

书店书刊出租和零售管理系统

结构化需求分析

姓名：樊昕昊

学号：201800800504

指导教师：姜秀娥

学院：机电与信息工程学院

专业班级： 2018级软件工程01班

**2020年4月12日**

目录

[**一、** **选题说明及需求介绍** 3](#_Toc38520756)

[**1.** **背景** 3](#_Toc38520757)

[**2.** **需求介绍** 3](#_Toc38520758)

[**二、** **系统功能模块划分** 4](#_Toc38520759)

[1. 顶层数据流图 5](#_Toc38520760)

[2. 一层数据流图 6](#_Toc38520761)

[3. 二层数据流图 7](#_Toc38520762)

[3.1 库存管理 7](#_Toc38520763)

[3.2 进货管理 7](#_Toc38520764)

[3.3 租借管理 8](#_Toc38520765)

[3.4 销售管理 8](#_Toc38520766)

[3.5 财务统计 9](#_Toc38520767)

[3.6 会员管理 9](#_Toc38520768)

[4. 数据字典 10](#_Toc38520769)

[4.1 数据结构描述 10](#_Toc38520770)

[4.2 数据存储描述 10](#_Toc38520771)

[4.3 数据流描述 11](#_Toc38520772)

[4.4 处理过程描述 12](#_Toc38520773)

[**三、** **数据库概念结构设计** 14](#_Toc38520774)

[1. 实体 14](#_Toc38520775)

[2. 分E-R图 15](#_Toc38520776)

[3. 总E-R图 17](#_Toc38520777)

[**四、** **数据库逻辑结构设计** 17](#_Toc38520778)

[1. 关系模式 17](#_Toc38520779)

[2. 规范化处理 17](#_Toc38520780)

[3. 表结构 18](#_Toc38520781)

[1.1 会员信息 18](#_Toc38520782)

[1.2 进货订单 18](#_Toc38520783)

[1.3 销售订单 19](#_Toc38520784)

[1.4 书籍信息 19](#_Toc38520785)

[1.5 书籍Items 20](#_Toc38520786)

[1.6 充值记录 20](#_Toc38520787)

[1.7 赔偿记录 21](#_Toc38520788)

[1.8 租借记录 21](#_Toc38520789)

[4. 数据库关系图 22](#_Toc38520790)

[**五、** **系统实现过程及完成效果介绍** 22](#_Toc38520791)

[1. 视图设计 22](#_Toc38520792)

[2. 索引设计 23](#_Toc38520793)

[3. 触发器设计 24](#_Toc38520794)

[4. 函数设计 24](#_Toc38520795)

[5. 存储过程设计 24](#_Toc38520796)

[6. 详情代码（点击最前方小三角可收起） 25](#_Toc38520797)

[**六、** **总结** 48](#_Toc38520798)

[**七、** **参考资料** 49](#_Toc38520799)

[**八、** **致谢** 49](#_Toc38520800)

1. **选题说明及需求介绍**
2. **背景**

随着书店规模越来越大，书籍越来越多，书店的信息量急剧增加，如果依旧采取传统的人工方式来管理这些信息，不仅会消耗大量的人力物力，并且还会造成书店信息混乱，增加了书店的运营成本。

这个时候就要用到一个软件来进行合理管理，和传统方式相比，软件管理有如下优点：查找方便、修改简单、可靠性高、保密性好、存储容量大、成本低等优点。基于以上优点，书店使用软件进行管理可以极大的提高工作效率、节省人力，能让书店更好的服务用户。

1. **需求介绍**

此系统主要控制权限在书店方，用户的注册由书店来实际操作完成

书店书刊出租和零售管理系统可分为以下几个模块：

「库存管理模块」、「进货管理模块」、「出租管理模块」、「销售管理模块」、「财务管理模块」、「会员管理模块」。具体如下。



1. **系统功能模块划分**

## 顶层数据流图



## 一层数据流图



## 二层数据流图

### 库存管理



### 进货管理



### 租借管理



### 销售管理



### 财务统计



### 会员管理



## 数据字典

### 数据结构描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **数据结构名** | **含义说明** | **组成** |
| E1 | 书店 | 书店 | 账号、密码 |

### 数据存储描述

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **编号** | **数据存储名** | **含义说明** | **流入的数据流** | **流出的数据流** | **组成** | **存取方式** | **存取位置** |
| DBMS1 | 会员信息 | 存储会员的一系列信息 | 注册会员的信息，会员充值的金额 | 会员账号、余额、手机号 | 会员账号、姓名、手机号、余额、冻结余额 | 文件 | 数据库 |
| DBMS2 | 书籍库存 | 记录书籍库存相关信息 | 进货信息、租借归还信息、图书丢失信息、销售信息 | 书籍号码、某书籍库存 | 书籍号码、库存数量 | 文件 | 数据库 |
| DBMS3 | 进货订单 | 存储每次进货时的订单信息 | 进货订单 | 进货单号、书籍号码、进货数量、进货单价 | 进货单号、书籍号码、进货数量、单价 | 文件 | 数据库 |
| DBMS4 | 销售表 | 记录书店的销售记录 | 购买信息 | 财务统计信息 | 日期、销售图书号码、销售金额 | 文件 | 数据库 |
| DBMS5 | 租借表 | 记录所有被租借的书籍情况 | 租借信息、归还信息、赔偿信息 | 租借信息统计 | 租借号、租借图书号码、租借会员账号、租借日期、是否续租 | 文件 | 数据库 |
| DBMS 6 | 充值记录 | 记录会员所有的充值记录 | 充值信息 | 财务统计信息 | 日期、会员账号、充值金额 | 文件 | 数据库 |
| DBMS7 | 赔偿表 | 记录租借图书丢失后赔偿的金额 | 赔偿图书信息、对应租借信息 | 财务统计信息 | 赔偿图书号码、赔偿会员账号、赔偿金额 | 文件 | 数据库 |
| DBMS8 | 书籍信息 | 记录所有书籍的信息 | 输入的书籍信息 | 信息查询 | 书籍号码、书名、类别、作者、书籍照片 | 文件 | 数据库 |
| DBMS9 | 租借历史表 | 记录所有被租借的书籍历史情况 | 租借信息、会员信息 | 租借信息统计 | 租借号、租借图书号码、租借会员账号、租借日期、归还日期 | 文件 | 数据库 |

### 数据流描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **含义说明** | **数据流来源** | **数据流去向** | **组成** |
| FBMS1.1 | 进行查询时输入的信息 | 书店 | 数据查询模块 | {类别、书名、书籍编号} |
| FBMS1.2 | 对某些书籍进行删除 | 书店 | 库存管理模块 | {书籍号码、数量} |
| FBMS1.3 | 查询或删除之后的返回信息 | 库存管理模块 | 书店 | {查询结果、删除结果} |
| FBMS1.4 | 进货的订单 | 书店 | 进货管理模块 | {进货单号、进货书籍、进货数量、进价} |
| FBMS1.5 | 进行图书租借处理时传入的信息 | 书店 | 租借管理模块 | {会员账号、租借书籍} |
| FBMS1.6 | 进行图书归还时传入信息 | 书店 | 租借管理模块 | {会员账号、租借书籍} |
| FBMS1.7 | 租借图书丢失后归还时传入的信息 | 书店 | 租借管理模块 | {会员账号、租借书籍} |
| FBMS1.8 | 购买图书时输入的信息 | 书店 | 销售管理 | {销售图书号、销售数量、单价} |
| FBMS1.9 | 购买图书操作完成后打印出的小票 | 销售管理 | 书店 | {销售图书号、销售图书名、数量、单价、总价格、日期} |
| FBMS1.10 | 财务统计系统返回的信息 | 书店 | 财务统计模块 | {进货支出、租借收入、销售收入、赔偿金额、总收入、盈利} |
| FBMS1.11 | 进行财务统计的日期范围 | 书店 | 财务统计模块 | {开始日期、结束日期} |
| FBMS1.12 | 会员进行注册时输入的信息 | 书店 | 会员管理模块 | {会员账号、姓名、手机号} |
| FBMS1.13 | 会员进行充值时的输入信息 | 书店 | 会员管理模块 | {会员账号、充值金额} |
| FBMS1.14 | 会员删除时输入的信息 | 书店 | 会员管理模块 | {会员账号} |
| FBMS1.15 | 查询会员信息时输入信息 | 书店 | 会员管理模块 | {会员账号} |
| FBMS1.16 | 对会员信息进行查询后返回的信息 | 会员管理模块 | 书店 | {会员账号、姓名、手机号、账户余额} |
| FBMS1.17 | 书店进行登陆时传入的信息 | 书店 |  | {账户、密码} |

### 处理过程描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **含义说明** | **输入数据流** | **输出数据流** | **处理** |
| PBMS2.1.1 | 查询条件 | 查询时输入的查询条件 | 转换后的查询条件 | 对输入的查询条件进行转换 |
| PBMS2.1.2 | 类别查询 | 经过转换后的查询 | 查询的结果 | 根据类别查询 |
| PBMS2.1.3 | 模糊查询 | 经过转换后的查询 | 查询的结果 | 根据书籍名或编号进行查询 |
| PBMS2.1.4 | 删除图书 | 选中的待删除图书 | 转换后结果 | 对输入的删除图书进行转换 |
| PBMS2.1.5 | 返回错误信息 | 删除图书时未选中图书 | 转换后的错误信息 | 对错误信息进行转换并输出 |
| PBMS2.1.6 | 检查租借情况 | 待删除图书信息 | 待删除图书信息、查询结果 | 查询待删除图书是否被租借 |
| PBMS2.1.7 | 进行删除 | 待删除图书信息 |  | 删除待删除图书 |
| PBMS2.2.1 | 记录进货订单 | 进货订单 | 订货单 | 对进货订单进行转换 |
| PBMS2.2.2 | 图书是否存在 | 订货单 | 订货单、查询结果 | 查询进货图书存在情况 |
| PBMS2.2.3 | 修改图书数量 | 订货单 |  | 对存在的图书直接进行修改库存 |
| PBMS2.2.4 | 添加图书信息并修改数量 | 订货单 |  | 对不存在的图书首先进行添加图书然后再修改图书数量 |
| PBMS2.3.1 | 获取图书信息 | 租借图书书名或编号、会员账号 | 图书信息、会员账号 | 根据书名或编号获取图书信息 |
| PBMS2.3.2 | 获取会员信息 | 图书信息、会员账号 | 图书信息、会员信息 | 根据会员账号获得会员的余额 |
| PBMS2.3.3 | 信息写入并冻结一定量余额 | 图书信息、会员信息 |  | 修改会员冻结余额、书记库存信息，并增加租借表信息 |
| PBMS2.3.4 | 获取会员信息 | 租借号、会员账号 | 租借号、会员信息 | 根据会员账号获得会员信息 |
| PBMS2.3.5 | 修改信息 | 租借号、会员信息 |  | 修改会员冻结余额、书记库存信息，并删除租借表信息 |
| PBMS2.3.6 | 获取会员信息 | 租借号、会员账号 | 租借号、会员信息 | 根据会员账号获得会员信息 |
| PBMS2.3.7 | 修改信息 | 租借号、会员信息 |  | 修改会员冻结余额、书记库存信息、删除租借表信息、并增加赔偿表信息 |
| PBMS2.3.8 | 获取账户信息 | 租借号、会员账号 | 租借号、会员信息 | 根据会员账号获得会员信息 |
| PBMS2.3.9 | 修改租借表 | 租借号、会员信息 |  | 根据租借号和会员账号修改租借信息 |
| PBMS2.4.1 | 修改书籍库存 | 付款信息 | 销售信息 | 进行信息转换并修改书籍的库存 |
| PBMS2.4.2 | 销售记录 | 销售信息 | 销售信息 | 将销售信息写入销售表 |
| PBMS2.4.3 | 销售小票打印 | 销售信息 | 小票 | 将销售信息转换为待打印的小票信息 |
| PBMS2.5.1 | 读入收入信息 | 统计时间 | 收入信息、统计时间 | 根据统计时间得到该时间段内的收入信息 |
| PBMS2.5.2 | 读入进货信息&库存图书 | 收入信息、统计时间 | 统计时间、收入、进货、库存信息 | 根据统计时间读入该时间段内进货和书籍库存信息 |
| PBMS2.5.3 | 打印报表 | 统计时间、收入、进货、库存信息 | 报表 | 根据收支信息等打印出报表 |
| PBMS2.6.1 | 会员信息读入 | 会员账号 | 会员信息 | 根据会员账号读入会员信息 |
| PBMS2.6.2 | 判断余额 | 会员信息 | 会员信息、余额判断结果 | 判断待删除会员余额是否为0 |
| PBMS2.6.3 | 删除失败 | 会员信息、余额判断结果 | 提示还有余额信息 | 返回删除失败提示 |
| PBMS2.6.4 | 会员删除 | 会员信息、余额判断结果 |  | 删除会员信息表信息 |
| PBMS2.6.5 | 会员信息读入 | 会员账号 | 会员信息 | 根据会员账号读入会员信息 |
| PBMS2.6.6 | 充值信息写入 | 会员信息 |  | 根据会员信息写入充值记录和会员信息 |
| PBMS2.6.7 | 会员信息读入 | 注册会员申请 | 会员信息 | 输入会员的信息 |
| PBMS2.6.8 | 手机号合理性检查 | 会员信息 | 会员信息 | 对手机号进行合理性检查 |
| PBMS2.6.9 | 会员信息写入 | 会员信息 |  | 对新注册会员信息进行写入 |

1. **数据库概念结构设计**

## 实体

* 1. 会员信息



* 1. 书籍库存



* 1. 进货订单



* 1. 销售清单



* 1. 书籍信息



* 1. 充值记录



* 1. 书籍Items



## 分E-R图

* 1. 会员信息-充值记录



* 1. 书籍Items-书籍信息-书籍库存



* 1. 书籍租借



* 1. 书籍赔偿



* 1. 书籍销售



* 1. 进货



## 总E-R图



1. **数据库逻辑结构设计**

## 关系模式

会员信息：（会员账号，姓名，手机号，余额，冻结余额，会员状态，备注）

书籍库存：（书籍号码，库存数量，全新数量）

进货订单：（进货单号，书籍号码，进货数量，单价，进货日期，备注）

销售清单：（日期，书籍号码，数量，销售单价，备注）

书籍信息：（书籍号码，书名，类别，作者，书籍照片，照片格式，库存数量, 全新数量，当前售价,书籍简介,备注）  
充值记录：（充值单号，日期，会员账号，充值金额，备注）

赔偿记录：（书籍号码，会员账号，赔偿金额，赔偿日期，备注）

租借记录：（租借号，会员账号，书籍号码，租借日期，是否续租，是否归还，应还日期，备注）

## 规范化处理

合并书记库存和书籍信息，减少数据冗余。

添加已归还表，将租借记录分为租借中和已归还俩个表。

添加书籍Item，具体化用于租借的每一本书。

修改后关系模式如下：

会员信息：（会员账号，姓名，手机号，余额，冻结余额，会员状态，备注）  
进货订单：（进货单号，书籍号码，进货数量，单价，进货日期，备注）  
销售清单：（日期，书籍号码，数量，销售单价，备注）  
书籍信息：（书籍号码，书名，类别，作者，书籍照片，照片格式，库存数量, 全新数量，当前售价,书籍简介,备注）  
书籍Item：（书籍ID，书籍号码，书籍状态，备注）  
充值记录：（充值单号，日期，会员账号，充值金额，备注）  
赔偿记录：（书籍ID，会员账号，赔偿金额，赔偿日期，备注）  
租借记录：（租借号，会员账号，书籍ID，租借日期，是否续租，应还日期，备注）  
已归还表：（租借号，会员账号，书籍ID，租借日期，归还日期，备注）

## 表结构

### 1.1 会员信息

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| memberID | char(8) | 否 | 前4位为注册年份 |  | 是 | 会员号 |
| memberName | nvarchar(10) | 否 |  |  | 否 | 会员名 |
| memberPhoneNumber | char(11) | 否 | 手机号长度为11位 |  | 否 | 会员手机号 |
| memberOverage | money | 否 | 大于等于0 |  | 否 | 会员账户余额 |
| memberFreezeOverage | money | 否 | 大于等于0 |  | 否 | 会员冻结余额 |
| memberState | nvarchar(2) | 否 | {‘正常’，‘注销’} | ‘正常’ | 否 | 会员状态 |
| Remarks | nvarchar(max) | 是 | - | NULL | 否 | 备注 |

### 1.2 进货订单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| PurchaseID | char(6) | 否 | 长度为6 | - | 是 | 进货单单号 |
| BookISBN | char(13) | 否 | 外键，参考书籍信息(BookISBN) | - | 是 | 进货的书籍ISBN号 |
| Amount | int | 否 | 大于等于0 | - | 否 | 此书籍进货数量 |
| BookPrice | money | 否 | 大于等于0 | - | 否 | 一本书籍的进价 |
| PurchaseDate | datetime | 否 | - | GETDATE() | 否 | 进货日期 |
| Remarks | nvarchar(max) | 是 | - | NULL | 否 | 备注 |

### 1.3 销售订单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| SellDate | DateTime | 否 | - | GETDATE() | 否 | 销售日期 |
| BookISBN | char(13) | 否 | 外键，参考书籍信息(BookISBN) | - | 否 | 销售的书籍ISBN号 |
| SellNum | int | 否 | 大于等于0 | 1 | 否 | 销售数量 |
| SellPrice | money | 否 | 大于等于0 | - | 否 | 销售单价 |
| Remarks | nvarchar(max) | 是 | - | NULL | 否 | 备注 |

### 1.4 书籍信息

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| BookISBN | char(13) | 否 | - | - | 是 | 书籍的ISBN号 |
| BookName | nvarchar(25) | 否 | - | - | 否 | 书名 |
| BookAuthor | nvarchar(15) | 否 | - | - | 否 | 作者名 |
| BookCategory | nvarchar(10) | 是 | - | NULL | 否 | 书籍类别 |
| BookPicture | image | 是 | - | NULL | 否 | 书籍照片 |
| PictureFormat | nvarchar(5) | 是 | - | NULL | 否 | 照片格式 |
| BookNum | int | 否 | 大于等于0 | - | 否 | 库存数量 |
| BrandNewBookNum | int | 否 | 大于等于0 | - | 否 | 全新数量 |
| BookPrice | money | 否 | 大于等于0 | - | 否 | 书籍售价 |
| BookSummary | varbinary(max) | 是 | 用来存储\*.txt文件 | - | 否 | 书籍简介 |
| Remarks | nvarchar(max) | 是 | - | - | 否 | 备注 |

### 1.5 书籍Items

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| BookID | char(7) | 否 | - | - | 是 | 此书籍的ID号 |
| BookISBN | char(13) | 否 | 外键，参考书籍信息(BookISBN) | - | 否 | 此书籍的ISBN号 |
| BookStatus | varchar(3) | 否 | ('被租借','在店','已丢失') | ‘在店’ | 否 | 此书籍状态 |
| Remarks | nvarchar(max) | 是 | - | - | 否 | 备注 |

### 1.6 充值记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| RechargeID | char(10) | 否 | 前四位位充值年份 | - | 是 | 充值单号 |
| RechargeDate | DateTime | 否 | - | GETDATE() | 否 | 充值日期 |
| memberID | char(8) | 否 | 外键，参考会员信息(memberID) | - | 否 | 充值会员号 |
| RechargeMoney | money | 否 | 大于等于0 | - | 否 | 充值金额 |
| Remarks | nvarchar(max) | 是 | - | - | 否 | 备注 |

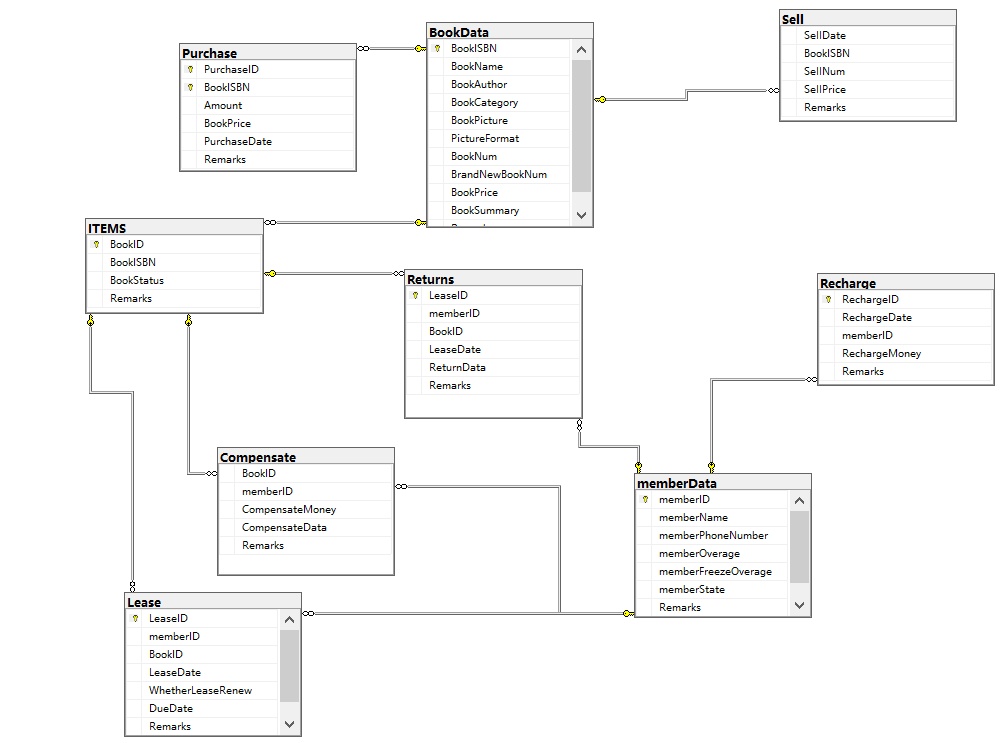
### 1.7 赔偿记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| BookID | char(7) | 否 | 外键，参考书籍Items(BookID) | - | 否 | 赔偿的书籍ID号 |
| memberID | char(8) | 否 | 外键，参考会员信息(memberID) | - | 否 | 赔偿会员的ID |
| CompensateMoney | money | 否 | 大于0 | - | 否 | 赔偿金额 |
| CompensateDate | DateTime | 否 | - | GETDATE() | 否 | 赔偿日期 |
| Remarks | nvarchar(max) | 是 | - | - | 否 | 备注 |

### 1.8 租借记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 取值类型 | 可否为空 | 取值 | 默认值 | 是否主键 | 备注 |
| LeaseID | char(10) | 否 | 前六位为租借年份月份 | - | 是 | 租借号 |
| memberID | char(8) | 否 | 外键，参考会员信息(memberID) | - | 否 | 租借会员ID |
| BookID | char(7) | 否 | 外键，参考书籍Items(BookID) | - | 否 | 租借书籍 |
| LeaseDate | DateTime | 否 | - |  | 否 | 租借日期 |
| WhetherLeaseRenew | nchar(1) | 否 | {‘是’,’否’} | ‘否’ | 否 | 是否续借 |
| WhetherReturn | nchar(1) | 否 | {‘是’,’否’} | ‘否‘ | 否 | 是否归还 |
| DueDate | DateTime | 否 | 大于LeaseDate | DATEADD(day,30,GETDATE()) | 否 | 归还截止日期 |
| Remarks | nvarchar(max) | 是 | - | - | 否 | 备注 |

## 数据库关系图



1. **系统实现过程及完成效果介绍**

搭建完整的后台数据库，实现相应的安全性，完整性控制，创建需要的视图，索引等；为前台的功能部分实现后台的触发器和存储过程，函数等。

## 视图设计

创建视图之后，后续查找可以基于视图查找，简化了后续程序编写难度。

* 1. *会员拓展视图(会员账号、总充值金额、现租借图书数量)*
  2. *逾期未归还图书视图(姓名、手机号、书籍名、应还日期)*
  3. *库存过少书籍(书籍号码、书名、库存数量、全新数量)*
  4. *本月销售清单(日期、书籍号码、书名、类别、销售金额)*
  5. *进货简化视图(进货单号、进货总数量、进货日期、进货总价)*
  6. *未归还图书视图(租借号、会员名、会员手机号、书名、租借日期、是否续借、租借截止日期)*
  7. *图书拓展视图(书籍号码、书名、书籍类别、书籍作者、库存数量、全新数量、图书价格、销售数量、租借次数)*
  8. *销售拓展视图(日期、书籍号码、书名、类别、销售数量、销售金额)*
  9. *创建图书-Items视图(除了Items.Remark都有)*

## 索引设计

对经常需要进行查询的属性设置索引，以加快系统速度，若设置索引属性的取值唯一可设置唯一索引。

* 1. *会员手机号添加唯一索引*
  2. *进货订单书籍号码添加索引*
  3. *销售清单书籍号码添加索引*
  4. *书籍信息书名添加索引*
  5. *书籍信息作者添加索引*
  6. *充值记录会员账号添加索引*
  7. *赔偿记录会员账户添加索引*
  8. *赔偿记录书籍号码添加索引*
  9. *租借记录会员账号添加索引*
  10. *租借记录书籍号码添加索引*
  11. *已归还书籍号码添加索引*
  12. *已归还会员账号添加索引*

## 触发器设计

设计触发器主要用来实现数据同步更新，且设置了一个触发器来限制会员同时租借俩本相同的书。因为触发器过多会增加系统维护的成本，所以这里除了必要的同步更新数据，并没有设置过多触发器。

* 1. *修改续租状态时修改应还日期*
  2. *删除归还记录时修改库存数量、冻结余额，并将数据转移到returns*
  3. *插入租借记录时修改库存数量、冻结余额，并检查此会员是否租借过此图书*
  4. *插入销售清单时修改全新数量*
  5. *插入进货订单时修改全新数量*
  6. *插入充值记录时修改会员信息*
  7. *插入赔偿记录时修改书籍库存信息*

## 函数设计

此处函数主要用于后续的会员号、充值号、和租借号的分配。

* 1. *获得会员总人数*
  2. *获得本年总充值记录数*
  3. *获得本年本月总租借数*
  4. *获得下一个Items的ID*

## 存储过程设计

* 1. *获得某一时间段销售记录*
  2. *通过手机号获得某会员租借记录*
  3. *通过会员ID获得某会员租借记录*
  4. *通过手机号获得某会员赔偿记录*
  5. *通过会员ID获得某会员赔偿记录*
  6. *通过手机号获得某会员充值记录*
  7. *通过会员ID获得某会员充值记录*
  8. *获得某一时间段进货清单*
  9. *根据某一进货单号获得具体进货信息*
  10. *根据图书号码获得某一图书具体信息*
  11. *根据图书名获得匹配到的所有图书*
  12. *根据作者名获得匹配到的所有图书*
  13. *根据类别获得匹配到的所有图书*
  14. *根据会员名获得匹配到的所有会员*
  15. *根据会员手机号匹配会员*
  16. *根据会员ID匹配会员*
  17. *根据图书名和作者名搜索图书*
  18. *根据图书名和类别搜索图书*
  19. *根据作者名和类别搜索图书*
  20. *根据图书名、作者名和类别搜索图书*
  21. *修改某一书籍用于租借的数量存储过程*

## 详情代码（点击最前方小三角可收起）

*--region 一、建立数据库*

CREATE DATABASE BookStore  
ON PRIMARY(  
 name='BookStore\_data',  
 filename='D:\Projects\SQLServer\BookStore\BookStore\_data.mdf',  
 size = 10MB,  
 MAXSIZE = UNLIMITED,  
 filegrowth=5MB  
 )  
LOG ON(  
 name = 'BookStore\_log',  
 filename='D:\Projects\SQLServer\BookStore\BookStore\_log.ldf',  
 size = 10MB,  
 MAXSIZE = UNLIMITED,  
 filegrowth=5MB  
 )  
GO  
*--endregion  
  
--region 二、建表*USE BookStore  
  
*--会员信息：（会员账号，姓名，手机号，余额，冻结余额，会员状态，备注）  
--进货订单：（进货单号，书籍号码，进货数量，单价，进货日期，备注）  
--销售清单：（日期，书籍号码，数量，销售单价，备注）  
--书籍信息：（书籍号码，书名，类别，作者，书籍照片，照片格式，库存数量, 全新数量，当前售价,书籍简介,备注）  
--书籍Item：（书籍ID，书籍号码，书籍状态，备注）  
--充值记录：（充值单号,日期，会员账号，充值金额，备注）  
--赔偿记录：（书籍ID，会员账号，赔偿金额，赔偿日期，备注）  
--租借记录：（租借号，会员账号，书籍ID，租借日期，是否续租，应还日期，备注）  
--已归还表：（租借号，会员账号，书籍ID，租借日期，归还日期，备注）  
  
--1.建会员表*CREATE TABLE memberData(  
 memberID char(8) PRIMARY KEY,  
 memberName nvarchar(10) Not NULL,  
 memberPhoneNumber char(11) UNIQUE Not Null,  
 memberOverage money DEFAULT 0 check(memberOverage >= 0) Not Null,  
 memberFreezeOverage money DEFAULT 0 check(memberFreezeOverage >= 0) Not Null,  
 memberState nvarchar(2) check(memberState IN (N'正常',N'注销')) DEFAULT N'正常' Not Null,  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--2.建进货表*CREATE TABLE Purchase(  
 PurchaseID char(6),  
 BookISBN char(13) FOREIGN KEY REFERENCES BookData(BookISBN) ON UPDATE CASCADE,  
 Amount int NOT NULL,  
 BookPrice money NOT NULL,  
 PurchaseDate DateTime NOT NULL DEFAULT GETDATE(),  
 Remarks nvarchar(max) DEFAULT NULL,  
 PRIMARY KEY (PurchaseID, BookISBN)  
)  
  
*--3.建销售表*CREATE TABLE Sell(  
 SellDate DateTime NOT NULL DEFAULT GETDATE(),  
 BookISBN char(13) Not NULL FOREIGN KEY REFERENCES BookData(BookISBN) ON UPDATE CASCADE,  
 SellNum int Not NULL DEFAULT 1 check(SellNum > 0),  
 SellPrice money NOT NULL check(SellPrice > 0),  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--4.建书籍表*CREATE TABLE BookData(  
 BookISBN char(13) PRIMARY KEY,  
 BookName nvarchar(25) NOT NULL,  
 BookAuthor nvarchar(15) NOT NULL,  
 BookCategory nvarchar(10) DEFAULT NULL,  
 BookPicture image DEFAULT NULL,  
 PictureFormat nvarchar(5) DEFAULT NULL,  
 BookNum int NOT NULL check (BookNum >=0),  
 BrandNewBookNum int NOT NULL check (BrandNewBookNum >=0),  
 BookPrice money NOT NULL check (BookPrice >=0),  
 BookSummary varbinary(max) DEFAULT NULL,  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--5.建书籍Item表*CREATE TABLE ITEMS(  
 BookID char(7) PRIMARY KEY,  
 BookISBN char(13) NOT NULL FOREIGN KEY REFERENCES BookData(BookISBN) ON UPDATE CASCADE,  
 BookStatus varchar(3) check (BookStatus in (N'被租借',N'在店',N'已丢失')) NOT NULL DEFAULT N'在店',  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--6.建充值表*CREATE TABLE Recharge(  
 RechargeID char(10) PRIMARY KEY,  
 RechargeDate DateTime NOT NULL DEFAULT GETDATE(),  
 memberID char(8) NOT NULL FOREIGN KEY REFERENCES memberData(memberID),  
 RechargeMoney money NOT NULL check (RechargeMoney >=0),  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--7.建赔偿表*CREATE TABLE Compensate(  
 BookID char(7) NOT NULL Foreign KEY REFERENCES Items(BookID),  
 memberID char(8) NOT NULL FOREIGN KEY REFERENCES memberData(memberID),  
 CompensateMoney money NOT NULL check (CompensateMoney > 0),  
 CompensateData DateTime NOT NULL DEFAULT GETDATE(),  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--8.建租借表*CREATE TABLE Lease(  
 LeaseID char(10) PRIMARY KEY,  
 memberID char(8) FOREIGN KEY REFERENCES memberData(memberID),  
 BookID char(7) NOT NULL Foreign KEY REFERENCES Items(BookID),  
 LeaseDate DateTime NOT NULL DEFAULT GETDATE(),  
 WhetherLeaseRenew nchar(1) CHECK(WhetherLeaseRenew in (N'是', N'否')) DEFAULT N'否',  
 DueDate DateTime NOT NULL DEFAULT DATEADD(day,30,GETDATE()) check(DATEDIFF(day,GETDATE(),DueDate)>0),  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--9.建已归还表*CREATE TABLE Returns(  
 LeaseID char(10) PRIMARY KEY,  
 memberID char(8) FOREIGN KEY REFERENCES memberData(memberID),  
 BookID char(7) NOT NULL Foreign KEY REFERENCES Items(BookID),  
 LeaseDate DateTime NOT NULL,  
 ReturnData DateTime NOT NULL DEFAULT GETDATE(),  
 Remarks nvarchar(max) DEFAULT NULL  
)  
  
*--endregion  
  
--region 三、创建视图  
  
-- 1.会员拓展视图(会员账号、总充值金额、现租借图书数量)  
-- 2.逾期未归还图书视图(姓名、手机号、书籍名、应还日期)  
-- 3.库存过少书籍(书籍号码、书名、库存数量、全新数量)  
-- 4.本月销售清单(日期、书籍号码、书名、类别、销售金额)  
-- 5.进货简化视图(进货单号、进货总数量、进货日期、进货总价)  
-- 6.租借拓展视图(租借号、会员名、会员手机号、书名、租借日期、是否续借、租借截止日期)  
-- 7.图书拓展视图(书籍号码、书名、书籍类别、书籍作者、库存数量、全新数量、图书价格、销售数量、租借次数)  
-- 8.销售拓展视图(日期、书籍号码、书名、类别、销售数量、销售金额)  
-- 9.创建图书-Items视图(除了Items.Remark都有)  
  
--region 1.会员拓展视图*GO  
create view memberView as  
 select A.memberID, AllRechargeMoney, (B.BorrowingBook + C.BorrowingBook) BorrowingBook  
 from (select memberData.memberID, sum(RechargeMoney) AllRechargeMoney from memberData, Recharge where memberData.memberID = Recharge.memberID and memberData.memberState=N'正常' group by memberData.memberID) as A,  
 (select memberData.memberID, count(LeaseID) BorrowingBook from memberData, Lease where memberData.memberID = Lease.memberID group by memberData.memberID) as B,  
 (select memberData.memberID, count(LeaseID) BorrowingBook from memberData, Returns where memberData.memberID = Returns.memberID group by memberData.memberID) as C  
 where A.memberID = B.memberID  
 and B.memberID = C.memberID  
*--endregion  
  
--region 2.逾期未归还图书视图*GO  
create view OverdueBook as  
 select LeaseID,Lease.BookID,memberName,memberPhoneNumber,BookName,LeaseDate,WhetherLeaseRenew,DueDate  
 from memberData,Lease,BookData,Items  
 where memberData.memberID = Lease.memberID  
 and Lease.BookID = Items.BookID  
 and Items.BookISBN = BookData.BookISBN  
 and 0>DATEDIFF(day,GETDATE(), Lease.DueDate)  
 and memberData.memberState=N'正常'  
*--endregion  
  
--region 3.库存过少图书*GO  
create view LowInventory as  
 select BookISBN, BookName, BookNum,BrandNewBookNum  
 from BookData  
 where BookNum<10  
 or BrandNewBookNum<5  
*--endregion  
  
---region 4.本月销售清单*GO  
create view ThisMonthSSalesList as  
 select SellDate,BookData.BookISBN,BookData.BookName,BookData.BookCategory,SellPrice  
 from Sell,BookData  
 where Sell.BookISBN = BookData.BookISBN  
 and DATEDIFF(month,GETDATE(), Sell.SellDate)=0  
*--endregion  
  
--region 5.进货简化视图*GO  
create view PurchaseView as  
 select PurchaseID,sum(Amount) sumCount,PurchaseDate,sum(Amount\*BookPrice) sumMoney  
 from Purchase  
 group by Purchase.PurchaseID, PurchaseDate  
*--endregion  
  
--region 6.现租借拓展视图*GO  
create view LeaseView as  
 select LeaseID,memberName,memberPhoneNumber,BookName,LeaseDate,WhetherLeaseRenew,DueDate  
 from Lease, memberData, BookData, Items  
 where memberData.memberID = Lease.memberID  
 and Lease.BookID=Items.BookID  
 and Items.BookISBN=BookData.BookISBN  
*--endregion  
  
--region 7.图书拓展视图*GO  
create view BookView as  
 select A.BookISBN, A.BookName,A.BookCategory,A.BookAuthor,A.BookNum,A.BrandNewBookNum,A.BookPrice,A.SellCount SellCount,(B.LeaseCount+C.LeaseCount) LeaseCount, D.AtStoreCount  
 from (select BookData.BookISBN, BookData.BookName,BookData.BookCategory,BookData.BookAuthor,BookData.BookNum,BookData.BrandNewBookNum,BookData.BookPrice,sum(Sell.SellNum) SellCount from BookData,Sell where BookData.BookISBN = Sell.BookISBN group by BookData.BookISBN,BookName,BookNum,BookCategory,BookAuthor,BrandNewBookNum,BookPrice) A,  
 (select BookData.BookISBN,count(\*) LeaseCount from BookData,Lease,Items where BookData.BookISBN=Items.BookISBN and Items.BookID = Lease.BookID group by BookData.BookISBN) B,  
 (select BookData.BookISBN,count(\*) LeaseCount from BookData,Returns,Items where BookData.BookISBN=Items.BookISBN and Items.BookID = Returns.BookID group by BookData.BookISBN) C,  
 (select BookData.BookISBN,count(\*) AtStoreCount from BookData,Returns,Items where BookData.BookISBN=Items.BookISBN and BookStatus = N'在店' group by BookData.BookISBN) D  
 where A.BookISBN=B.BookISBN  
 and B.BookISBN=C.BookISBN  
 and C.BookISBN=D.BookISBN  
*--endregion  
  
--region 8.销售拓展视图*GO  
create view SellView as  
 select SellDate,BookData.BookISBN,BookData.BookName,BookData.BookCategory,SellNum,SellPrice  
 from Sell,BookData  
 where Sell.BookISBN = BookData.BookISBN  
*--endregion  
  
--region 9.创建图书-Items视图*Go  
create view BookItemsView as  
 select Items.BookID, ITEMS.BookISBN, ITEMS.BookStatus,  
 BookData.BookName, BookData.BookAuthor, BookData.BookCategory, BookData.BookPicture, BookData.PictureFormat, BookData.BookNum, BookData.BrandNewBookNum, BookData.BookPrice, BookData.BookSummary, BookData.Remarks  
 from BookData,Items where BookData.BookISBN = Items.BookISBN  
*--endregion  
--endregion  
  
--region 四、添加索引  
  
--1. 会员手机号添加唯一索引  
--2. 进货订单书籍号码添加索引  
--3. 销售清单书籍号码添加索引  
--4. 书籍信息书名添加索引  
--5. 书籍信息作者添加索引  
--6. 充值记录会员账号添加索引  
--7. 赔偿记录会员账户添加索引  
--8. 赔偿记录书籍号码添加索引  
--9. 租借记录会员账号添加索引  
--10. 租借记录书籍号码添加索引  
--11. 已归还书籍号码添加索引  
--12. 已归还会员账号添加索引  
  
--region 1.会员手机号添加唯一索引*create UNIQUE  
INDEX PhoneNumberIndex  
ON memberData(memberPhoneNumber)  
*--endregion  
  
--region 2.进货订单书籍号码添加索引*create Index PurchaseIndex  
on Purchase(BookISBN)  
*--endregion  
  
--region 3.销售清单书籍号码添加索引*create Index SellIndex  
on Sell(BookISBN)  
*--endregion  
  
--region 4.书籍信息书名添加索引*create Index BookNameIndex  
on BookData(BookName)  
*--endregion  
  
--region 5.书籍信息作者添加索引*create Index BookAuthorIndex  
on BookData(BookAuthor)  
*--endregion  
  
--region 6.充值记录会员账号添加索引*create Index RechargeMemberIDIndex  
on Recharge(memberID)  
*--endregion  
  
--region 7.赔偿记录会员账户添加索引*create Index CompensateMemberIdIndex  
on Compensate(memberID)  
*--endregion  
  
--region 8.赔偿记录书籍号码添加索引*create Index CompensateBookISBNIndex  
on Compensate(BookID)  
*--endregion  
  
--region 9.租借记录会员账号添加索引*create Index LeaseMemberIDIndex  
on Lease(memberID)  
*--endregion  
  
--region 10.租借记录书籍号码添加索引*create Index LeaseBookISBNIndex  
on Lease(BookID)  
*--endregion  
  
--region 11. 已归还书籍号码添加索引*Create Index ReturnsBookISBNIndex  
on Returns(BookID)  
*--endregion  
  
--region 12. 已归还会员账号添加索引*Create Index ReturnsMemberIDIndex  
on Returns(MemberID)  
*--endregion  
  
--endregion  
  
--region 五、创建触发器  
-- 触发器：审核、数据同步功能，当写入某个数据时检查数据是否合理，是否与其他数据冲突、且同步其他表中数据  
--1.修改续租状态时修改应还日期  
--2.删除归还记录时修改库存数量、冻结余额，并将数据转移到returns  
--3.插入租借记录时修改库存数量、冻结余额  
--4.插入销售清单时修改全新数量  
--5.插入进货订单时修改全新数量  
--6.插入充值记录时修改会员信息  
--7.插入赔偿记录时修改书籍库存信息、冻结金额和余额，并删除租借图书记录，修改Items状态  
  
--region 1.修改续租状态时修改应还日期*Go  
Create TRIGGER alterLeaseRenew  
on Lease  
AFTER Update  
as  
 begin  
 if Update(WhetherLeaseRenew)  
 begin  
 declare @NewDueData DateTime  
 declare @WhetherLeaseRenew char(1)  
 declare @LeaseID char(10)  
 select @NewDueData = DATEADD(day,60,inserted.LeaseDate),@WhetherLeaseRenew = inserted.WhetherLeaseRenew,@LeaseID = inserted.LeaseID from inserted  
 if @WhetherLeaseRenew=N'是'  
 begin  
 update Lease  
 set Lease.DueDate = @NewDueData  
 where Lease.LeaseID = @LeaseID  
 end  
 end  
 end  
*--endregion  
  
--region 2.插入已归还表时，删除归还记录，并修改库存数量、冻结余额*GO  
CREATE TRIGGER insertReturns  
on Returns  
after insert  
as  
 begin  
 declare @memberID char(8)  
 declare @BookID char(7)  
 declare @LeaseDate DateTime  
 declare @BookPrice money  
 declare @BookISBN char(13)  
 declare @LeaseID char(10)  
 declare @LeaseDay int  
  
 select @LeaseID = LeaseID,@BookID = BookID,@memberID = MemberID,@LeaseDate = LeaseDate from inserted  
  
 *--Delete Lease Data* delete from Lease where LeaseID = @LeaseID  
  
 set @LeaseDay = DATEDIFF(day,@LeaseDate,GETDATE())  
  
 select @BookPrice = BookPrice,@BookISBN = BookISBN  
 from BookItemsView where BookID= @BookID  
  
 *--Update Book count* update BookData  
 set BookNum = BookNum+1  
 where BookData.BookISBN = @BookISBN  
  
 *--Update Member Overage* update memberData  
 set memberFreezeOverage -= @BookPrice,  
 memberOverage += (@BookPrice - @LeaseDay\*0.1)  
 where memberData.memberID = @memberID  
end  
*--endregion  
  
--region 3.插入租借记录时修改库存数量、冻结余额*Go  
CREATE TRIGGER insertLeaseData  
on Lease  
after INSERT  
as  
 begin  
 declare @memberID char(8)  
 declare @BookID char(7)  
 declare @BookISBN char(13)  
 declare @BookPrice money  
  
 select @memberID = inserted.memberID,@BookID = inserted.BookID from inserted  
 select @BookPrice = BookPrice,@BookISBN = BookISBN  
 from BookItemsView where BookID = @BookID  
  
 if exists (select \* from Lease,BookItemsView where BookISBN = @BookISBN and Lease.memberID = @memberID and Lease.BookID = BookItemsView.BookID)  
 begin  
 rollback transaction;  
 end  
  
 update BookData  
 set BookNum -= 1  
 where BookData.BookISBN = @BookISBN  
  
 update memberData  
 set memberFreezeOverage += @BookPrice  
 where memberData.memberID = @memberID  
 end  
*--endregion  
  
--region 4.插入销售清单时修改全新数量、库存数量*GO  
CREATE TRIGGER insertSellData  
on Sell  
after insert  
as  
 begin  
 declare SellCursor cursor FORWARD\_ONLY static  
 for select BookISBN,SellNum from inserted  
  
 declare @BookISBN char(13)  
 declare @BookNum int  
  
 open SellCursor  
  
 fetch next from SellCursor into @BookISBN,@BookNum  
  
 While @@fetch\_status = 0  
 begin  
 update BookData  
 set BookNum -= 1,  
 BrandNewBookNum -= 1  
 where BookData.BookISBN = @BookISBN  
 fetch next from SellCursor into @BookISBN,@BookNum  
 end  
 close SellCursor  
 deallocate SellCursor  
 end  
*--endregion  
  
--region 5.插入进货订单时修改全新数量*Go  
CREATE TRIGGER insertPurchaseData  
on Purchase  
after insert  
as  
 begin  
 declare PurchaseCursor CURSOR FORWARD\_ONLY static  
 for select BookISBN,Amount from inserted  
  
 declare @BookISBN char(13)  
 declare @Amount int  
  
 open PurchaseCursor  
 fetch next from PurchaseCursor into @BookISBN,@Amount  
 while @@fetch\_status = 0  
 begin  
 update BookData  
 set BookNum += @Amount,  
 BrandNewBookNum += @Amount  
 where BookData.BookISBN = @BookISBN  
 fetch next from PurchaseCursor into @BookISBN,@Amount  
 end  
 close PurchaseCursor  
 deallocate PurchaseCursor  
 end  
*--endregion  
  
--region 6.插入充值记录时修改会员信息*Go  
CREATE TRIGGER insertRechargeData  
on Recharge  
after insert  
as  
 begin  
 declare @memberID char(8)  
 declare @RechargeMoney money  
 select @memberID = inserted.memberID,@RechargeMoney = inserted.RechargeMoney from inserted  
  
 update memberData  
 set memberOverage += @RechargeMoney  
 where memberData.memberID = @memberID  
 end  
*--endregion  
  
--region 7.插入赔偿记录时修改书籍库存信息、冻结金额和余额，并删除租借图书记录，修改Items状态*Go  
CREATE TRIGGER insertCompensateData  
on Compensate  
after insert  
as  
 begin  
 declare @memberID char(8)  
 declare @BookISBN char(13)  
 declare @CompensationMoney money  
 declare @BookID char(7)  
  
 select @memberID = inserted.memberID,@BookID = inserted.BookID,@CompensationMoney = inserted.CompensateMoney from inserted  
  
 select @BookISBN = BookISBN from BookItemsView where BookItemsView.BookID = @BookID  
  
 update BookData  
 set BookNum -= 1  
 where BookData.BookISBN = @BookISBN  
  
 update memberData  
 set memberOverage -= @CompensationMoney,  
 memberFreezeOverage -= @CompensationMoney  
 where memberData.memberID = @memberID  
  
 update ITEMS  
 set BookStatus = N'已丢失'  
 where @BookID = Items.BookID  
  
 delete from Lease  
 where Lease.memberID = @memberID  
 and Lease.BookID= @BookID  
  
 delete from Items  
 where Items.BookID = @BookID  
 end  
*--endregion  
  
--endregion  
  
-- region 六、创建函数  
-- 函数：必须有返回值、用来拓展标准SQL语句，可以嵌入SQL语句中  
--1.获得会员总人数  
--2.获得本年总充值记录数  
--3.获得本年本月总租借数  
--4.获得下一个Items的ID  
  
--region 1.获得会员总人数*Go  
Create Function getAllMembersNum()  
returns int  
as  
 begin  
 declare @count int  
 select @count = count(\*) from memberData  
 return @count  
 end  
*--endregion  
  
--region 2.获得本年总充值记录数*Go  
create function getYearRechargeCount()  
returns int  
as  
 begin  
 declare @count int  
 select @count = count(\*) from Recharge  
 where DATEPART(year,Recharge.RechargeDate) = DATEPART(year,GETDATE())  
 return @count  
 end  
*--endregion  
  
--region 3.获得本年本月总租借数*Go  
create function getThisMonthLeaseCount()  
returns int  
as  
 begin  
 declare @count int  
 select @count = count(\*) from Lease  
 where DATEPART(year,Lease.LeaseDate) = DATEPART(year,GETDATE())  
 and DATEPART(month,Lease.LeaseDate) = DATEPART(month,GETDATE())  
 return @count  
 end  
*--endregion  
  
--region 4.获得下一个Items的ID*Go  
Create Function getNextID(@BookISBN char(13))  
returns char(7)  
as  
 begin  
 declare @count int  
 select @count = count(\*) from Items where Items.BookISBN = @BookISBN  
 return replace(str(@count,7),'','0')  
 end  
*--endregion  
  
--endregion  
  
--region 七、存储过程  
-- 存储过程：对某些功能进行封装， 避免了程序和数据库通信时传输大量代码  
--1.获得某一时间段销售记录  
--2.通过手机号获得某会员租借记录  
--3.通过会员ID获得某会员租借记录  
--4.通过手机号获得某会员赔偿记录  
--5.通过会员ID获得某会员赔偿记录  
--6.通过手机号获得某会员充值记录  
--7.通过会员ID获得某会员充值记录  
--8.获得某一时间段进货清单  
--9.根据某一进货单号获得具体进货信息  
--10.根据图书号码获得某一图书具体信息  
--11.根据图书名获得匹配到的所有图书  
--12.根据作者名获得匹配到的所有图书  
--13.根据类别获得匹配到的所有图书  
--14.根据会员名获得匹配到的所有会员  
--15.根据会员手机号匹配会员  
--16.根据会员ID匹配会员  
--17.根据图书名和作者名搜索图书  
--18.根据图书名和类别搜索图书  
--19.根据作者名和类别搜索图书  
--20.根据图书名、作者名和类别搜索图书  
--21.修改某一书籍用于租借的数量存储过程  
  
  
--region 1.获取某一时间段销售记录*Go  
CREATE PROCEDURE getPartTimeSellData  
 @beginTime DateTime,  
 @endTime DateTime = NULL  
as  
 begin  
 if @endTime is NULL  
 set @endTime = GETDATE()  
 select \*  
 from SellView  
 where DATEDIFF(day,@beginTime,SellView.SellDate)>0  
 and DATEDIFF(day,SellView.SellDate,@endTime)>0  
 end  
*--endregion  
  
--region 2.通过手机号获得某会员租借记录*Go  
CREATE PROCEDURE getLeaseDataByPhoneNumber  
 @PhoneNumber char(11)  
as  
 begin  
 select memberData.memberID,memberName,memberPhoneNumber,BookItemsView.BookISBN,BookName,Lease.LeaseDate  
 from memberData,Lease,BookItemsView  
 where memberData.memberID= Lease.memberID  
 and Lease.BookID = BookItemsView.BookID  
 and memberData.memberPhoneNumber = @phoneNumber  
 union  
 select memberData.memberID,memberName,memberPhoneNumber,BookItemsView.BookISBN,BookName,Returns.LeaseDate  
 from memberData,Returns,BookItemsView  
 where memberData.memberID= Returns.memberID  
 and Returns.BookID = BookItemsView.BookID  
 and memberData.memberPhoneNumber = @phoneNumber  
 end  
*--endregion  
  
--region 3.通过会员ID获得某会员租借记录*GO  
CREATE PROCEDURE getLeaseDataByMemberID  
 @memberID varchar(8)  
as  
 begin  
 select memberData.memberID,memberName,memberPhoneNumber,BookItemsView.BookISBN,BookName  
 from memberData,Lease,BookItemsView  
 where memberData.memberID= Lease.memberID  
 and Lease.BookID = BookItemsView.BookID  
 and memberData.memberID like '%'+@memberID+'%'  
 union  
 select memberData.memberID,memberName,memberPhoneNumber,BookItemsView.BookISBN,BookName  
 from memberData,Returns,BookItemsView  
 where memberData.memberID= Returns.memberID  
 and Returns.BookID = BookItemsView.BookID  
 and memberData.memberID like '%'+@memberID+'%'  
 end  
*--endregion  
  
--region 4.通过手机号获得某会员赔偿记录*GO  
CREATE PROCEDURE getCompensateDataByPhoneNumber  
 @phoneNumber char(11)  
as  
 begin  
 select memberData.memberID,memberName,memberPhoneNumber,BookItemsView.BookID,BookName  
 from Compensate,memberData,BookItemsView  
 where memberData.memberID = Compensate.memberID  
 and BookItemsView.BookID = Compensate.BookID  
 and memberData.memberPhoneNumber = @phoneNumber  
 end  
*--endregion  
  
--region 5.通过会员ID获得某会员赔偿记录*Go  
CREATE PROCEDURE getCompensateDataByMemberID  
 @memberID varchar(8)  
as  
 begin  
 select memberData.memberID,memberName,memberPhoneNumber,BookItemsView.BookID,BookName  
 from Compensate,memberData,BookItemsView  
 where memberData.memberID = Compensate.memberID  
 and BookItemsView.BookID = Compensate.BookID  
 and memberData.memberPhoneNumber like '%'+@memberID+'%'  
 end  
*--endregion  
  
--region 6.通过手机号获得某会员充值记录*GO  
CREATE PROCEDURE getRechargeDataByPhoneNumber  
 @phoneNumber char(11)  
as  
 begin  
 select memberData.memberID,memberName,memberPhoneNumber,RechargeID,RechargeDate,RechargeMoney  
 from Recharge,memberData  
 where memberData.memberID = Recharge.memberID  
 and memberPhoneNumber = @phoneNumber  
 end  
*--endregion  
  
--region 7.通过会员ID获得某会员充值记录*GO  
CREATE PROCEDURE getRechargeDataByMemberID  
 @memberID varchar(11)  
as  
 begin  
 select memberData.memberID,memberName,memberPhoneNumber,RechargeID,RechargeDate,RechargeMoney  
 from Recharge,memberData  
 where memberData.memberID = Recharge.memberID  
 and memberData.memberID like '%'+@memberID+'%'  
 end  
*--endregion  
  
--region 8.获得某一时间段进货清单*Go  
CREATE PROCEDURE getPartTimePurchaseData  
 @beginTime DateTime,  
 @endTime DateTime = NULL  
as  
 begin  
 if @endTime IS NULL  
 set @endTime = GETDATE()  
 select PurchaseID,PurchaseDate,BookData.BookISBN,BookName,Purchase.BookPrice,Purchase.Amount  
 from Purchase,BookData  
 where BookData.BookISBN = Purchase.BookISBN  
 and DATEDIFF(day,@beginTime,PurchaseDate)>0  
 and DATEDIFF(day,PurchaseDate,@endTime)>0  
 end  
*--endregion  
  
--region 9.根据某一进货单号获得具体进货信息*Go  
CREATE PROCEDURE getPurchaseDataByPurchaseId  
 @PurchaseID varchar(6)  
as  
 begin  
 select PurchaseID,PurchaseDate,BookData.BookISBN,BookName,Purchase.BookPrice,Purchase.Amount  
 from Purchase,BookData  
 where BookData.BookISBN = Purchase.BookISBN  
 and PurchaseID like '%'+@PurchaseID+'%'  
 end  
*--endregion  
  
--region 10.根据图书号码获得某一图书具体信息*Go  
CREATE PROCEDURE getBookSellDataByBookISBN  
 @BookISBN varchar(13)  
as  
 begin  
 select \*  
 from BookView  
 where BookISBN like '%'+@BookISBN+'%'  
 end  
*--endregion  
  
--region 11.根据图书名获得匹配到的所有图书*Go  
CREATE PROCEDURE getBookDataByBookName  
 @BookName nvarchar(25)  
as  
 begin  
 select \*  
 from BookView  
 where BookName like '%'+@BookName+'%'  
 end  
*--endregion  
  
--region 12.根据作者名获得匹配到的所有图书*Go  
CREATE PROCEDURE getBookDataByAuthor  
 @Author nvarchar(15)  
as  
 begin  
 select \*  
 from BookView  
 where BookAuthor like '%'+@Author+'%'  
 end  
*--endregion  
  
--region 13.根据类别获得匹配到的所有图书*Go  
CREATE PROCEDURE getBookDataByCategory  
 @Category nvarchar(10)  
as  
 begin  
 select \*  
 from BookView  
 where BookCategory = @Category  
 end  
*--endregion  
  
--region 14.根据会员名获得匹配到的所有会员*Go  
CREATE PROCEDURE getMemberDataByMemberName  
 @memberName nvarchar(10)  
as  
 begin  
 select \*  
 from memberData  
 where memberName like '%'+@memberName+'%'  
 end  
*--endregion  
  
--region 15.根据会员手机号匹配会员*Go  
CREATE PROCEDURE getMemberDataByPhoneNumber  
 @phoneNumber char(11)  
as  
 begin  
 select \*  
 from memberData  
 where memberPhoneNumber = @phoneNumber  
 end  
*--endregion  
  
--region 16.根据会员ID匹配会员*Go  
CREATE PROCEDURE getMemberDataByMemberID  
 @memberID varchar(8)  
as  
 begin  
 select \*  
 from memberData  
 where memberID like '%'+@memberID+'%'  
 end  
*--endregion  
  
--region 17.根据图书名和作者名搜索图书*Go  
CREATE PROCEDURE getBookDataByBookNameAndAuthor  
 @BookName nvarchar(25),  
 @Author nvarchar(15)  
as  
 begin  
 select \*  
 from BookView  
 where BookName like '%'+@BookName+'%'  
 or @Author like '%'+@Author+'%'  
 end  
*--endregion  
  
--region 18.根据图书名和类别搜索图书*Go  
CREATE PROCEDURE getBookDataByBookNameAndCategory  
 @BookName nvarchar(25),  
 @Category nvarchar(10)  
as  
 begin  
 select \*  
 from BookView  
 where BookName like '%'+@BookName+'%'  
 and BookCategory = @Category  
 end  
*--endregion  
  
--region 19.根据作者名和类别搜索图书*Go  
CREATE PROCEDURE getBookDataByAuthorAndCategory  
 @Author nvarchar(15),  
 @Category nvarchar(10)  
as  
 begin  
 select \*  
 from BookView  
 where BookAuthor like '%'+@Author+'%'  
 and BookCategory = @Category  
 end  
*--endregion  
  
--region 20.根据图书名、作者名和类别搜索图书*Go  
CREATE PROCEDURE getBookDataByBookName\_AuthorAndCategory  
 @BookName nvarchar(25),  
 @Author nvarchar(15),  
 @Category nvarchar(10)  
as  
 begin  
 select \*  
 from BookView  
 where BookName like '%'+@BookName+'%'  
 or BookAuthor like '%'+@Author+'%'  
 and BookCategory = @Category  
 end  
*--endregion  
  
--region 21.修改某一书籍用于租借的数量存储过程*Go  
CREATE PROCEDURE AlterLeaseCountByBookISBN  
 @BookISBN char(13),  
 @Count int  
as  
begin  
 declare @NowCount int  
 declare @BookID char(7)  
 select @NowCount = count(\*) from BookItemsView where BookISBN = @BookISBN  
 if @Count < @NowCount  
 begin  
 declare mycursor cursor scroll for select BookID from BookItemsView where BookISBN = @BookISBN  
 open mycursor  
 fetch last from mycursor into @BookID  
 while @Count < @NowCount and @@FETCH\_STATUS = 0  
 begin  
 delete from Items where BookID = @BookID  
 set @NowCount = @NowCount - 1  
 fetch PRIOR from mycursor into @BookID  
 end  
 close mycursor  
 deallocate mycursor  
 end  
 else  
 begin  
 update BookData set BrandNewBookNum -= @Count - @NowCount where BookISBN = @BookISBN  
 while @NowCount > @Count  
 begin  
 set @BookID = dbo.getNextID(@BookISBN)  
 insert into Items(BookID,BookISBN)  
 Values (@BookID,@BookISBN)  
 end  
 end  
end  
*--endregion  
  
--endregionbian*

1. **总结**

此项目是参考我们家附近的一个书店经营模式所开发的，选择这个项目也是因为这个系统的基本流程还是比较清晰的，数据库设计和预期目标较为符合，可以实现书店基本的需求。因为这个系统是服务于书店管理员的，所以在设计时只考虑书店管理员，还是比较方便的，不过在设计整个数据库的过程中还是发现了很多自己的不足。

在刚开始设计数据库时候会发现自己考虑的不是很周到，经常会发生在写存储过程时候意识到，我某个表里还缺点东西，所以还得更改表并且更改其他相关的东西。并且在完整性和安全性控制方面考虑的也不够周到，不过经过这次数据库的设计，之后再进行数据库设计的话应该会省力很多，这些应该注意的点之后也都会注意到。

这个系统还是存在着一定的问题，首先功能还是比较少的，对于会员方面还可以加入评级功能，根据消费金额、充值金额、租借图书数量和注册时长等属性来考量，然后根据会员的等级可以在租借图书时给予不同的折扣；另外表的结构和一些主码的设计还有一些不太合理的地方，但是有些去修改的话我还是感觉有些无从下手。

这个数据库的设计还是让我收获了很多，了解到了更多更深层次的数据库知识，知道了Sql Server的image是二进制数据类型，也知道了图片存储可以采用直接存图片和存图片的路径，同时在从网上查询资料的时候也学习到了很多知识，比如说后端调用数据库的存储过程如何得到存储过程中查询的结果，后端如何调用数据库的函数等，也理解了函数和存储过程存在的必要性。

1. **参考资料**
2. SQL Server官方文档：<https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-2017>
3. W3schoolSQL教程：<https://www.w3school.com.cn/sql/index.asp>
4. StackOverflow回答：<https://stackoverflow.com/questions/13373168/sql-server-after-insert-for-insert-rollback/13374004#13374004>
5. 博客园关于Sql Server image类型的讲解博客：<https://www.cnblogs.com/samcn/articles/1275072.html>
6. **致谢**

在此感谢老师提供此机会让我了解到了数据库对图片存储的具体方式，且对触发器、存储过程、函数等有了进一步的认识，认识到了在项目中视图的重要性。同时也要感谢网络上积极撰写博客和回答问题分享知识的开发者，在数据库的构建中通过网络学习到了很多自己不知道的知识，也进一步了解到了数据库存储过程、触发器函数等和后端语言的交互途径。