_DAY(CURRENT_TIMESTAMP());
```

- 将所有打折商品的活动截止时间推迟一个小时。

此查询所用的 SQL 语句如下：

```
UPDATE SALE
SET Deadline = TIMESTAMPADD(SQL_TSI_HOUR, 1, Deadline);
COMMIT;
```

(2) 将查询结果的某些列的某些值转换成特殊的形式（例如：若商场所在地区为武昌，则在所在地区列中显示“附近”；否则显示“遥远”）

此查询所用的 SQL 语句及执行结果如图 2.15 所示。

SELECT PName, PRegion, REPLACE(REPLACE(REPLACE(PRegion, '武昌', '附近'), '汉阳', '遥远'), '汉口', '遥远') Distance FROM PLAZA;			
<			
消息 结果集			
	PNAME	PREGION	DISTANCE
1	大洋百货	武昌	附近
2	汉商购物中心	汉阳	遥远
3	群光广场	武昌	附近
4	王府井百货	汉口	遥远
5	校园超市	武昌	附近
6	新世界百货	汉口	遥远

图 2.15 提高题查询（2）的 SQL 代码和执行结果

(3) 查询价格个位数为 8 元的商品名称、所在商场名称和价格；

此查询所用的 SQL 语句及执行结果如图 2.16 所示。

(4) 假设品牌名不可能包含“*”号，查询所有商品的品牌。

此查询所用的 SQL 语句及执行结果如图 2.17 所示。

<pre>SELECT GName, PName, Price FROM SALE WHERE FLOOR(Price)%10 = 8;</pre>			
<div>消息 结果集</div>			
	GNAME	PNAME	PRICE
1	奥利奥*饼干	汉商购物中心	8.0
2	奥利奥*饼干	校园超市	8.5
3	创维*电视	群光广场	3998.0
4	格力*空调	大洋百货	4388.0
5	格力*空调	群光广场	4588.0
6	海尔*冰箱	汉商购物中心	3888.0
7	小天鹅*洗衣机	大洋百货	2288.0

图 2.16 提高题查询（3）的 SQL 代码和执行结果

<pre>SELECT SUBSTR(GName, 1, INSTR(GName, '*') - 1) BRAND FROM GOODS;</pre>	
<div>消息 结果集</div>	
	BRAND
1	Lee
2	奥利奥
3	晨光
4	创维
5	得力
6	格力
7	哈森
8	海尔
9	海天
10	老干妈
11	老人头
12	七匹狼
13	小天鹅

图 2.17 提高题查询（4）的 SQL 代码和执行结果

2.2.5 选做题（事务隔离级）

对以下两个事务，分别在不同事务的隔离级下并发执行，注意尝试各种可能的并发顺序，记录运行结果并分析原因。

1. T1 的 SQL 代码分别如下：

```
START TRANSACTION;
UPDATE R SET B=22 WHERE A='C1';
INSERT INTO R VALUES ('C4', 0);
UPDATE R SET B=38 WHERE A='C1';
COMMIT;
```

2. T2 的 SQL 代码分别如下：

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;
SELECT SUM(B) FROM R;
SELECT AVG(B) FROM R;
COMMIT;
```

进行实验时，为了实现同时执行两个事务，需要开启两个“DM 管理工具”的窗口，每个窗口运行一个事务（单步运行），即可达到两个事务交叉运行的目的。在 DM 数据库中，事务的隔离级分为 4 个等级：Read Uncommitted、Repeatable Read、READ Committed 和 Serializable，这四个等级隔离程度依次递增。

本个子任务有诸多可能，限于篇幅不一一陈述。以代码中的 Serializable 为隔离级，即序列化（最高隔离级）为例。Serializable 保证了事务从开始执行到执行结束整个执行过程中读到的是“同一个”数据库，即原始数据库的一个副本。对于本子任务中两个事务 T1 和 T2，一旦 T2 开始执行，无论在其执行过程中 T1 执行了多少个 SQL 改变表 R，T2 计算出来的 SUM(B) 和 AVG(B) 都是一致的，都是根据 T2 开始执行时表 R 的数据计算的。例如，假如 T1 和 T2 同时开始执行，即 T2 读到的是最原始的 R 表数据，无论二者的执行顺序如何，其最终 T2 的运行结果均如图 2.18 所示。

	SUM(B)			AVG(B)	
1	150			1	50.000000

图 2.18 事务隔离级测试用例运行结果

2.3 任务总结

在完成任务一的过程中，主要遇到了两大问题：

一是提高题中的第一题，在数据库中对时间类型的数据进行操作是课程上老师没有提及的部分，所以只能查阅达梦数据库自带的《DM7 程序员手册.pdf》，找到相关的时间数据类型以及对应的时间操作函数，来完成任务中的与时间类型相关的数据查询和操作；

二是选做题中的事务隔离级，由于不同的 DBMS 对事务隔离级的支持有所不同，语法也有些区别，因此也需要查阅《DM7 程序员手册.pdf》来获悉，此外，遇到的最大问题其实是一开始的时候，并不知道如何并行地执行两个事务，总是 start 了一个事物再 start 另一个事务时出现报错，报错信息如图 2.19 所示。

[执行语句2]:
START TRANSACTION;
执行失败(语句2)
试图在事务运行中, 改变其属性

图 2.19 单窗口运行第二个事务报错信息

为了解决上述问题，正如 2.2.4 中所说，必须开启两个“DM 管理工具”窗口，相当于开启两个进程。

3 实验任务二

3.1 任务要求

1. 在第一次实验的基础上，使用 Microsoft SQL SERVER 或 DM DBMS 进行权限控制及完整性控制实验；
2. 选作：查阅文档帮助并编写存储过程或函数。

3.2 完成过程

3.3.1 权限控制

- (1) 查阅联机帮助文档，学习在 DBMS 中创建新用户的方法。

查阅《DM7 程序员手册.pdf》，了解创建用户的语法和要求。

- (2) 在以上基础上完成 P153 习题 7，实现对不同用户的权限授予与回收。

首先创建两张基本表，包括员工表 STAFF 和部门表 DEPARTMENT，创建两张表的 SQL 语句如下：

```
CREATE TABLE STAFF (SNum CHAR(20),
                     SName CHAR(10),
                     Age INT,
                     Duty CHAR(20),
                     Wage FLOAT,
                     DNum CHAR(20));

CREATE TABLE DEPARTMENT (DNum CHAR(20),
                          DName CHAR(20),
                          Manager CHAR(10),
                          Address CHAR(50),
                          Phone CHAR(16));
```

然后，根据习题要求，创建习题中涉及到的用户，SQL 语句如下所示：

```
CREATE USER WangMing IDENTIFIED BY wm12345678
LIMIT CONNECT_TIME 3;
CREATE USER LiYong IDENTIFIED BY ly12345678
LIMIT CONNECT_TIME 3;
CREATE USER LiuXing IDENTIFIED BY lx12345678
LIMIT CONNECT_TIME 3;
CREATE USER ZhangXin IDENTIFIED BY zx12345678
LIMIT CONNECT_TIME 3;
CREATE USER ZhouPing IDENTIFIED BY zp12345678
LIMIT CONNECT_TIME 3;
CREATE USER YangLan IDENTIFIED BY yl12345678
LIMIT CONNECT_TIME 3;
```

- ① 用户王明对两个表有 SELECT 权限

赋予权限的 SQL 语句如下：

```
GRANT SELECT ON TABLE STAFF TO WangMing;
GRANT SELECT ON TABLE DEPARTMENT TO WangMing;
```

- ② 用户李勇对两个表有 INSERT 和 DELETE 权限

赋予权限的 SQL 语句如下：

```
GRANT INSERT, DELETE ON TABLE STAFF TO LiYong;
GRANT INSERT, DELETE ON TABLE DEPARTMENT TO LiYong;
```

③ 每个职工只对自己的记录有 SELECT 权限

注意，针对此题，在 DM 数据库下可执行即达梦支持的 SQL 语句如下：

```
CREATE VIEW MYSELF
AS
SELECT * FROM STAFF
WHERE SName = USER;
GRANT SELECT ON MYSELF TO USER;
```

对于其他数据库，一种更简洁地 SQL 语句如下：

```
GRANT SELECT ON TABLE STAFF
WHEN USER() = SName
TO ALL;
```

④ 用户刘星对职工表有 SELECT 权力，对工资字段具有更新权

赋予权限的 SQL 语句如下：

```
GRANT SELECT, UPDATE (Wage) ON TABLE STAFF TO LiuXing;
```

⑤ 用户张新具有修改这两个表的结构权力

```
GRANT ALTER TABLE ON TABLE STAFF TO ZhangXin;
GRANT ALTER TABLE ON TABLE DEPARTMENT TO ZhangXin;
```

⑥ 用户周平具有对两个表所有权力（读，插，改，删数据），并具有给其他用户授权的权力

```
GRANT ALL PRIVILEGES ON TABLE STAFF TO ZhouPing WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON TABLE DEPARTMENT TO ZhouPing WITH GRANT OPTION;
```

⑦ 用户杨兰具有从每个部门职工中 SELECT 最高工资、最低工资、平均工资的权力，他不能查看每个人的工资

```
CREATE VIEW DWAGE
AS
SELECT DEPARTMENT.DName, MAX(Wage), MIN(Wage), AVG(Wage)
FROM STAFF, DEPARTMENT
WHERE STAFF.DNum = DEPARTMENT.DNum
GROUP BY DEPARTMENT.DName;
```

(3) 设计一些语句，验证上述权限控制操作的效果。

限于篇幅，只对用户 WangMing 和用户 LiYong 的权限进行验证。

① 验证用户王明只对两个表有 SELECT 权限

首先检验用户 WangMing 是否具有 SELECT 权限，执行的 SQL 语句和执行结果如图 2.1 所示。

SELECT * FROM SYSDBA.STAFF;

	SNUM	SNAME	AGE	DUTY	WAGE	DNUM
1	1	WangMing	26	manager	5000.0	1
2	2	LiuXing	26	clerk	3000.0	1
3	3	ZhangXin	26	clerk	3000.0	1
4	4	LiYong	26	manager	5000.0	2
5	5	ZhouPing	26	clerk	3000.0	2
6	6	YangLan	26	clerk	3000.0	2

图 3.1 验证 WangMing 的 SELECT 权限

然后验证用户 WangMing 是否具有 INSERT 权限，执行的 SQL 语句和执行结果如图 3.2 所示。显然，WangMing 只具有 SELECT 权限，不具有 INSERT 权限，所以对 WangMing 的权限控制操作是正确的。

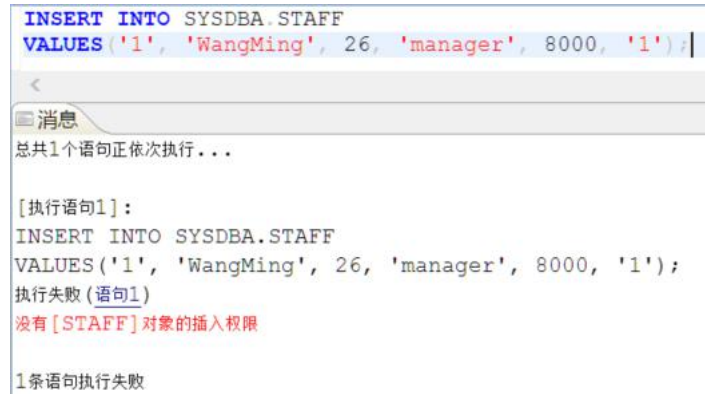


图 3.2 验证 WangMing 的 SELECT 权限

- ② 验证用户李勇对两个表只具有 SELECT 的权限

执行的 SQL 语句和执行结果如图 3.3 所示。

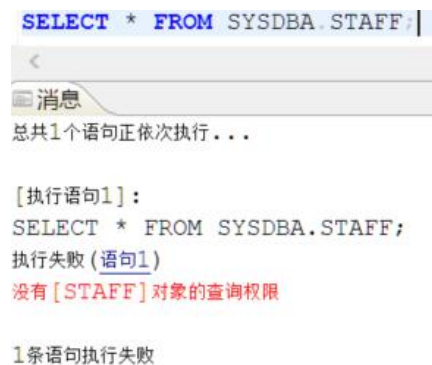


图 3.3 验证 LiYong 的 SELECT 权限

- ③ 不再进行验证
- ④ 验证用户刘星对职工表有 SELECT 权力，对工资字段具有更新权
只验证用户 LiuXing 对工资字段的更新权力，结果如图 3.4 所示。

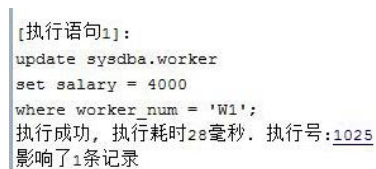


图 3.4 验证用户 LiuXing 对工资字段的 Update 权利

- ⑤ 验证用户张新具有修改这两个表的结构的权利

以职工表为例，执行的 SQL 语句和执行结果如图 3.5 所示。

```
alter table sysdba.worker
add
phone_num char(20);
commit;
```

消息

总共2个语句正依次执行...

[执行语句1]:

```
alter table sysdba.worker
add
phone_num char(20);
```

执行成功, 执行耗时28毫秒. 执行号:1052
影响了0条记录

[执行语句2]:

```
commit;
```

执行成功, 执行耗时1毫秒. 执行号:1053
影响了0条记录

2条语句执行成功

图 3.5 验证用户张新具有修改表的结构权力

⑥ 验证用户周平具有对两个表所有权力（读，插，改，删数据），并具有给其他用户授权的权力

检查对其他用户授予权限的权限，执行的 SQL 语句和执行结果如图 3.6。

```
grant select
on table sysdba.worker
to zhangxin;
--没有的权限
```

消息

总共1个语句正依次执行...

[执行语句1]:

```
grant select
on table sysdba.worker
to zhangxin;
```

执行成功, 执行耗时4毫秒. 执行号:1070
影响了0条记录

1条语句执行成功

图 3.6 验证用户周平具有授权的权力

⑦ 不再进行验证

3.3.2 使用 SQL 对数据进行完整性控制

(1) 以 P173 习题 6 中的表为基础，定义以下完整性约束：

① 定义每个模式的主码。

定义模式主码的 SQL 语句如下：

```
ALTER TABLE STAFF
ADD CONSTRAINT StaffKey PRIMARY KEY (SNum);
ALTER TABLE DEPARTMENT
```

```
ADD CONSTRAINT DepartmentKey PRIMARY KEY (DNum);
```

② 定义参照完整性约束，要求：当部门号改变时，该部门职工记录的部门号也做出相应改变；当部门被删除时，将该部门职工记录的部门号置空。

定义参照完整性约束的 SQL 语句如下：

```
ALTER TABLE STAFF
ADD CONSTRAINT ForeignKey FOREIGN KEY (DNum) REFERENCES DEPARTMENT (DNum)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

③ 职工年龄不能超过 60 岁。

定义该用户定义完整性约束的 SQL 语句如下：

```
ALTER TABLE STAFF
ADD CONSTRAINT AgeRangeCheck CHECK (Age <= 60);
```

④ 职工姓名和部门名称都不允许取空值。

定义该用户定义完整性约束的 SQL 语句如下：

```
ALTER TABLE STAFF
ADD CONSTRAINT SNameNullCheck CHECK (SName IS NOT NULL);
```

⑤ 部门名称不允许重复。

定义该用户定义完整性约束的 SQL 语句如下：

```
ALTER TABLE DEPARTMENT
ADD CONSTRAINT DNameUniCheck UNIQUE (DName);
```

⑥ 同一个部门不应该有一个以上的经理，并且从职工表查询得到的部门经理的信息应该与从部门表查询得到的信息一致。如果由于用户的某个操作导致了不一致现象，则以职工表的信息为准。

该约束相对复杂，无法通过简单的约束语句实现，只能利用触发器实现该约束条件，且需要两个触发器分别作用于职工表 STAFF 和部门表 DEPARTMENT。两个触发器的 SQL 代码如下：

```
CREATE OR REPLACE TRIGGER OneMoreBoss
BEFORE UPDATE OF Duty OR INSERT ON STAFF
FOR EACH ROW
WHEN (:NEW.Duty = 'manager')
DECLARE
Total INT;
One More Manager EXCEPTION FOR -20001;
BEGIN
Total = 0;
SELECT COUNT(SNum) INTO Total
FROM STAFF
WHERE Duty = 'manager' AND DNum = :NEW.DNum AND SNum != :NEW.SNum;
IF Total >= 1
THEN RAISE One More Manager;
END IF;
IF Total = 0
THEN
UPDATE DEPARTMENT
SET Manager = :NEW.SName
WHERE DNum = :NEW.DNum;
END IF;
```

```

END;

CREATE OR REPLACE TRIGGER BossUnchange
BEFORE UPDATE OF Manager ON DEPARTMENT
FOR EACH ROW
DECLARE Department_Change_Manager EXCEPTION FOR -20002;
BEGIN
RAISE Department_Change_Manager;
END

```

(2) 以 P173 习题 6 中的表为基础，定义以下完整性约束：

① 验证职工年龄不能超过 60 岁。

向职工表 STAFF 表插入一个年龄大于 60 岁的员工记录，执行的 SQL 语句：

```

INSERT INTO STAFF
VALUES ('xxx', 'Test', 61, 'manager', 5000, '1');

```

执行结果如图 3.7 所示。

```

执行失败 (语句1)
违反CHECK约束[CONS134218807]

1条语句执行失败

```

图 3.7 验证年龄不超过 60 岁约束

② 验证职工姓名和部门名称都不允许取空值。

向职工表 STAFF 表中插入一条职工姓名为空的记录，执行的 SQL 语句为：

```

INSERT INTO STAFF
VALUES ('xxx', null, 56, 'manager', 5000, '1');

```

执行结果如图 3.8 所示。

```

执行失败 (语句1)
违反列[NAME]非空约束

```

图 3.8 验证职工姓名非空约束

向部门表 DEPARTMENT 表中插入一条部门姓名为空的记录，执行的 SQL 语句如下，执行结果如图 3.9 所示：

```

INSERT
INTO DEPARTMENT
VALUES ('1', 'HumanResource', 'WangMing', 'HubeiWuhan', '18888888888');

```

```

执行失败 (语句1)
违反列[NAME]非空约束

```

图 3.9 验证部门名非空约束

③ 验证部门名称不允许重复

向部门表 DEPARTMENT 表插入一条已有记录，执行的 SQL 语句如下：

```

INSERT
INTO DEPARTMENT
VALUES ('1', 'HumanResource', 'WangMing', 'HubeiWuhan', '18888888888');

```

执行时会报错，提示违反了唯一性约束。

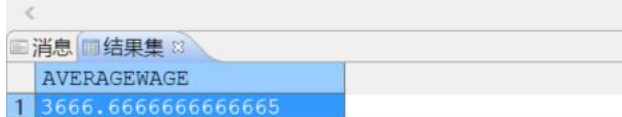
3.3.3 存储过程及存储函数

- ① 根据输入的部门名称返回一个部门的平均工资；

实现该功能的存储函数的 SQL 和函数功能测试的结果如图 3.10 所示。

```
CREATE OR REPLACE FUNCTION AvgWage
(Name CHAR(20))
RETURN FLOAT
AS DECLARE
    Aid CHAR(20);
    AvgWage FLOAT;
    GetId_Sql VARCHAR :=
        'SELECT DNum
        FROM DEPARTMENT
        WHERE DName = ?;';
    GetAvg_Sql VARCHAR :=
        'SELECT AVG(Wage)
        FROM STAFF
        WHERE DNum = ?;';
BEGIN
    EXECUTE IMMEDIATE GetId_Sql
    INTO Aid USING Name;
    EXECUTE IMMEDIATE GetAvg_Sql
    INTO AvgWage USING Aid;
    RETURN AvgWage;
END;

SELECT AvgWage('Technology') AS AverageWage
```



消息 结果集 3	
AVERAGEWAGE	
1	3666.666666666665


图 3.10 存储函数（1）代码及功能测试

- ② 对所有员工进行工资调整：IT 部门的员工工资上调 50%；销售部门的员工工资上调 20%倍；其他部门的员工工资上调 10%；部门经理的工资上调 1.5 倍；同时满足两个调薪条件的员工按调薪幅度高者调薪。

实现该功能的存储过程的 SQL 和过程功能测试的结果如图 3.11 所示。

```
CREATE OR REPLACE PROCEDURE ImproveWage()
AS
BEGIN
    UPDATE STAFF
    SET Wage = Wage * 1.5
    WHERE DNum = (SELECT DNum FROM DEPARTMENT WHERE DName = 'Technology');
    UPDATE STAFF
    SET Wage = Wage * 1.2
    WHERE Duty = 'clerk' AND DNum = (SELECT DNum FROM DEPARTMENT WHERE DName = 'HumanResource');
    UPDATE STAFF
    SET Wage = Wage * 1.5
    WHERE Duty = 'manager' AND DNum != (SELECT DNum FROM DEPARTMENT WHERE DName = 'Technology');
END;

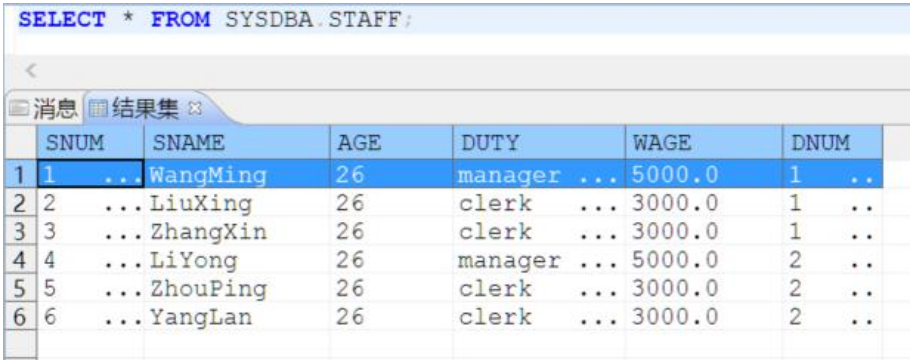
ImproveWage();
Commit;
SELECT * FROM STAFF;
```



	SNUM	SNAME	AGE	DUTY	WAGE	DNUM
1	1	...WangMing	26	manager	7500.0	1
2	2	...LiuXing	26	clerk	3600.0	1
3	3	...ZhangXin	26	clerk	3600.0	1
4	4	...LiYong	26	manager	7500.0	2
5	5	...ZhouPing	26	clerk	4500.0	2
6	6	...YangLan	26	clerk	4500.0	2

图 3.11 存储过程（2）代码及功能测试

而未进行工资调整前，员工表 STAFF 表的数据如图 3.12 所示，经过与图 3.11 进行对比，可以得知所编写的存储过程，成功实现了题目要求的工资调整方案。



The screenshot shows a SQL query window with the command `SELECT * FROM SYSDBA.STAFF;` and its results. The results are displayed in a table with columns: SNUM, SNAME, AGE, DUTY, WAGE, and DNUM. There are 6 rows of data.

	SNUM	SNAME	AGE	DUTY	WAGE	DNUM
1	1	WangMing	26	manager	5000.0	1
2	2	LiuXing	26	clerk	3000.0	1
3	3	ZhangXin	26	clerk	3000.0	1
4	4	LiYong	26	manager	5000.0	2
5	5	ZhouPing	26	clerk	3000.0	2
6	6	YangLan	26	clerk	3000.0	2

图 3.12 未进行工资调整前员工表

3.3 任务总结

在完成任务二的过程中，主要遇到了两大问题：

一是用户管理的验证部分，以新建的用户登录 DM 数据库失败。。当我用 SYSDBA 用户成功执行了用户创建和用户授权的 SQL 语句后，使用新建的用户登录 DM 数据库来验证，总是登录失败，如图 3.13 所示。上网查了下原因，发现时达梦要求口令必须是大写，按照网上所说将口令全改为大写时终于成功登陆；

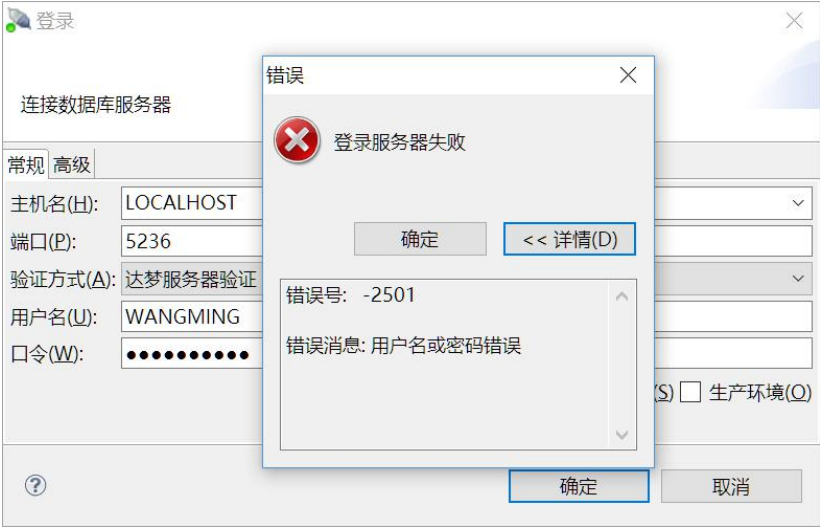


图 3.13 新建用户登录失败

二是达梦不支持断言，这也是我在查阅《DM7 程序员手册.pdf》后获悉的。因此有些情况下的约束使用 DM 并不方便实现，只能使用较为复杂的触发器，查阅相关手册了解了 DM 中触发器的实现语句和方法后，又看了几个触发器的例子，才有了了解决实验中最后一个完整性约束的思路；

三是存储过程和函数，两个一开始完全不清楚的概念，最终通过学习手册和请教同学解决了问题。

4 实验任务三

4.1 任务要求

1. 编程工具和数据库编程接口任选（本人选择了 Qt 和 MySQL）
2. 实现一个出版物管理系统。出版物的信息包括标题、作者、出版年份、出版机构、出版地、页数等属性。系统应提供界面友好的出版物信息维护功能和查询功能。其中，查询功能应采用尽可能灵活的检索方式以满足用户的查询需求。

4.2 完成过程

1. 设计数据库

本任务的数据库相对简单，只涉及到一个出版物的信息数据库，因此只需建一张表，包括 ISBN(ID)、标题、作者、出版年份、出版机构、出版地、页数共 7 个属性。其中出版年份为日期(Date)数据类型，页数为整数型(Int)数据类型，其余属性均为字符型数据类型。根据实际情况，使用 MySQL 建立出版物信息数据库和对应出版物信息表 Books 的 SQL 语句如下：

```
CREATE DATABASE `Publications`;  
CREATE TABLE `Books` (  
  `id` char(13) PRIMARY KEY,  
  `title` char(36),  
  `author` char(10),  
  `pubdate` date,  
  `publisher` char(36),  
  `pubcity` char(10),  
  `pages` int(11));
```

设计好数据库后，从京东网站上找出一些热门图书信息作为测试数据，插入到新建的表 Book 中，插入的 SQL 语句和记录如下：

```
insert into Books values  
( '9787530216194', '人民的名义', '周梅森', '2017-01-01', '北京十月文艺出版社', '北京市', 384),  
( '9787550013247', '摆渡人', '克莱儿·麦克福尔', '2015-06-01', '百花洲文艺出版社', '南昌市', 280),  
( '9787201094014', '浮生六记', '沈复', '2015-08-01', '天津人民出版社', '天津市', 217),  
( '9787302423287', '机器学习', '周志华', '2016-01-01', '清华大学出版社', '北京市', 368),  
( '9787115373557', '数学之美（第二版）', '吴军', '2014-11-01', '人民邮电出版社', '北京市', 312),  
( '9787121198854', '高性能 MySQL', 'Baron Schwartz, Peter Zaitsev, Vadim Tkachenko', '2013-04-01', '电子工业出版社', '北京市', 764);
```

这样，便建好了本任务所需要的出版物数据库，并向其中插入了 6 本出版物信息作为测试数据。

2. 设计用户界面

在设计界面时，采用了传统的菜单栏式的图形界面，用户可以通过选择菜单栏中的各个选项来执行各种操作。初始界面如图 4.1 所示。

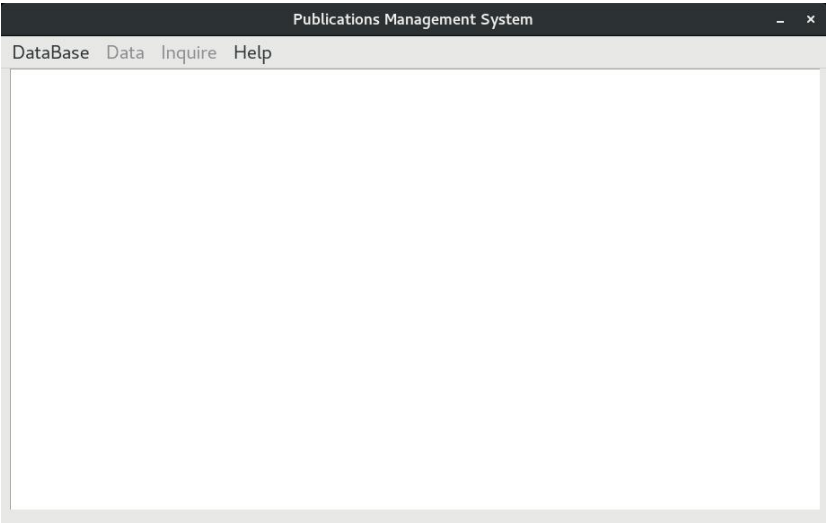


图 4.1 初始化界面

根据任务要求，用户界面需要提供数据维护和数据查询的功能，因此菜单栏中分别用“Data”和“Inquire”两个菜单栏选项提供相应功能。此外，还应包含一个提供连接数据库、保存数据库功能的“DataBase”选项和提供程序、作者信息的“Help”选项。注意到，如图 4.1 所示，在未连接数据库之前，“Data”和“Inquire”选项是灰色的即不可操作的。

3. “DataBase”菜单选项设计

“DataBase”主要提供连接数据库和保存到数据库的功能。连接数据库容易理解，保存到数据库则是基于数据库的“事务”特性实现的。每次连接到数据库，都会开启一个事务；每次保存到数据库则会提交当前事务并开启一个新的事务。连接数据库后，将会显示当前数据库中所有的出版物的相关信息，如图 4.2。

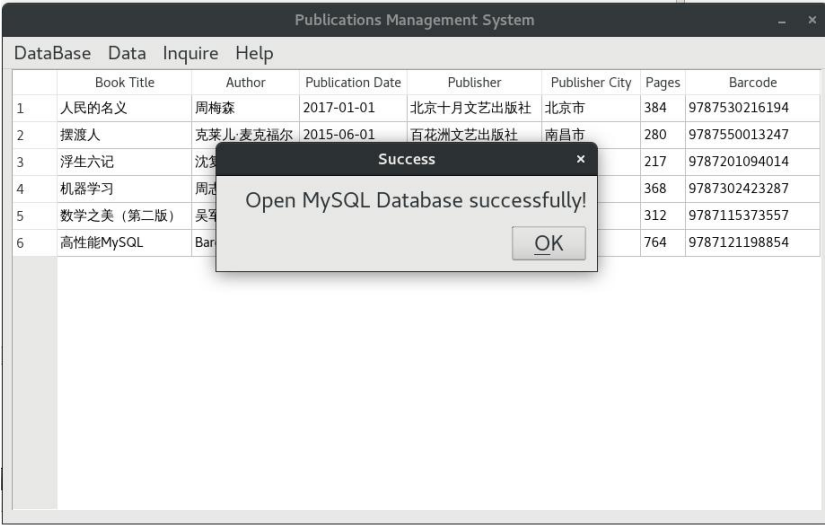


图 4.2 “Connect DB” 执行结果

4. “Data” 菜单选项设计

“Data” 主要提供数据维护的功能，包括录入出版物信息 “Input”、修改出版物信息 “Modify”、删除出版物信息 “Delete” 和撤销之前的操作 “Revert”。对于 “Input”，实现相对简单，当点击 “Input” 子菜单栏选项时，会弹出一个 “Input” 对话框，在对话框中输入新的出版物信息然后点击 “Confirm” 按钮即可。对于 “Modify” 和 “Delete”，为了方便用户的操作，当选择这两个子菜单栏选项时，程序会自动识别当前被选中的出版物，然后在弹出的对话框中自动填充相应出版物信息，如当选中 “机器学习” 这本书，点击了 “Modify” 后弹出的对话框如图 4.3 所示，注意 “Barcode” 作为出版物的唯一标识不可修改，其余内容均可修改；点击了 “Delete” 弹出的对话框如。对于 “Revert”，其功能与 3 中提到的 “Save to DB” 正好相反，“Revert” 则是回滚(rollback)当前事务，并再开启一个新的事物，“Revert” 和 “Save to DB” 一起对用户提供了撤销操作和保存操作的功能。

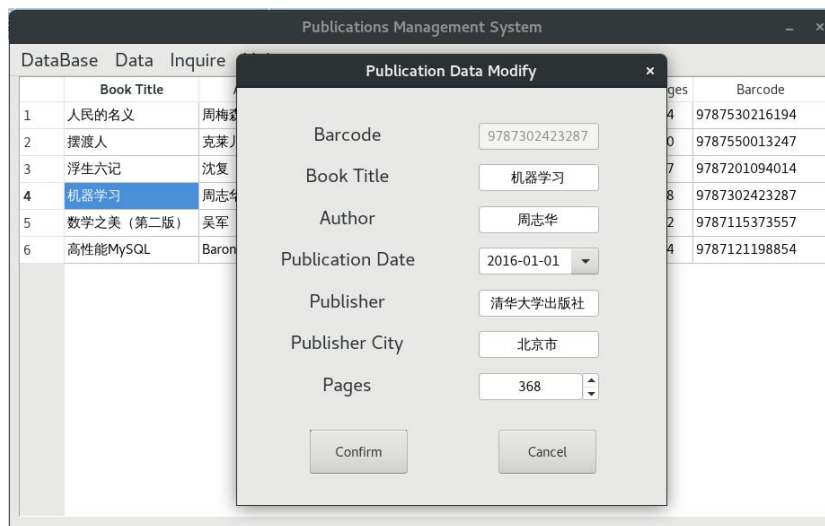


图 4.3 “Modify” 对话框

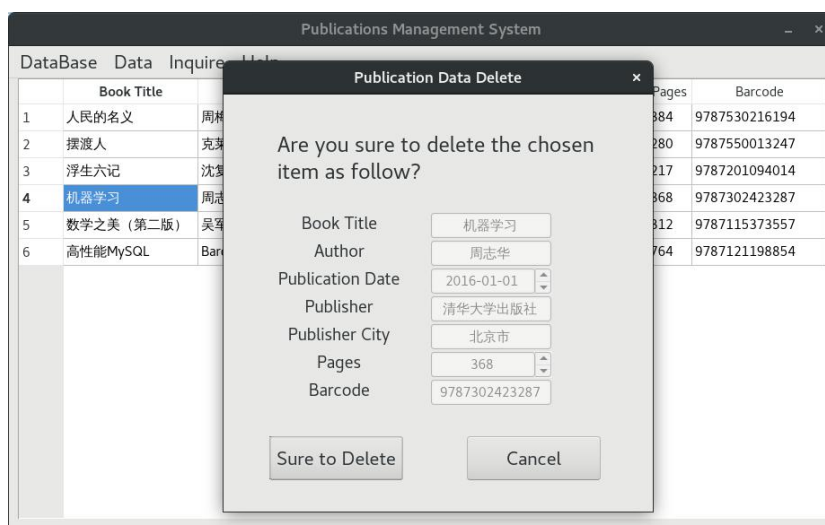


图 4.4 “Delete” 对话框

5. “Inquire” 菜单选项

“Inquire” 主要提供数据查询的功能，包括 “Select All” 和 “Select Ones” 两个子菜单栏选项。“Select All” 实际上就是 “Select * from Books”，显示当前数据库中所有的出版物的相关信息，其具体的效果跟连接数据库后的显示是一样的。

“Select Ones” 则是较为复杂的查询功能，会弹出一个如图 4.5 所示的对话框，该对话的查询条件主要可分为 3 类：一是出版日期区间查询，选中 “Publication” 前面的 CheckBox 表明使用该条件，设置 “From” 和 “To” 两个日期输入框即可；二是图书页数区间查询，选中 “Pages” 前面的 CheckBox 表明使用该条件（图 4.5 中未使用该条件）；三是书名、作者、出版社、出版社所在城市和 ISBN 文本类查询，通过 “+” 和 “-” 可手动增减该类查询条件（最多支持 5 个条件），每个查询条件可选择与前面查询条件的关系（与或非），选择查询的条目，每个条目支持两个查询子条件，并可选择是精确查询 “=” 还是模糊查询 “like”。

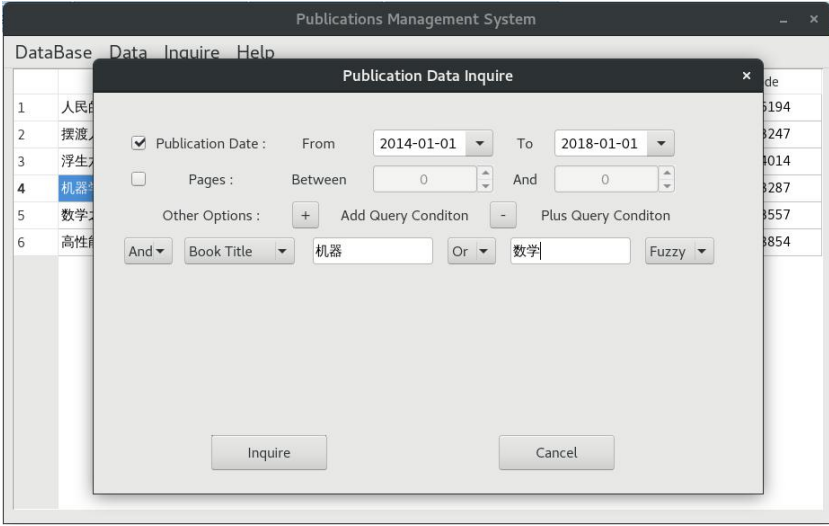


图 4.5 “Select Ones” 对话框

	Book Title	Author	Publication Date	Publisher	Publisher City	Pages	Barcode
1	机器学习	周志华	2016-01-01	清华大学出版社	北京市	368	9787302423287
2	数学之美 (第二版)	吴军	2014-11-01	人民邮电出版社	北京市	312	9787115373557

图 4.6 “Select Ones” 查询结果

如图 4.5 所示，所要查询的是出版日期在“2014-01-01”到“2018-01-01”且书名中包含“机器”或“数学”的出版物。查询结果如图 4.6 所示，通过对比出版信息，《机器学习》和《数学之美》这两本出版物符合查询条件，也可以验证条件查询的正确性。

如果继续进行查询，查询条件如图 4.7 所示，即所要查询的出版物需要页数在 200 到 500 页之间、ISBN（条形码）中含有“97875”或“97871”。查询结果如图 4.8 所示，符合条件的出版物共有 3 本，分别是《人民的名义》、《摆渡人》和《数学之美》。它们的页数均在 200 和 500 之前，且它们的 ISBN（条形码）要么以“97875”开头要么以“97871”开头，符合查询条件，同样可以验证条件查询的正确性。

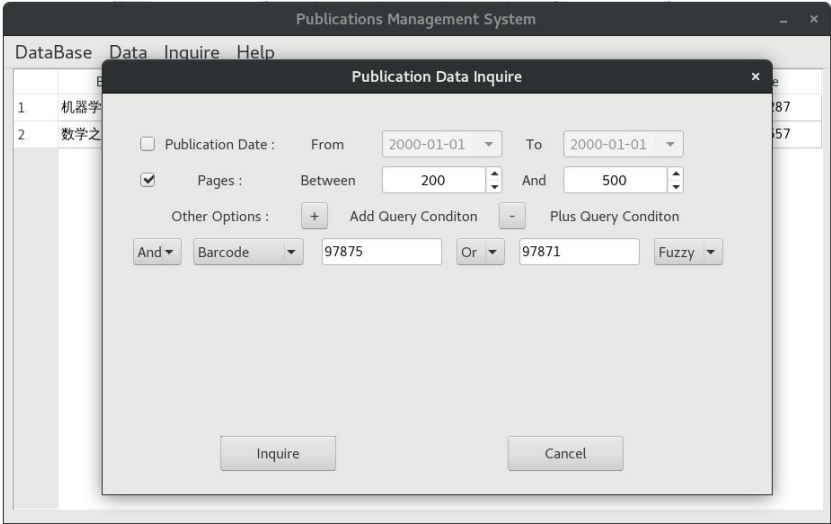


图 4.7 “Select Ones” 测试

The image shows the same "Publications Management System" window, but now displaying a table of search results. The table has the following data:

	Book Title	Author	Publication Date	Publisher	Publisher City	Pages	Barcode
1	人民的名义	周梅森	2017-01-01	北京十月文艺出版社	北京市	384	9787530216194
2	摆渡人	克莱儿·麦克福尔	2015-06-01	百花洲文艺出版社	南昌市	280	9787550013247
3	数学之美 (第二版)	吴军	2014-11-01	人民邮电出版社	北京市	312	9787115373557

图 4.8 “Select Ones” 测试结果

6. “Help” 菜单选项

“Inquire”主要提供程序和作者的相关信息，包括“About Program”和“About

Me”，其中“About Program”会弹出一个对话框说明本程序的相关信息，如图 4.9 所示；“About Me”会弹出一个对话框说明作者的相关信息，如图 4.10 所示。

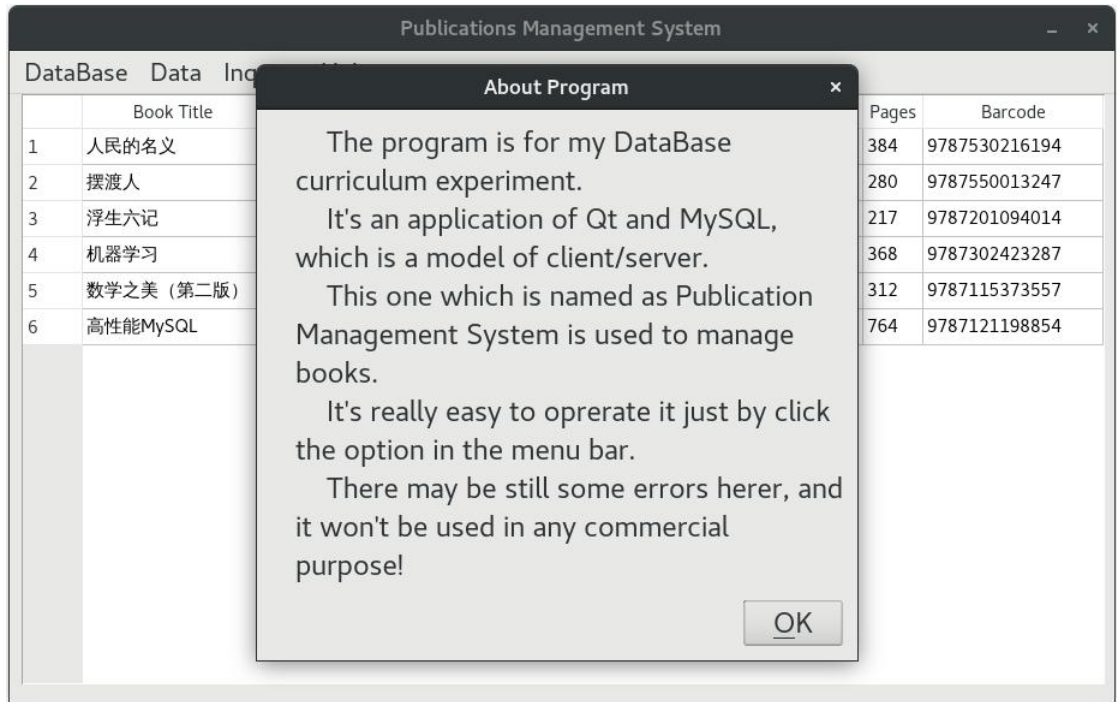


图 4.9 “About Program” 对话框

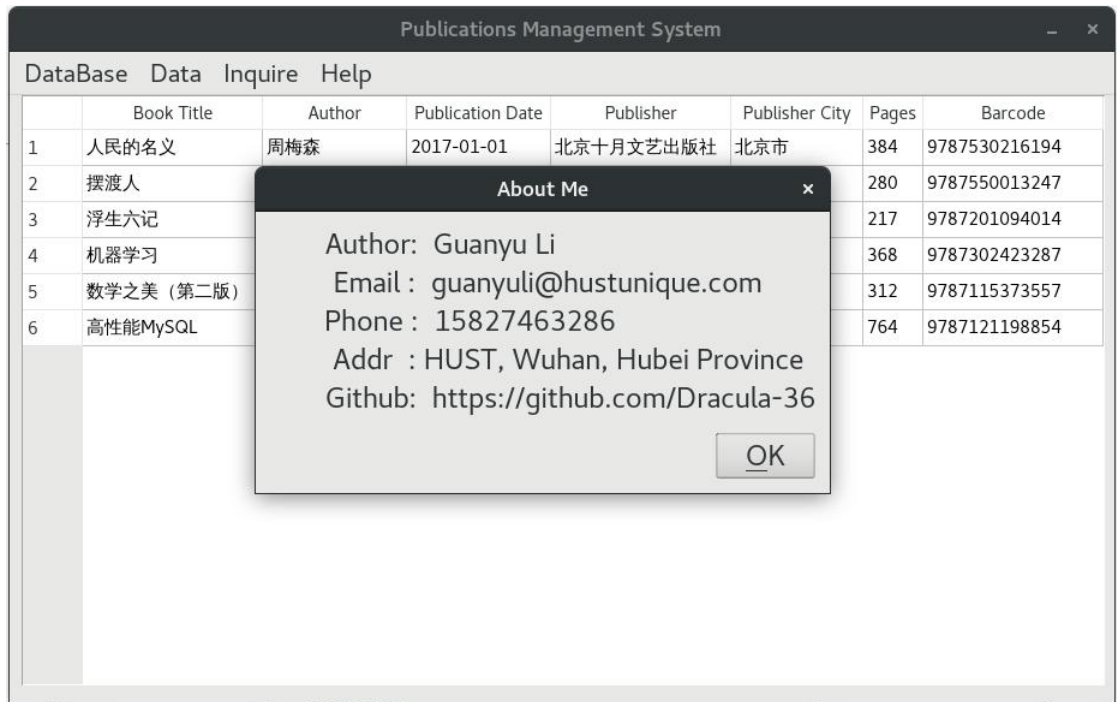


图 4.10 “About Me” 对话框

4.3 任务总结

本次实验任务的目的在于为最后的综合时间任务做准备，让我们提前熟悉下数据库和图形界面相结合的实际应用的编写方法，最重要的是确定使用的编程工

具、数据库管理系统即 DBMS 和图形界面工具。

由于之前有过写 Qt 的经验，因此本次实验任务选择了 C++ 作为编程语言，Qt 作为图形库，而 DBMS 最终使用了 MySQL。

在 DBMS 的选择上实际上遇到了一些问题。一开始，我选择的 DBMS 并非 MySQL 而是 SQLite，因为以前做项目的时候使用过 SQLite，因此电脑上已经配好了 SQLite 的环境，感觉直接使用 SQLite 可以节省很多搭建环境的时间。事实上也是如此，在我的电脑上只需添加一个头文件便可以使 Qt 直接连接 SQLite。但是，在写程序的过程中，我突然意识到 SQLite 是一种单机版本的数据库，即 SQLite 的客户端和服务端必须在一台主机上，不支持远程访问。所以，如果用 SQLite 写出来的 Qt 程序，只能是一种表面上的“C/S”模型，而并非真正的“C/S”模型。

所以，我决定使用 MySQL 代替 SQLite。但是当在 Qt 中将数据库类型由 SQLite 变成 MySQL 时，Qt 的 IDE 开始报错。查阅了网上相关资料后，发现是“libqsqlmysql.so”的动态链接库有问题（我使用的 Linux 系统），在使用“ld libqsqlmysql.so”命令后，可以通过输出找到有问题的动态链接库，然后进行对应修改即可。最终，成功将 Qt 和 MySQL 结合在了一起。

5 综合实践任务

5.1 系统设计目标

5.1.1 应用背景

随着图书馆中图书的日益增多，以及图书馆的规模不断变大，人为管理的难度越来越大，读者和图书馆管理员都迫切需要一个图书管理系统。例如，图书馆内馆藏书目的增加，导致图书种类随之增多，因此对图书查询提出了新的挑战；部分读者由于工作繁忙，常常抽不出时间将图书在指定期限前归还到图书馆，因此需要向其提供远程的图书续借功能并需要有对应的超期归还罚金制度。

5.1.2 总体目标

为了便于管理，需要设计一种图书管理系统来方便读者和管理员。借助该图书管理系统，读者可以很方便地查询书籍的相关信息或者续借书籍，管理员则可以很方便地管理用户、图书信息并执行借书和罚款等操作。管理员和读者都有权访问图书馆数据库，但他们拥有的功能是不同的，因此该管理系统需要提供登录机制并根据登录的账户提供不同的服务界面。

5.2 需求分析

5.2.1 功能需求

1. 对于图书馆管理员，图书管理系统应提供如下功能

(1) 借书：核实读者身份并检查是否存在下述情况：

- 该读者借书的数额超标；
- 该读者借过的书中有过期未还的书；
- 该读者曾因借书过期或者损坏图书被罚款而未交；
- 该读者所要借的图书不在书架上（被借走了）；

如不存在上述情况，则登记借书信息。

(2) 还书：检查所还图书是否损坏或过期，是则登记罚单信息并打印罚单，在交纳罚金前，不允许该读者继续借书。若图书损坏，注销该图书信息，否则进行还书登记。

(3) 罚款：根据罚单收取罚金，同时取消该读者的借书限制。

(4) 图书信息维护：新书上架、旧书下架及图书信息查询。

(5) 读者信息维护：录入、注销、修改及查询读者信息。

2. 对于普通读者，图书管理系统应提供如下功能

同时读者可以完成：

(1) 查询图书信息，可参考实验任务 3。

- (2) 查询自己的基本信息。
- (3) 查询自己借过的书籍。
- (4) 续借,注意只有在续借次数不为 0 的情况下读者才可以对借过的书进行续借操作。

5.2.2 性能需求

图书馆管理系统需要考虑到应用的具体环境,即在现实生活中,图书馆的所有数据库往往是运行在图书馆馆内的服务器中,而图书馆管理系统这一客户端则可以运行在任意网络的任意位置。换句话说,图书管理系统的客户端与其数据库往往不在一台主机上,应当为一种 C/S 模型。

考虑到这样的应用场景,图书管理系统应该支持多个用户同时使用客户端访问数据库。传统的 C/S 模型,数据的交互过程应该是客户端与服务器程序交互,服务器程序与数据库交互,但是考虑到本次实验任务的时间有限以及我们的能力有限,我在设计时忽略了服务器程序而使客户端直接与远程数据库交互。MySQL 本身是支持并发访问的,即多个数据库用户可同时访问数据库,因此为了方便设计,我将图书馆的用户(管理员和读者)与数据库的用户合二为一,即使用图书馆用户的账户和密码同时也是访问数据库的账户和密码。通过这样的方法,既能简化设计和编程,还可以保证并发访问而适用于实际的应用场景。

除此之外,对于数据库的设计有如下性能需求:

1. 数据精度性:在精度需求上,根据实际需要,数据在输入、输出及传输的过程中要满足各种精度的需求根据关键字精度的不同。如:查找可分为精确查找和泛型查找,精确查找可精确匹配与输入完全一致的查询结果,泛型查找,只要满足与输入的关键字相匹配的输入即输出,可供查找。
2. 时间特性:系统响应时间应在人的感觉和视觉范围内(<5 s),能够满足用户要求。
3. 可操作性:操作界面简单明了,易于操作,对格式和数据类型限制的数据,进行验证,包括客户端验证和服务器验证,并采用错误提醒机制,提示用户输入正确数据和正确的操作系统。
4. 安全保密性:只有合法用户才能登录使用系统,对每个用户都有权限设置。对登录名、密码、以及用户重要信息进行加密,保证账号信息安全。
5. 可维护性:系统采用了记录日志,用于记录用户的操作及故障信息,同时本系统采用的 C/S 模式,结构清晰,便于维护人员进行维护。

5.2.3 数据完整性需求

1. 实体完整性约束

对于用户(包括读者和管理员)这个实体而言,用户 ID(账户名)应为主

属性且不能为空；对于出版物（实际上是实体书的抽象，是其详细的出版信息）这个实体而言，出版物的 ISBN 应为主属性且不能为空；对于实体书这个实体而言，实体书在图书馆中的 ID 号应为主属性且不能为空。

2. 参照完整性约束

对于实体书而言，它涉及到一个实体的主属性，即出版物的 ISBN，这时实体书的外码；对于借阅记录而言，它涉及到两个实体的主属性，分别是读者的 ID 和实体书在图书馆中的 ID，它们是借阅记录的外码。

5.2.4 数据流图

经过分析，数据实际上是在读者和管理员之间传递，通过一些操作进行流动。例如读者向管理员发出了借书或者还书的请求，然后管理员通过客户端为读者借书或者还书。

整个系统的数据流图如图 5.1 所示。

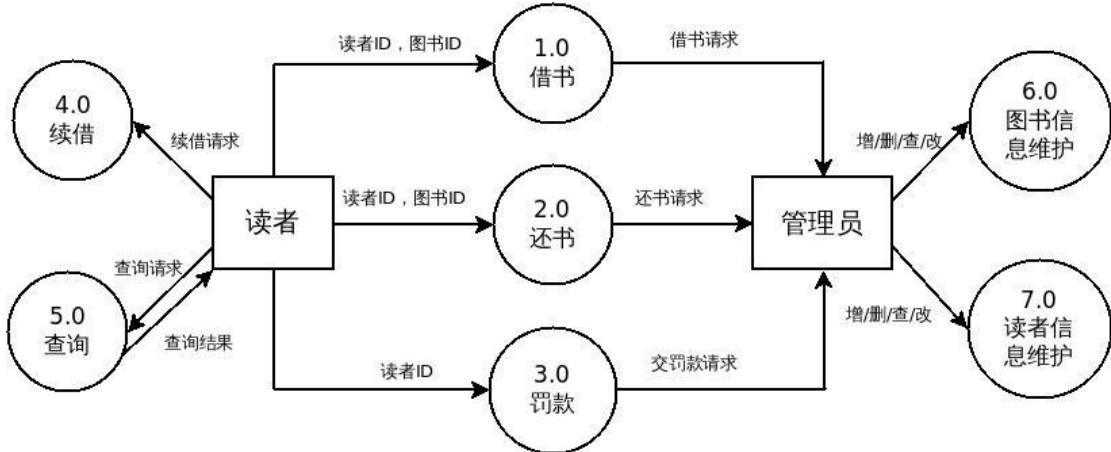


图 5.1 系统整体数据流图

借书的详细数据流图如图 5.2 所示。

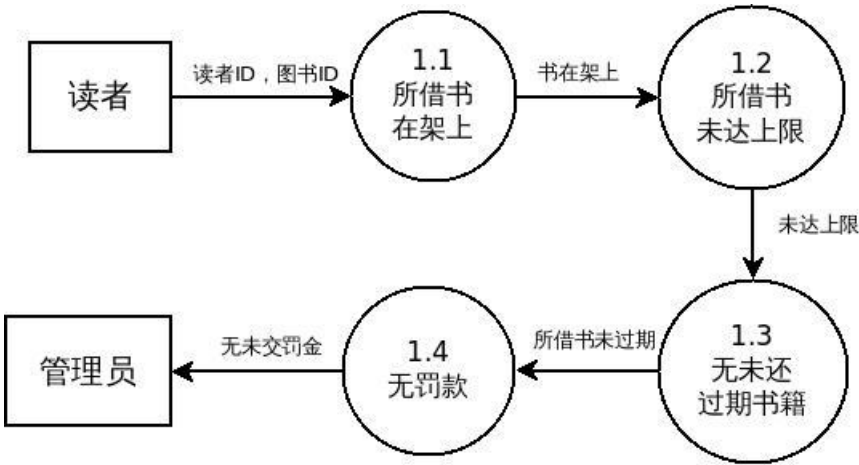


图 5.2 借书数据流图

还书的详细数据流图如图 5.3 所示。

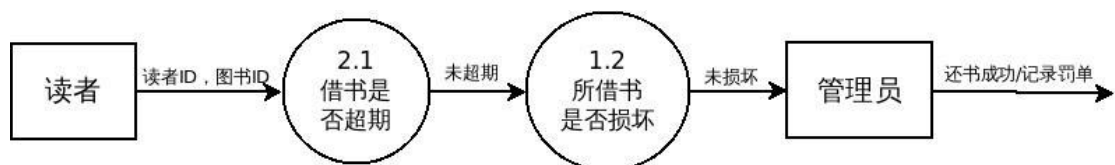


图 5.3 还书数据流图

罚款的详细数据流图如所示。



5.4 罚款数据流图

5.2.5 数据字典

图书管理系统的数据字典如表格 5.1 所示。

表格 5.1 图书管理系统数据字典

结构名	成员名	成员类型	说明
Admin	aid	varchar(10)	管理员 ID
	rid	varchar(16)	管理员密码
Reader	rid	varchar(10)	读者 ID
	pwd	varchar(16)	读者密码
	name	varchar(20)	读者姓名
	type	varchar(8)	读者种类
	phone	varchar(11)	读者手机
	email	varchar(36)	读者邮箱
	borrowed	int	读者已借阅书数量
	maxborrow	int	读者最大借阅数量
	fine	double	读者罚金
Publication	isbn	varchar(13)	出版物 ISBN
	title	varchar(36)	出版物书名
	author	varchar(20)	出版物作者
	pubdate	date	出版物出版日期
	publisher	varchar(20)	出版物出版社
	pubcity	varchar(10)	出版物出版社城市
	pages	int	出版物页数
	type	varchar(16)	出版物类别

	price	double	出版物价格
	intro	varchar(200)	出版物简介
	curnum	int	出版物馆内实体书数量
Book	bid	varchar(10)	图书馆图书 ID
	isbn	varchar(13)	图书馆图书 ISBN
	position	varchar(10)	图书馆图书馆内位置
	state	varchar(1)	图书馆图书状态
Borrow	rid	varchar(10)	读者 ID
	bid	varchar(10)	图书馆图书 ID
	borrowtime	datetime	借阅记录借书时间
	duetime	datetime	借阅记录还书期限
	renewtimes	int	借阅记录可续借次数

5.3 总体设计

5.3.1 系统 C/S 架构

本图书管理系统采用了 C/S 模式作为设计模式，如 5.2.2 所述，本系统的图形界面作为客户端，供图书管理员以及读者使用来实现以上功能需求；部署了 Mysql 的服务器作为服务端，没有专门的服务器程序。C/S 架构图如

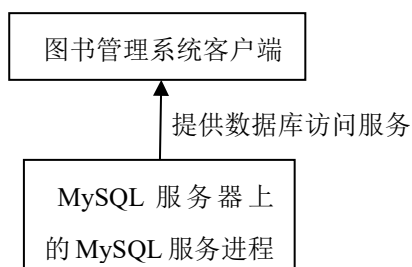


图 5.5 图书管理系统 C/S 架构图

编程语言方面，采用了 C++，因为图形界面采用了 Qt。以 Qt Creator 作为主要开发工具，选用 MySQL 作为远程数据库，配合功能强大的 SQL 查询语言实现建立关系数据库，访问数据库，对数据库的更新，较好地实现了预定的需求功能。

5.3.2 系统功能模块

本图书管理系统根据使用者，可以划分为两大功能模块，分别为图书管理员功能模块与读者功能模块，具体提供哪个功能模块需要根据用户登录时使用的账户类别进行区分。系统功能模块图如图 5.6 所示。

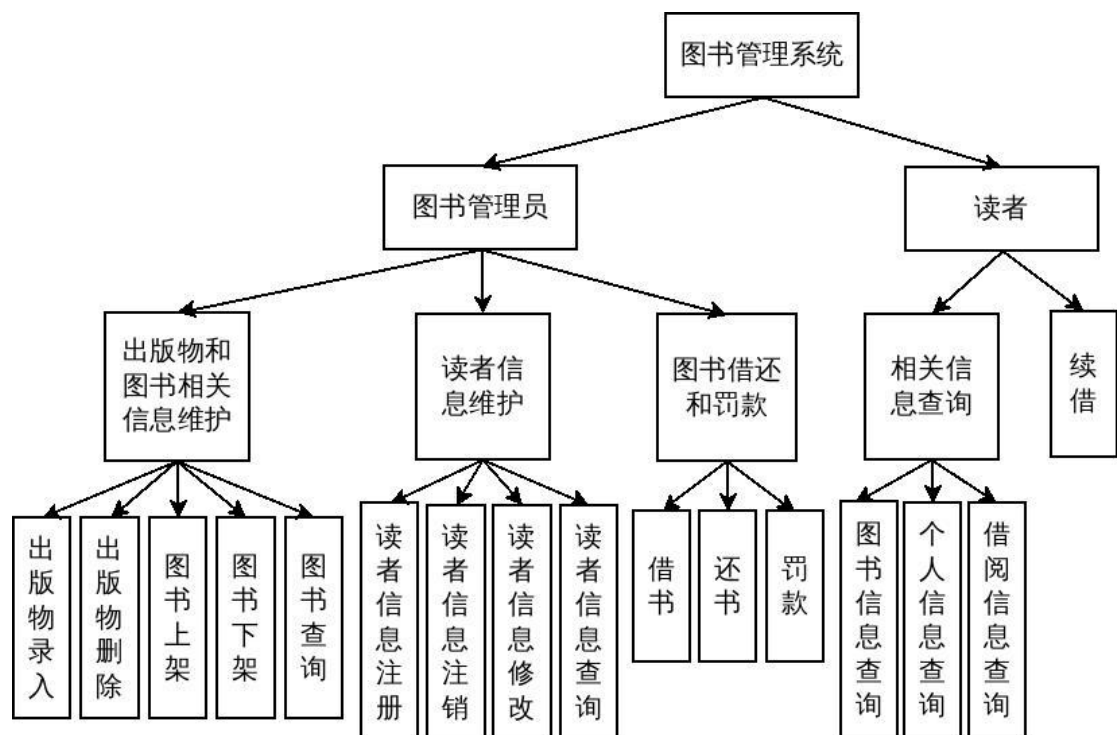


图 5.6 系统功能模块图

如图 5.6 所示，从图书管理员的角度考虑，本系统提供三大功能模块——出版物和图书相关信息维护、读者信息维护和图书借还与罚款；从读者的角度考虑，本系统提供两大功能模块——相关信息查询和续借。

对于图书管理员，出版物和图书相关信息维护功能模块又包括——出版物信息录入、出版物信息删除、图书上架、图书下架和图书查询；读者信息维护模块又包括——读者信息注册、读者信息注销、读者信息修改和读者信息查询；图书借还与罚款模块又包括——借书、还书和罚款。

对于读者，相关信息查询模块又包括——图书信息查询、个人信息查询和借阅信息查询。

再次强调，本系统一开始的登陆界面有两种登陆方式，一种是图书管理员登录，一种是图书借阅者登录。前者登录后，系统会显示对应图书管理员功能模块的图形界面；后者登录后，系统会显示对应读者功能模块的图形界面。

5.4 数据库设计

5.4.1 ER 图设计及其说明

本图书管理系统涉及到的实体表如表格 5.2 所示。

表格 5.2 本图书管理系统实体表

实 体	描 述
图书管理员	ID、密码
读者	ID、密码、姓名、类别、手机、邮箱、已借阅书数量、最大借阅数量、罚金
出版物	ISBN、书名、作者、出版日期、出版社、出版社城市、页数、类别、价格、简介、馆内实体书数量
图书（实体书）	ID、ISBN、馆内位置、状态

根据实体表和前述分析，作出本图书管理系统的 ER 图，如图 5.7 所示。

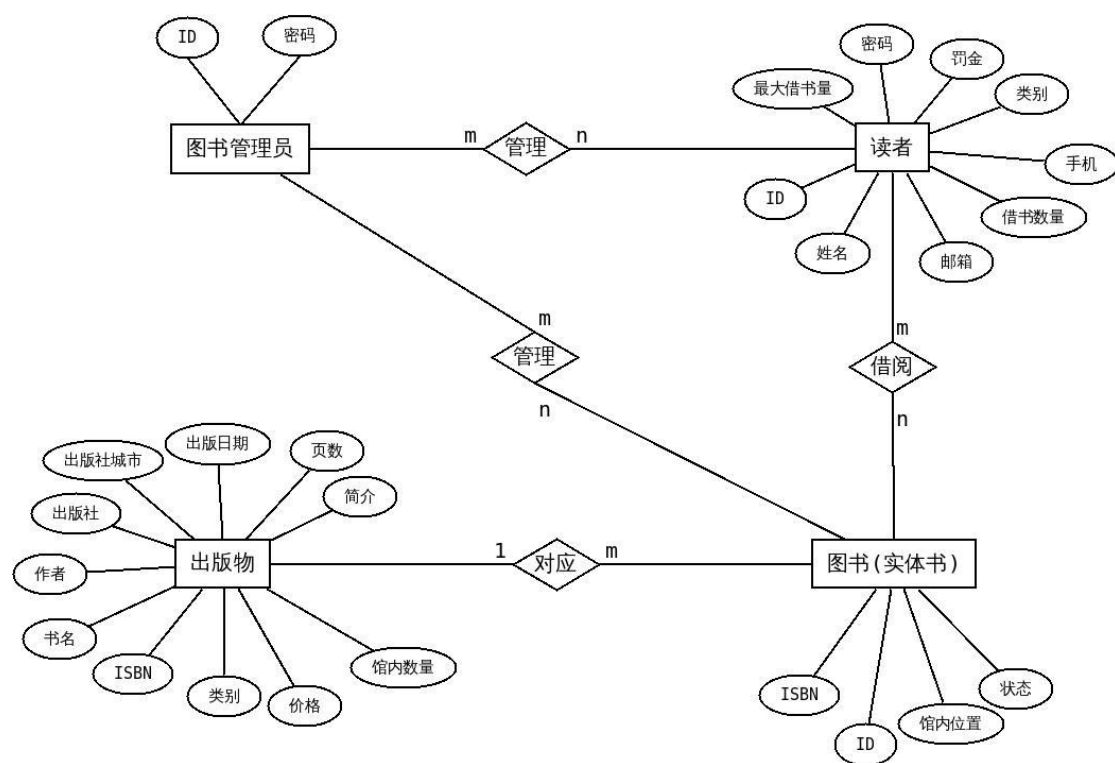


图 5.7 图书管理系统 ER 图

注意到，在实体表表格 5.2 和 ER 图图 5.7 中，均没有出现罚单这个实体。在本系统设计时，我认为罚单这个实体可有可无，因为读者自身已经具有“罚金”这个属性了，其代表当前读者所需交的罚款和。所以，没再设计实体罚单和对应的关系。

除此之外，本图书管理系统对出版物和实体书进行了区分：认为出版物是一种抽象的书籍信息，不可借阅；实体书才是真实存在图书馆书架上的图书，可以被借阅。

5.4.2 数据库逻辑结构设计

1. 根据 ER 图, 不难得到对应的数据表, 共 5 张表:

(1) 管理员表 Admin, 其字段说明见表格 5.3。

表格 5.3 Admin 表字段说明

字段名	字段类型	说明
aid	varchar(10)	管理员 ID
rid	varchar(16)	管理员密码

建表的 SQL 语句为

```
create table Admin(aid varchar(10), pwd varchar(16));
```

(2) 读者表 Reader, 其字段说明见表格 5.4。

表格 5.4 Reader 表字段说明

字段名	字段类型	说明
rid	varchar(10)	读者 ID
pwd	varchar(16)	读者密码
name	varchar(20)	读者姓名
type	varchar(8)	读者种类
phone	varchar(11)	读者手机
email	varchar(36)	读者邮箱
borrowed	int	读者已借阅书数量
maxborrow	int	读者最大借阅数量
fine	double	读者罚金

建表的 SQL 语句为

```
create table Reader(rid varchar(10), pwd varchar(16), name varchar(20),  
                    type varchar(8), phone varchar(11), email varchar(36),  
                    borrowed int, maxborrow int, fine double);
```

(3) 出版物表 Publication, 其字段说明见表格 5.5。

表格 5.5 Publication 表字段说明

字段名	字段类型	说明
isbn	varchar(13)	出版物 ISBN
title	varchar(36)	出版物书名
author	varchar(20)	出版物作者
pubdate	date	出版物出版日期
publisher	varchar(20)	出版物出版社
pubcity	varchar(10)	出版物出版社城市
pages	int	出版物页数

type	varchar(16)	出版物类别
price	double	出版物价格
intro	varchar(200)	出版物简介
curnum	int	出版物馆内实体书数量

建表的 SQL 语句为

```
create table Publication(isbn varchar(13), title varchar(36),
    author varchar(20), pubdate date, publisher varchar(20),
    pubcity varchar(10), pages int, type varchar(16),
    price double, intro varchar(200), curnum int);
```

(4) 图书（实体书）表 Book，其字段说明见表格 5.6。

表格 5.6 Book 表字段说明

字段名	字段类型	说明
bid	varchar(10)	图书馆图书 ID
isbn	varchar(13)	图书馆图书 ISBN
position	varchar(10)	图书馆图书馆内位置
state	varchar(1)	图书馆图书状态

建表的 SQL 语句为

```
create table Book(bid varchar(10), isbn varchar(13), position varchar(10),
    state varchar(1));
```

(5) 借阅记录表 Borrow，其字段说明见表格 5.7。

表格 5.7 Borrow 表字段说明

字段名	字段类型	说明
rid	varchar(10)	读者 ID
bid	varchar(10)	图书馆图书 ID
borrowtime	datetime	借阅记录借书时间
duetime	datetime	借阅记录还书期限
renewtimes	int	借阅记录可续借次数

建表的 SQL 语句为

```
create table Borrow(rid varchar(10), bid varchar(10),
    borrowtime datetime, duetime datetime, renewtimes int);
```

2. 关系表之间的完整性约束

(1) 注意到，Admin、Reader、Publication 和 Book 都有明显的主属性，因此将其作为主码是自然而然的，它们都应具有主码完整性约束。

对于管理员表 Admin，主码为 aid，利用 SQL 语句添加对应的主码约束如下：

```
alter table Admin add constraint PK_Admin primary key Admin(aid);
```

对于读者表 Reader，主码为 rid，利用 SQL 语句添加对应的主码约束如下：

```
alter table Reader add constraint PK_Reader primary key Reader(rid);
```

对于出版物表 Publication，主码为 isbn，添加对应的主码约束的语句如下：

```
alter table Publication add constraint  
PK_Publication primary key Publication(isbn);
```

对于图书表 Book，主码为 bid，添加对应的主码约束的 SQL 语句如下：

```
alter table Book add constraint PK_Book primary key Book(bid);
```

(2) 前面提到过，出版物是图书（实体书）的抽象信息，因此出版物表 Publication 跟图书表 Book 之间存在参照完整性约束；除此之外，借阅信息表 Borrow 与读者表 Reader 和图书表 Book 之间也存在参照完整性约束。

对于出版物表 Publication，外码为 isbn，利用 SQL 语句添加对应的外码约束如下：

```
alter table Book add constraint  
FK_Book foreign key Book(isbn) references Publication(isbn);
```

对于借阅信息表 Borrow，外码为 rid 和 bid，利用 SQL 语句添加对应的外码约束如下：

```
alter table Borrow add constraint  
FK_Borrow_Reader foreign key Borrow(rid) references Reader(rid);  
alter table Borrow add constraint  
FK_Borrow_Book foreign key Borrow(bid) references Book(bid);
```

5.5 详细设计与实现

1. 图书管理员功能模块

(1) 借书

借书功能是根据读者的图书证号来判断读者是否存在以下 4 种情况：一，所借的书籍是否真的在图书馆书架上；二，借书数额是否超标；三，以前借的书是否超期未还；四，是否有罚单未缴纳。如存在以上任意一种情况则弹出相应提示并拒绝借书，如果不存在上述情况则允许借书，借书成功需要进行的操作是让相应书的库存减 1，并登记借阅信息入借阅表中。具体流程图如图 5.8 所示。

借书操作对话框的设计如图 5.9 所示。在进行借书操作前，图书管理员必须输入图书的 ID 和读者的 ID 并进行 Verify 验证。Verify 主要验证的是两个 ID 是否合法。整个借书的核心技术均在“Borrow”这个按钮的回调函数，其流程为：首先从数据库中读取图书 ID 为输入图书 ID 的图书信息，判断其状态是否为在架上，如果不在表明已经被借走无法再借阅，则违反第一类规则；然后读取数据库中关于 ID 为输入的读者 ID 的读者的最大可借数量和借阅数量，比较两数是否相等，若相等则违反第二类规则；再用数据库语言读取当前时间，查询该读者借阅信息中的还书时间 DueTime，如果现在时间晚于还书时间，则判断为超期，标记第三类违反规则；最后查询读者信息中的罚金字段是否为 0，如果不是则表明有罚金未支付，违反规则 3。最后根据违反规则打印提示信息，如果没有违反规则则允许借书，根据读者 ID 和书 ID 记录借阅信息，并使相应图书库存减 1。

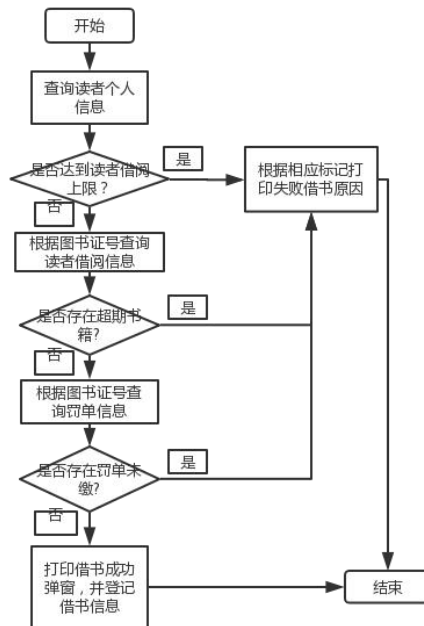


图 5.8 管理员为读者借书流程图

图 5.9 借书对话框

(2) 还书

还书操作对话框的设计如图 5.10 所示。同样地，在进行借书操作前，图书管理员必须输入图书的 ID 和读者的 ID 并进行 Verify 验证，Verify 主要验证的是两个 ID 是否合法，并且同时查询对应图书和对应读者的相关信息。还书时主要有 3 种情况：一，读者按期归还，所谓按期是指的借阅记录中的还书期限早于当前时间，这种情况下直接还书即可，删除 Borrow 表中对应的借阅信息；二，读者超期归还，根据还书期限与当前时间的差值计算超期天数，然后计算对应的罚金，这种情况不仅要删除 Borrow 表中对应的借阅信息还要更新 Reader 表中读者的罚金字段；三，读者损坏了图书，这种情况下，读者须原价赔偿图书。

Reader ID : Verify

Name :

User Type:

Current Fines :

Book ID : Verify

Book Title :

Book ISBN :

Book Price :

Book State :

Overdue Span: Day Hour

Book Damage : ☐ Intact ☐ Damaged

Fines to be paid:

图 5.10 还书对话框

(3) 罚款

罚款的逻辑相对简单,唯一需要注意的地方是要保证用户所交的罚款不能超过其罚金。

(4) 其余数据维护和数据查询操作

实现方法与实验任务 3 中类似,因此虽然麻烦但大都是重复性的东西,这里限于篇幅不再详细阐述了。

2. 读者功能模块

(1) 续借

续借对话框如图 5.11 所示。仍然,最核心的功能在于“Renew”按钮的回调函数。当按下“Renew”时,要判断“Renewable”(可续借次数)是否大于 0。如果可续借次数为 0,则表示读者不能续借,要弹出对话框进行警告。

Book ID :

Title :

Author :

Publisher :

Borrow At :

Overdue At :

Renewable :

After you renew it ...

Overdue At :

Renewable :

图 5.11 续借对话框

- (2) 其余数据查询操作
重复性内容，不详细陈述。

5.6 系统测试

5.6.1 测试数据

测试数据仍然采用实验任务 3 中使用的 6 个出版物，但是每个出版物都有若干个实体书。读者共有两人，分别是“Reader1”和“Reader2”。

5.6.2 测试过程

1. 登录功能测试

登录界面如图 5.12 所示，有两个登录选项——读者 Reader 和图书管理员 Administer。若远程数据库出现故障或者因为网络问题无法访问远程服务器，都会造成登录超时，报错如图 5.13 所示。

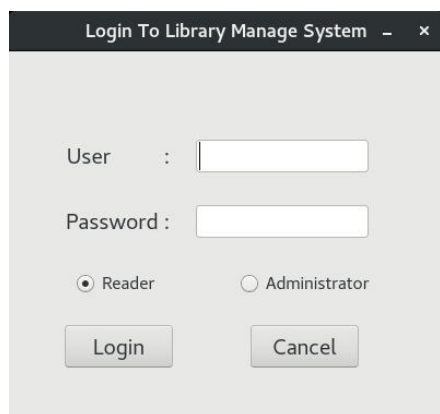


图 5.12 登录界面

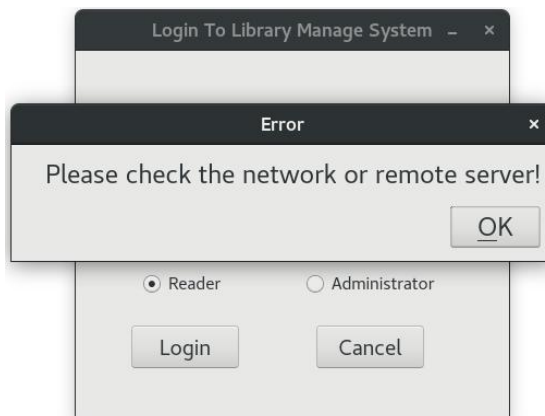


图 5.13 登录超时

如果登录时 ID 和密码输入错误，或者以错误身份登录，同样也会报错给用户以提示信息，具体的报错分别如图 5.14 的左图和右图所示。

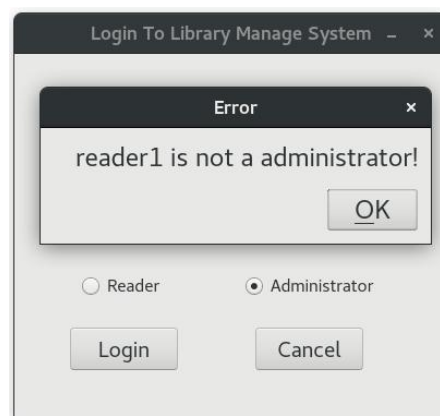
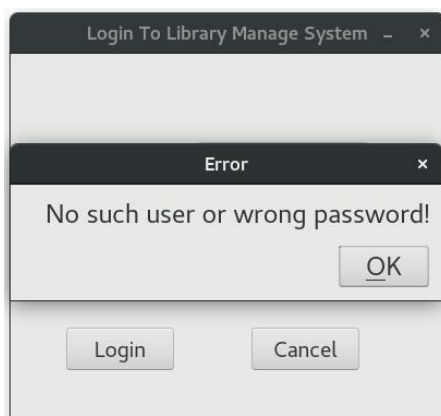


图 5.14 错误登录信息造成的登录报错

综上，本图书管理系统提供了功能强大的登录界面和功能，可以针对不同的登录错误情况给出不同的报错信息，保证了系统安全性的同时也方便了用户使用。

2. 读者功能模块测试

若以读者 Reader 身份成功登录，本系统会显示基于读者功能模块的图形界面，如图 5.15 所示。可见，菜单栏包含 3 个选项——Inquire、Renew、Help。



图 5.15 Reader 功能模块初始化界面

(1) 个人信息查询

查询结果如图 5.16 所示，注意到 reader 已经借阅了两本书籍，达到上限。

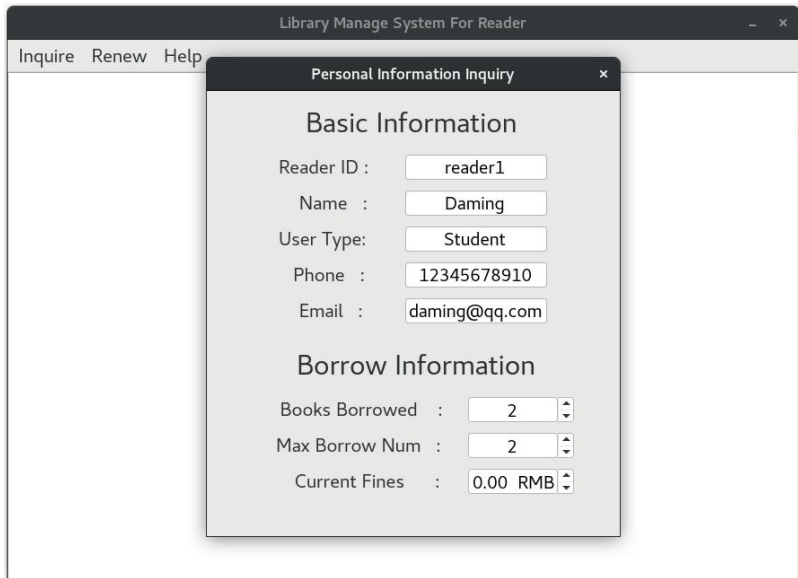


图 5.16 个人信息查询测试结果

(2) 借阅书籍查询

查询结果如图 5.17 所示，可见 reader1 借阅的两本书为《人民的名义》和《浮生六记》，双击《人民的名义》所在表项，可弹出对话框展示图书详细信息。在图 5.17 右图中，不仅可以看到《人民的名义》的出版物信息，还可以获知其在图书馆内的所有实体书的情况，极其详细和方便。

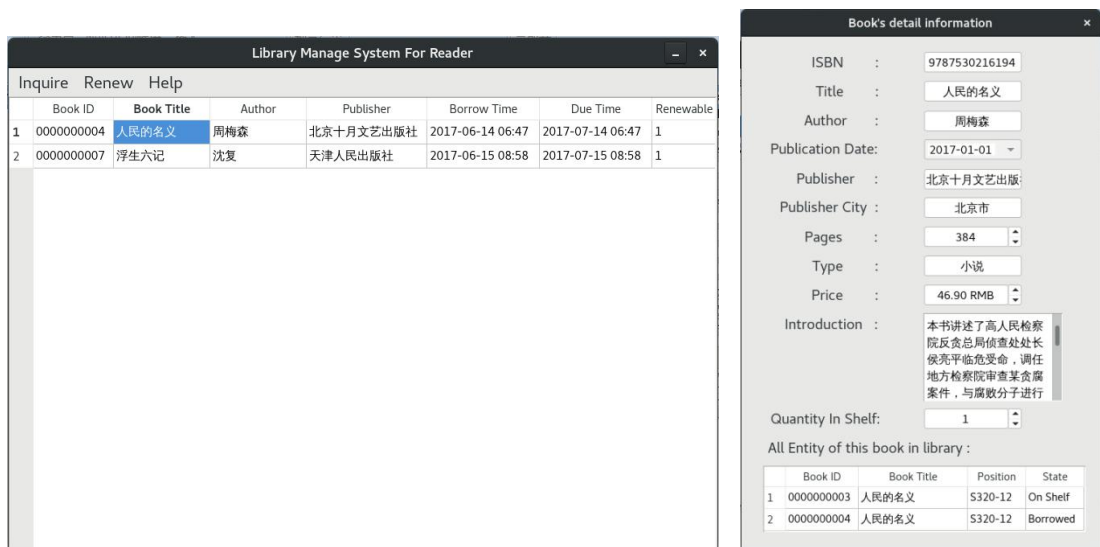


图 5.17 借阅查询测试结果

(3) 图书信息查询

查询对话框如图 5.18 所示，与实验任务 3 基本相同，只是添加了部分查询条件，具体的查询过程和结果不再测试。

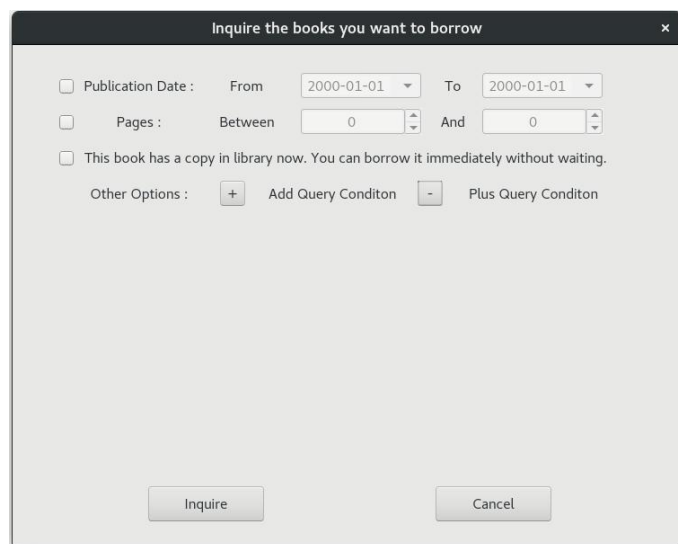


图 5.18 图书查询对话框

(4) 续借

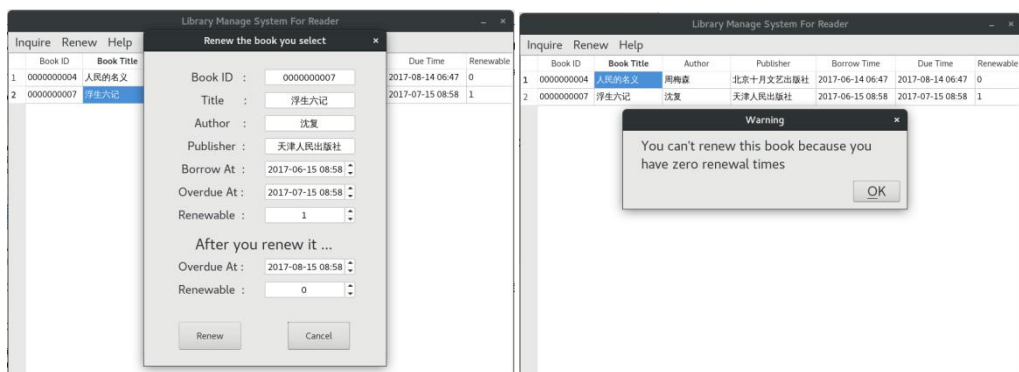


图 5.19 续借功能测试

如图 5.19 所示，当借阅记录的续借次数不为 0 时，读者可以选中所要续借的图书表项，然后选择菜单栏“Renew”下的“Renew One”进行续借。但若续借次数为 0，系统会弹出对话框警告不能续借。

3. 图书管理员功能模块测试

(1) 借书

前面提到过，读者 reader1 已经达到了最大借阅数量，故再次借书时系统会报错提示读者 reader1 无法继续借阅书籍，如图 5.20 所示。

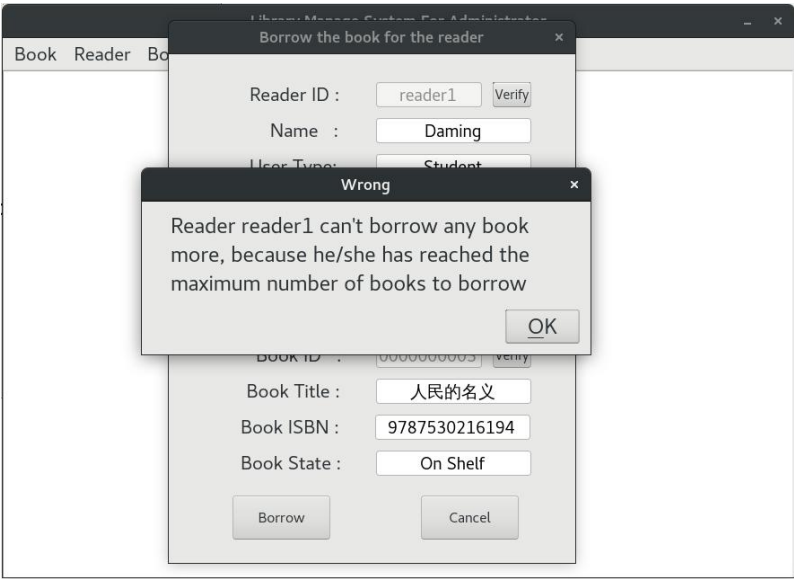


图 5.20 达到上限借阅书籍测试

(2) 还书

为了达到测试效果，假设 reader1 损害了其借阅的《人民的名义》这本书，归还后，再次查询《人民的名义》在馆中的图书信息，对比图 5.17 的右图和图 5.21 的右图，不难发现图书 ID 为 0000000004 的那本书的记录已经被销毁，《人民的名义》也只剩下了另一本图书。

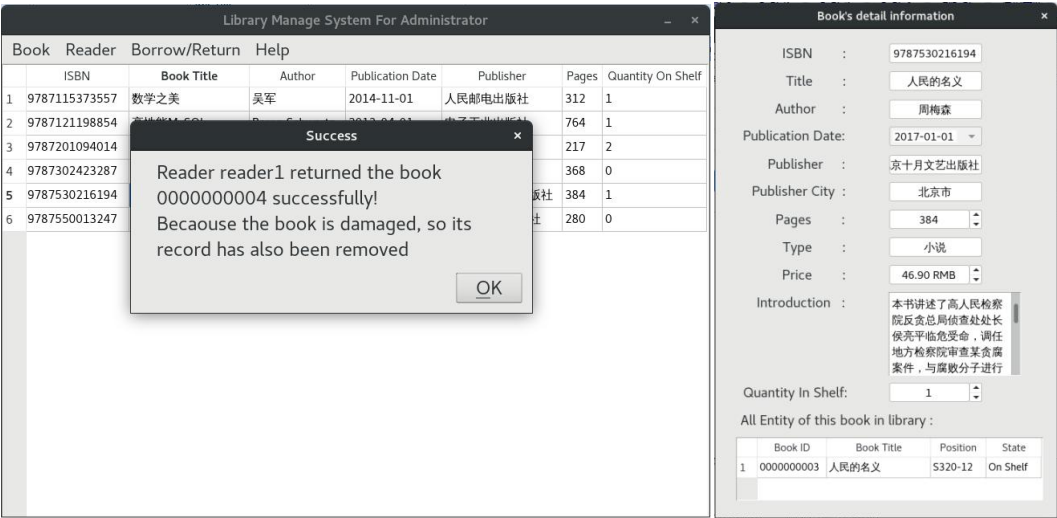


图 5.21 损害图书情况下还书测试

(3) 罚款

在还书测试中，由于 reader1 损坏了图书，因此其罚金是损害图书的定价——46.90RMB。在进行罚款测试时，分两次进行，第一次支付 40RMB，第二次支付 6.90RMB，两次支付罚款的情况分别如图 5.22 和图 5.23 所示。由图 5.22 可以看出，“Left Fines”会随“Paid Fines”的变化而变化；由图 5.23 可知，当用户支付的罚款比其需要交的多时，系统会报错提示。

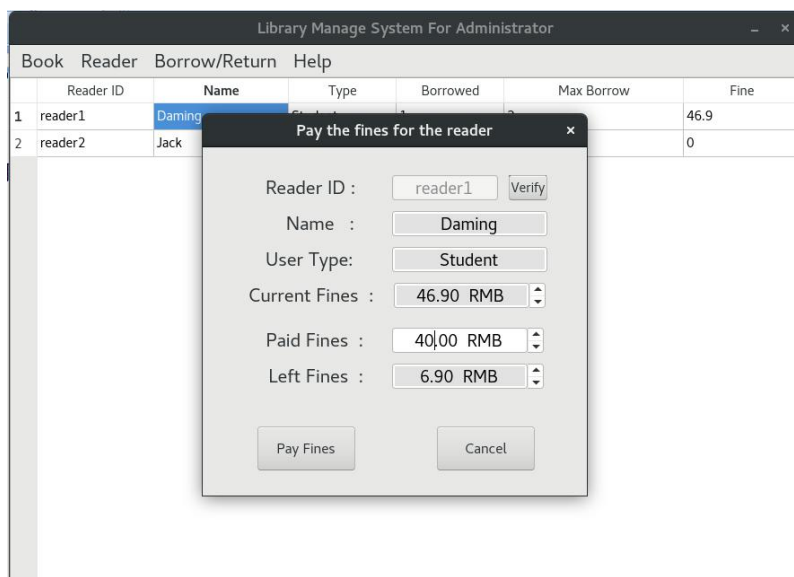


图 5.22 罚款测试第一次

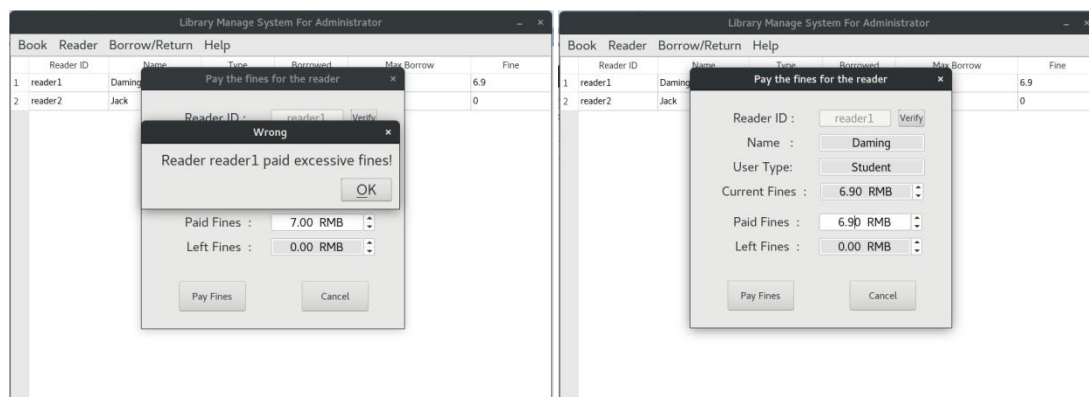


图 5.23 罚款测试第二次

(4) 图书上架

出版物的录入和删除在实验任务 3 中已经实现，故不再测试。

图书上架需要通过 ISBN 来指明上架图书的出版信息，该 ISBN 的获取有两种方式：第一种，如果当前窗口主体显示的不是图书信息，则要上架一本新书时需要手动输入 ISBN 并进行 Verif (Verify 在本系统中非常常见，可以保证某些主属性的输入数据的有效性，如果不 Verify 是不能点击 Confirm 按键的)，然后再上架，该方式的操作过程如图 5.24 所示；第二种，如果当前窗口主体显示的是图书信息（图书查询的结果），则鼠标选中一个图书表项时，再选择上架新书，

在弹出的对话框中，会自动获取选中图书的 ISBN 并获取相关出版信息，该方式的操作过程如图 5.25 所示。

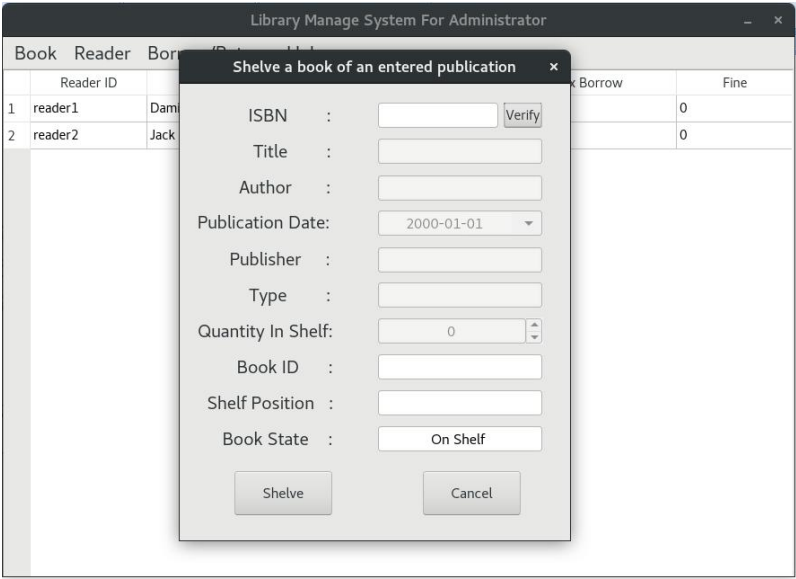


图 5.24 上架新书获取 ISBN 方式 1

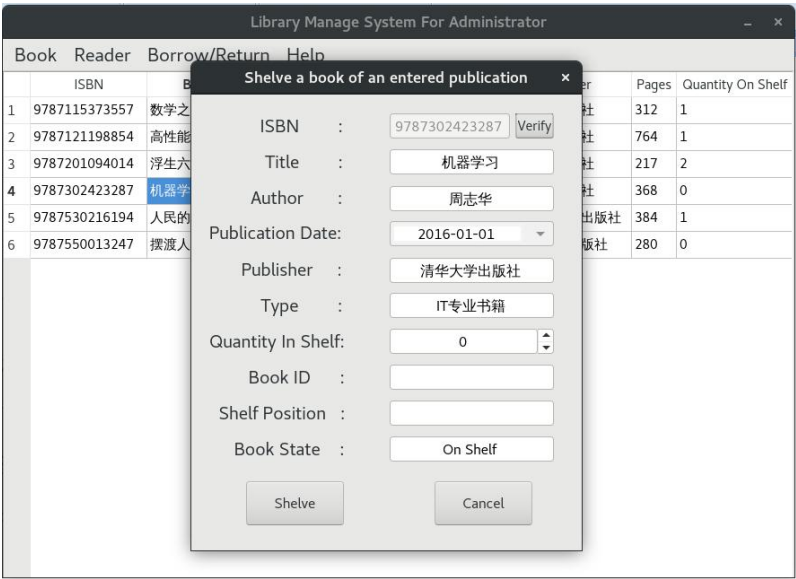


图 5.25 上架新书获取 ISBN 方式 2

(5) 图书下架

前面已经提到过 Verify 按钮在本系统中的广泛应用，在图书下架时仍然用到了。如果不对输入的 ISBN 进行 Verify，就会造成系统报错，而无法进行 Verify，如图 5.26 所示，通过这种方法可以保证输入的 ISBN 是有效的 ISBN，本质上是一种参照完整性约束。

考虑另外一种可能，如果一本图书已经被读者借阅了而不再馆中，这样的图书是不能下架的，如图 5.27 所示。如果真的要下架这样的图书，要么等读者还回来这本书，要么替读者以损害状态归还这本书。

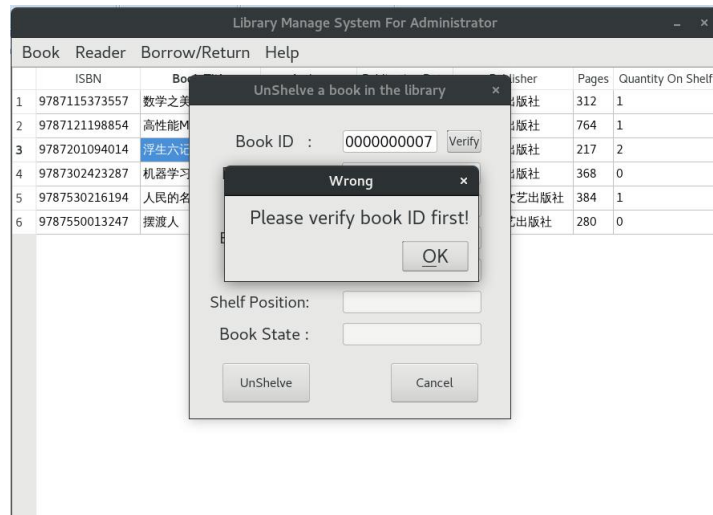


图 5.26 下架图书未 Verify ISBN

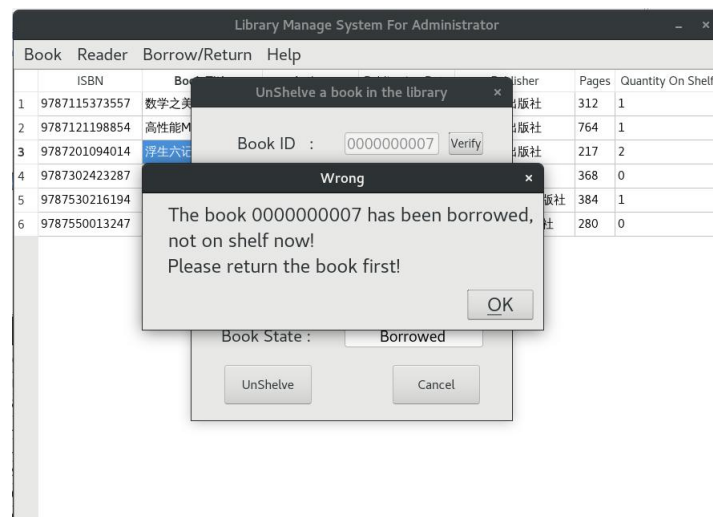


图 5.27 下架一本正在被借阅的图书

(6) 读者注册

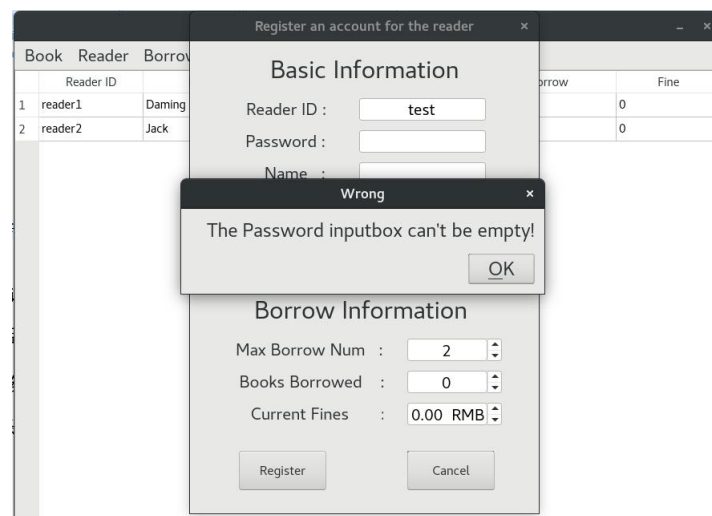


图 5.28 注册读者信息时密码不能为空

在注册一个新的读者时，读者 ID 不能为空，密码也不能为空（如果密码为空怎么登录），因此在注册时需对用户输入的信息进行检查。如果一些关键信息为空，那么需要报错提示，如图 5.28 所示。但对于手机、邮箱这样的信息则可以空，因为其不影响数据库的安全性。

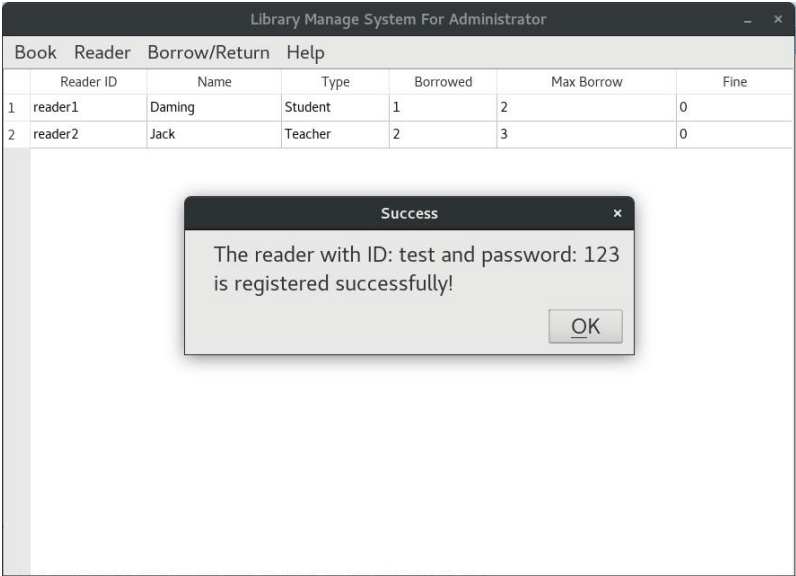


图 5.29 注册用户成功

注册时将密码设为 123，同时将姓名设置为 test，进行注册，如图 5.29，注册成功。然后使用新注册的用户 test 重新登录本图书管理系统，并查询个人信息，其结果如图 5.30 所示。

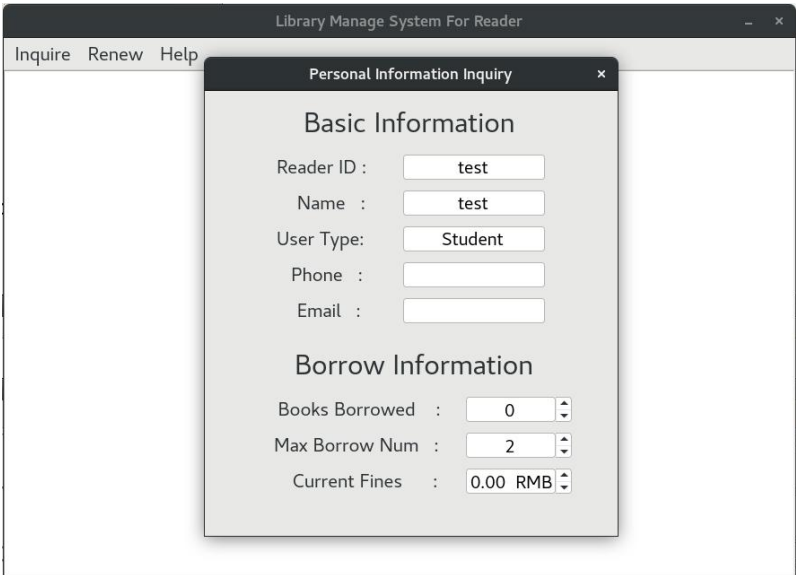


图 5.30 使用新用户登录并查询个人信息

(7) 读者注销

注销读者时需要考虑两个问题：第一，如果读者有借阅的图书，可否注销读者；第二，如果读者有未支付的罚金，可否注销读者。

在本图书管理系统中，针对第一个问题，是不能注销这样的读者的，如图

5.31 所示，因为如果强制注销读者会造成其借阅图书的信息丢失，从而有可能造成数据库各表中的数据不一致。

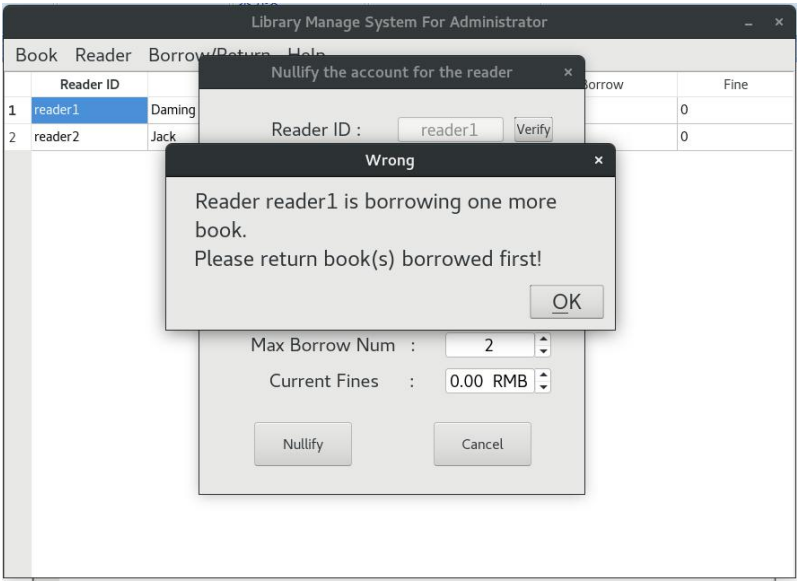


图 5.31 注销有书未还的读者

对于有书未还的用户，如果真的必须注销也是有办法的，那就是帮该读者以损害图书情况还书，这样该读者不再有借书记录，而是有未支付的罚金，这就变成了第二个问题。

针对第二个问题，本系统会弹出一个警告对话框，如图 5.32 所示，如果仍要注销读者则仍旧可以注销，如图 5.33 所示。因为这样只是经济损失，不会破坏数据库的一致性。

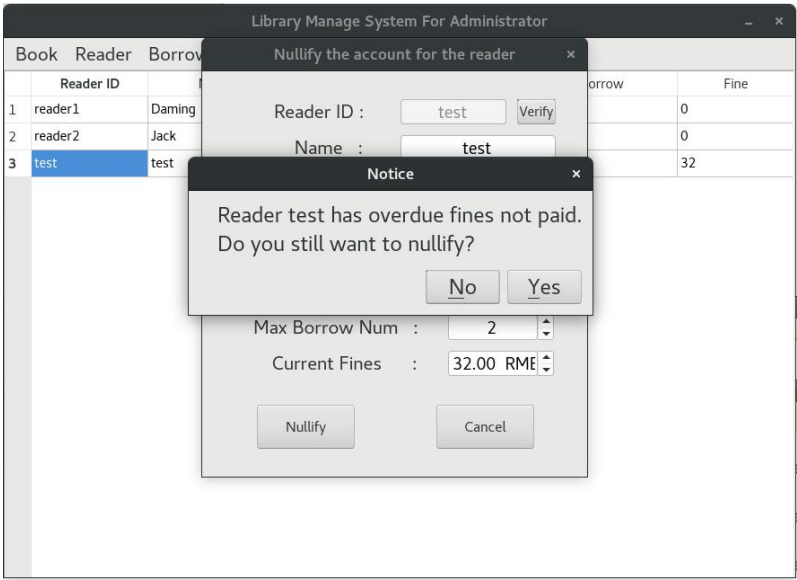


图 5.32 注销有罚金未交的读者

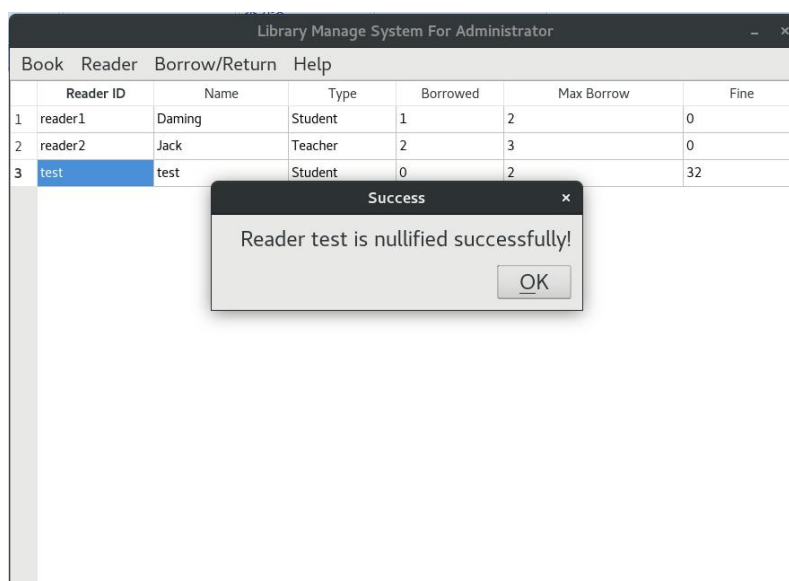


图 5.33 强制注销有罚金未交的读者

5.6.3 测试结果分析

通过测试可以看到，本图书管理系统的可用性非常强，有多种手段保证数据有效性和系统安全性。例如 Verif 按钮的使用，关键信息的自动填充，以及测试过程中各种特殊情形下本系统的处理方法等。这些，均可体现出我在设计系统时的周全考虑。

5.7 系统设计与实现总结

应用系统设计应按照以下步骤进行：

1. 需求分析，全面思考业务流程，编好数据字典并画好 ER 图等；
2. 根据 ER 图设计数据库，主要是各表的逻辑结构，有时还需考虑物理结构；
3. 选择合适的开发工具、图形界面库以及 DBMS 进行开发；
4. 设计一个较完整的界面，可以暂时不加入任何功能；
5. 逐步添加各项功能，添加时要充分考虑到各种情形，不能图快，应放慢节奏全面考虑；
6. 录入多种情况的数据，尤其是极端情况下的数据，对各种情况进行完整测试，并测试系统的鲁棒性；
7. 美化图形化界面，给用户更好的使用体验。

6 课程总结

6.1 主要工作

1. 完成了实验任务一，熟悉并掌握了使用 SQL 语言进行表的创建、增加、删除、修改、查询等操作；
2. 完成了实验任务二，学会了使用 DBMS 进行权限控制及完整性控制，并且自学和了解了存储过程及存储函数的概念和用法；
3. 完成了实验任务三，了解了基本的数据库应用编程方法与技术，实现了一个出版物管理系统，并且系统可以提供界面友好的出版物信息维护功能和查询功能，除此之外，我所设计的系统还提供了类似“保存操作”和“撤销”操作的功能，极大地方便了用户的使用；
4. 完成了实验任务四（综合实践任务），实现了一个基于 C/S 结构、基于 Qt 和 MySQL 实现的图书管理系统，掌握了数据库应用设计的整个流程，从需求分析，到数据库设计，再到最后的程序设计，除此之外，我设计的系统对各种可能情况都有所考虑，充分利用弹出对话框帮助用户的使用，实用性和安全性较好。

6.2 心得体会

这次数据库课设再次证明了实践和理论是相辅相成的，无论你觉得自己在课堂上学习的有多好，当真的动手实践时总会又觉得自己理解得原来并不是那么深刻。举个例子，当时课堂上在讲几种隔离级时，我觉得自己基本都听懂了，但当做第一次实验的选做题时又发现，它跟我想象的好像不太一样，例如 `Serializable` 可串行化，我一直只记得它是最高级别，但当做实验时才发现自己并没有理解其为什么隔离级最高。类似的例子还有很多，很多课堂上的东西到了实验时总会有新的收获。

但在这次实验里自己也有做得不好的地方，例如实验报告的问题，都是最后的突击而不是每次做完一个就写一部分，所以导致最后时间非常赶。不过，我还是有个小的建议，那就是把数据库的实验或者课设放到暑假里去做，因为大三下学期的事情确实有点多，我就一直在准备大四实习的申请、面试，准备保研夏令营的申请、面试，所以做起实验来总有种心有余而力不足的感觉。

不过，总的来说，这次课设还是收获颇丰。

附录