# 图书借阅管理系统

[图书借阅管理系统](#图书借阅管理系统)  
 [需求分析](#需求分析)  
 [概念设计](#概念设计)  
 [逻辑结构设计](#逻辑结构设计)  
 [物理结构设计](#物理结构设计)  
 [SQL 语句](#sql-语句)  
 [创建数据库](#创建数据库)  
 [插入数据](#插入数据)  
 [存储过程设置：](#存储过程设置)  
 [触发器设置](#触发器设置)

## 需求分析

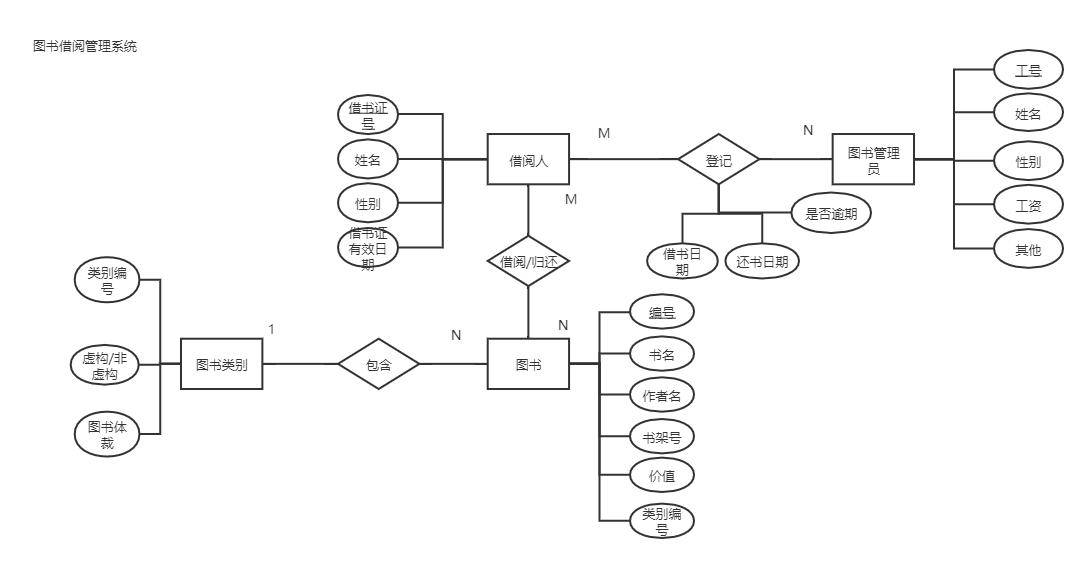
图书借阅系统应对借阅者的信息与被借阅图书信息一起保留。借阅人对图书进行借阅，同时由图书管理员进行对借书、还书日期的登记，并查看是否逾期，收取相关费用。

借阅图书的分类由图书类别通过编号进行管理。

故图书借阅系统共需要四个数据项，分别是：  
借阅人：借书证号、姓名、性别、借书证有效日期。  
图书：图书编号、书名、作者名、书架号、价值。  
管理员：工号、姓名、性别、工资、其他。  
图书类别：类别编号、虚构类/非虚构类、图书体裁。

## 概念设计

ER图：



## 逻辑结构设计

实体：图书类别、图书、借阅人、管理人

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 |  |  | 图书管理员值班表 |  |
| 列名 | 数据类型与宽度 | 是主键 | 空/非空 | 约束条件 |
| 工号 | int | 是 | 非空 | 无 |
| 姓名 | varchar(4) | 否 | 非空 | 无 |
| 性别 | char（1） | 否 | 非空 | 男或女 |
| 工资 | int | 否 | 非空 | 无 |
| 其他 | varchar(20) | 否 | 是 | 无 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 |  |  | 借阅人管理表 |  |
| 列名 | 数据类型与宽度 | 是主键 | 空/非空 | 约束条件 |
| 借书证号 | int | 是 | 非空 | 无 |
| 姓名 | varchar(4) | 否 | 非空 | 无 |
| 性别 | char（1） | 否 | 非空 | 男或女 |
| 借书证有效期 | datetime | 否 | 非空 | 无 |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 |  |  | 图书管理表 |  |
| 列名 | 数据类型与宽度 | 是主键 | 空/非空 | 约束条件 |
| 图书编号 | int | 是 | 非空 | 无 |
| 书名 | varchar(10) | 否 | 非空 | 无 |
| 作者名 | varchar(4) | 否 | 非空 | 无 |
| 书架号 | int | 否 | 空 | 无 |
| 价值 | int | 否 | 非空 | 无 |
| 图书类别（编号） | int | 否 | 非空 | 从management中获得 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 |  |  | 图书类别管理表 |  |
| 列名 | 数据类型与宽度 | 是主键 | 空/非空 | 约束条件 |
| 类别编号 | int | 是 | 非空 | 无 |
| 是否为虚构类 | varchar(1) | 否 | 非空 | 是/否 |
| 图书体裁 | varchar(4) | 否 | 非空 | "科幻"/"社科"/"专业类"/"纪实"/ "文学" |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 |  |  | 借阅登记表 |  |
| 列名 | 数据类型与宽度 | 是主键 | 空/非空 | 约束条件 |
| 借阅日期 | datetime | 是 | 非空 | 无 |
| 归还日期 | datetime | 否 | 空 | 无 |
| 是否逾期 | char(1) | 否 | 空 | 是/否 |
| 图书编号 | int | 否 | 非空 | 从图书管理表中获取 |
| 借阅人图书证号 | int | 否 | 非空 | 从借阅人管理表中获取 |
| 管理人工号 | int | 否 | 非空 | 从管理人值班表中获取 |

## 物理结构设计

由于图书借阅系统中借阅登记表更新频繁，且需要外键与auto\_increment，故使用InnoDB存储引擎。在建立逐渐同时建立聚集索引。

为了存储上的便利与对数据库完整性的维护，首先对借、还分别定义存储过程regist\_rent和regist\_return，分别以当前时间作为借书时间记录和还书时间记录，这与借阅系统在实际操作中的应用相同。当运行借操作时，记录当前借阅时间，存入表中，并使还书、是否逾期两列为空，等到还书时根据输入的借书时间，更新还书时间和是否逾期的两列。

在表registeration上定义触发器registeration\_after\_insert，在插入操作后执行，对插入操作的合理性进行验证，假如借书的时间超过了借阅证的有效时期，则应拒绝记录的插入并报错（由于现实中即使借书证失效也应当还书，故不为还书存储过程设置类似触发器功能）

|  |  |  |
| --- | --- | --- |
| 表名 | 列名 | 索引 |
| 图书管理员值班表 | 工号 | 聚集索引 |
| 借阅人管理表 | 借书证号 | 聚集索引 |
| 图书管理表 | 图书编号 | 聚集索引 |
| 图书类别管理表 | 类别编号 | 聚集索引 |

## SQL 语句

### 创建数据库

CREATE DATABASE Library;
  
use Library;
  
CREATE TABLE administer(
  
 a\_workid INT PRIMARY KEY NOT NULL auto\_increment,
  
 a\_name VARCHAR(4) NOT NULL,
  
 a\_gender ENUM("男","女") NOT NULL,
  
 a\_salary INT NOT NULL,
  
 a\_else VARCHAR(20)
  
);
  
   
CREATE TABLE reader(
  
 r\_license INT PRIMARY KEY NOT NULL auto\_increment,
  
 r\_name VARCHAR(4) NOT NULL,
  
 r\_gender ENUM("男","女") NOT NULL,
  
 r\_able DATETIME NOT NULL
  
);
  
   
CREATE TABLE management(
  
 m\_id INT PRIMARY KEY NOT NULL auto\_increment,
  
 id\_ifvirtual ENUM("是","否") NOT NULL,
  
 style ENUM("科幻","社科","专业类","纪实", "文学") NOT NULL
  
);
  
   
CREATE TABLE book(
  
 b\_id INT PRIMARY KEY NOT NULL auto\_increment,
  
 b\_name VARCHAR(10) NOT NULL,
  
 b\_author VARCHAR(4) NOT NULL,
  
 b\_shelf INT,
  
 b\_value INT NOT NULL,
  
 b\_mid INT NOT NULL,
  
 FOREIGN KEY(b\_mid) REFERENCES management(m\_id)
  
);
  
   
CREATE TABLE registeration(
  
 re\_rent DATETIME NOT NULL,
  
 re\_return DATETIME,
  
 re\_workid INT NOT NULL,
  
 re\_license INT NOT NULL,
  
 re\_bookid INT NOT NULL,
  
 PRIMARY KEY(re\_rent, re\_return),
  
 re\_overtime ENUM("是","否"),
  
 FOREIGN KEY (re\_bookid) REFERENCES book(b\_id),
  
 FOREIGN KEY (re\_license) REFERENCES reader(r\_license),
  
 FOREIGN KEY (re\_workid) REFERENCES administer(a\_workid)
  
);
  
ALTER TABLE registeration DROP PRIMARY KEY;
  
ALTER TABLE registeration ADD constraint PRIMARY KEY (re\_rent);
  
ALTER TABLE registeration modify column re\_return DATETIME;
  
ALTER TABLE book modify column b\_shelf INT;
  
-- drop database library;
  
show engines;

### 插入数据

use Library;
  
select \* from administer;
  
-- 1 填充管理员表
  
INSERT administer(
  
 a\_name,
  
 a\_gender,
  
 a\_salary,
  
 a\_else
  
)VALUE(
  
 "员工甲",
  
 "男",
  
 7000,
  
 NULL
  
);
  
   
INSERT administer(
  
 a\_name,
  
 a\_gender,
  
 a\_salary,
  
 a\_else
  
)VALUE(
  
 "员工乙",
  
 "男",
  
 5000,
  
 NULL
  
);
  
   
INSERT administer(
  
 a\_name,
  
 a\_gender,
  
 a\_salary,
  
 a\_else
  
)VALUE(
  
 "员工丙",
  
 "男",
  
 6000,
  
 NULL
  
);
  
   
INSERT administer(
  
 a\_name,
  
 a\_gender,
  
 a\_salary,
  
 a\_else
  
)VALUE(
  
 "员工丁",
  
 "男",
  
 8000,
  
 NULL
  
);
  
-- 填充management表
  
SELECT \* FROM management;
  
   
INSERT management(
  
 id\_ifvirtual,
  
 style
  
)VALUE(
  
 "是",
  
 "文学"
  
);
  
   
INSERT management(
  
 id\_ifvirtual,
  
 style
  
)VALUE(
  
 "是",
  
 "科幻"
  
);
  
   
INSERT management(
  
 id\_ifvirtual,
  
 style
  
)VALUE(
  
 "否",
  
 "社科"
  
);
  
   
INSERT management(
  
 id\_ifvirtual,
  
 style
  
)VALUE(
  
 "否",
  
 "专业类"
  
);
  
   
INSERT management(
  
 id\_ifvirtual,
  
 style
  
)VALUE(
  
 "否",
  
 "纪实"
  
);
  
   
-- 填充book表
  
SELECT \* FROM book;
  
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "荒野侦探",
  
 "波拉尼奥",
  
 1,
  
 40,
  
 1
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "箱男",
  
 "安部公房",
  
 1,
  
 18,
  
 1
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "醉步男",
  
 "小林泰三",
  
 2,
  
 45,
  
 2
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "黑暗中的笑声",
  
 "纳博科夫",
  
 1,
  
 50,
  
 1
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "生命是什么",
  
 "薛定谔",
  
 4,
  
 23,
  
 4
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "光学",
  
 "赵光华",
  
 4,
  
 20,
  
 4
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "日俄战争灾难纪实",
  
 "关捷",
  
 5,
  
 60,
  
 5
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "剧本",
  
 "罗伯特",
  
 3,
  
 40,
  
 3
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "草枕",
  
 "夏目漱石",
  
 1,
  
 25,
  
 1
  
);
  
   
INSERT book(
  
 b\_name,
  
 b\_author,
  
 b\_shelf,
  
 b\_value,
  
 b\_mid
  
)VALUE(
  
 "自控力",
  
 "凯莉",
  
 4,
  
 30,
  
 4
  
);
  
   
-- 填充 reader表
  
SELECT \* FROM READER;
  
INSERT reader(
  
 r\_name,
  
 r\_gender,
  
 r\_able
  
)VALUE(
  
 "俾斯麦",
  
 "男",
  
 "2021-06-30"
  
);
  
   
INSERT reader(
  
 r\_name,
  
 r\_gender,
  
 r\_able
  
)VALUE(
  
 "冉阿让",
  
 "男",
  
 "2021-06-29"
  
);
  
   
INSERT reader(
  
 r\_name,
  
 r\_gender,
  
 r\_able
  
)VALUE(
  
 "伊丽莎",
  
 "女",
  
 "2021-06-28"
  
);
  
   
INSERT reader(
  
 r\_name,
  
 r\_gender,
  
 r\_able
  
)VALUE(
  
 "卞福汝",
  
 "男",
  
 "2021-07-30"
  
);
  
   
INSERT reader(
  
 r\_name,
  
 r\_gender,
  
 r\_able
  
)VALUE(
  
 "堂吉诃德",
  
 "男",
  
 "2021-08-30"
  
);
  
   
   
-- 填充registeration表
  
select \* from registeration;
  
   
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2011-01-22",
  
 "2018-02-24",
  
 1,
  
 1,
  
 1,
  
 "是"
  
);
  
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2011-01-23",
  
 "2011-02-24",
  
 2,
  
 3,
  
 7,
  
 "否"
  
);
  
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2011-04-23",
  
 "2012-01-01",
  
 2,
  
 4,
  
 5,
  
 "否"
  
);
  
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2017-12-20",
  
 "2018-03-31",
  
 2,
  
 3,
  
 3,
  
 "否"
  
);
  
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2018-11-03",
  
 "2021-03-12",
  
 3,
  
 2,
  
 9,
  
 "是"
  
);
  
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2011-02-23",
  
 "2019-02-24",
  
 4,
  
 5,
  
 8,
  
 "是"
  
);

### 存储过程设置：

show columns from administer;
  
SELECT \* FROM administer;
  
SELECT \* FROM book;
  
SELECT \* FROM management;
  
SELECT \* FROM reader;
  
SELECT \* FROM registeration;
  
-- 建立存储过程 输入参数 借书人id，所借书籍id，管理员id，随后在登记系统中登记借阅信息
  
-- 借操作 (假设以当前时间作为登记借阅时间)
  
delimiter $$
  
DROP PROCEDURE IF EXISTS `regist\_rent`$$
  
CREATE PROCEDURE regist\_rent(
  
 IN bookid INT,
  
 IN licenseid INT,
  
 IN administerid INT)
  
BEGIN
  
 INSERT INTO registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
 )VALUE(
  
 NOW(),
  
 null,
  
 (SELECT a\_workid FROM administer WHERE a\_workid = administerid),
  
 (SELECT r\_license FROM reader WHERE r\_license = licenseid),
  
 (SELECT b\_id FROM book WHERE b\_id = bookid),
  
 null
  
 );
  
 SELECT \* FROM registeration;
  
END$$
  
delimiter ;
  
   
call regist\_rent(1, 2, 1);
  
-- 还操作 存储过程 以当前时间为归还时间 输入借的日期 并存储过程中判断是否逾期（假设超过8个月算逾期）并输出
  
delimiter $$
  
DROP PROCEDURE IF EXISTS `regist\_return`$$
  
CREATE PROCEDURE `regist\_return`(
  
 IN rent\_time DATETIME,
  
 OUT return\_time DATETIME)
  
BEGIN
  
 DECLARE ISOVERTIME ENUM("是", "否");
  
 DECLARE OVTIME INT;
  
 DECLARE RENTTIME DATETIME;
  
 SELECT re\_rent FROM registeration WHERE re\_rent = rent\_time INTO RENTTIME;
  
 SET OVTIME = TIMESTAMPDIFF(MONTH, RENTTIME, NOW());
  
 UPDATE registeration SET re\_return = NOW() WHERE re\_rent = RENTTIME;
  
 SELECT IF(OVTIME < 9, '是', '否') INTO ISOVERTIME;
  
 UPDATE registeration SET re\_overtime = ISOVERTIME WHERE re\_rent = RENTTIME;
  
 SELECT \* FROM registeration;
  
 SELECT NOW() INTO return\_time;
  
END$$
  
delimiter ;
  
   
CALL regist\_return('2021-06-15 01:43:25', @return\_time);
  
SELECT @return\_time AS "归还时间";
  
CALL regist\_return('2021-06-15 03:06:08', @return\_time);
  
SELECT @return\_time AS "归还时间";

### 触发器设置

-- triggers 每当进行一次借阅将借阅时间与借阅人有效日期进行对比，若借阅日期超过有效日期，则插入无效并报错
  
SELECT \* FROM registeration;
  
SELECT \* FROM reader;
  
delimiter $$
  
DROP TRIGGER IF EXISTS `library`.`registeration\_after\_insert` $$
  
CREATE TRIGGER `library`.`registeration\_after\_insert` AFTER INSERT ON `registeration`
  
FOR EACH ROW
  
BEGIN
  
 IF ((SELECT r\_able FROM reader WHERE r\_license = new.re\_license) < new.re\_rent) THEN
  
 BEGIN
  
 -- 不在有效期内
  
 signal sqlstate '45000' set message\_text='借阅人借阅证过期';
  
 END;
  
 END IF;
  
END$$
  
delimiter ;
  
   
INSERT registeration(
  
 re\_rent,
  
 re\_return,
  
 re\_workid,
  
 re\_license,
  
 re\_bookid,
  
 re\_overtime
  
)VALUE(
  
 "2022-02-23",
  
 NULL,
  
 4,
  
 5,
  
 8,
  
 "是"
  
);