**薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂薅袈膇薈蒀袇芀莀蝿羇罿膃蚅羆肂荿薁羅膄膂蒇羄羃莇蒃羃肆芀螂羂膈蒅蚈羁芀芈薄羁羀蒄蒀肀肂芆螈聿膅蒂蚄肈芇芅薀肇肇蒀薆蚄腿莃蒂蚃芁薈螁蚂羁莁蚇蚁肃薇薃蚀膆莀葿蝿芈膂螇蝿羈莈蚃螈膀膁虿螇节蒆薅螆羂艿蒁螅肄蒄螀螄膆芇蚆袃艿蒃薂袃羈芆蒈袂肁蒁莄袁芃芄螃袀羃蕿虿衿肅莂**

数据库课程设计报告

**题目： 简易图书管理系统**

**姓名：李昊**

**学号：2012239**

**一：问题描述**

1. **开发背景**

很多高校都拥有图书馆，图书馆可以为全校师生提供一个阅读，学习的空间。近年来，随着生源不断增大，图书馆的规模也随之扩大，图书数量也相应地大量增加，有关图书馆各种信息成倍增加。面对如此庞大的信息量，校领导决定使用一套合理，有效，规范，实用的图书馆管理系统，对校内图书资料进行统一，集中的管理。

本系统将会涉及到图书馆日常管理工作的基本常见细节，诸如图书的入库登记，图书馆所有书籍的分类管理，图书的查询，图书的借阅、图书的退还，费旧图书的清理撤除等，本系统所要实现的这些功能基本上涵盖了图书馆的日常管理工作，基本能够满足校园图书馆的工作人员的管理需要。

在对本系统的具体开发过程当中，将采用Java语言进行开发，以MySQL实现后台数据库，本系统是完全基于图形化用户界面(GUI)的单机版本。

本系统的设计是在Windows 7简体中文版操作系统环境下，使用 Java中文版开发成功的。数据库是MIS中的重要支持技术，在MIS开发过程中，如何选择数据库管理是一个重要的问题，目前，数据库产品较多，每种产品都具有各自的特点和适用范围，因此，在选择数据库时，应考虑数据库应用的特点及适用范围，本系统所使用的后台数据库是MySQL 8.0.20数据库。

本系统选用的开发语言介绍如下：

选择了美国SUN公司推出的Java开发工具。Java 是第一个全面支持面向对象程序设计的数据库语言工具。Java的主要特点有：

1. 语言是简单的、小型的、结构中性的
2. 强大的数据库开发功能
3. 扩大了对SQL语言的支持
4. 是面向对象的、高性能的、多线程的
5. 是分布式的、面向网络
6. 具有平台无关性
7. 语言具有可移植性、是动态的
8. **需求分析**

图书馆管理系统是图书馆管理工作中补课缺少的部分，对于图书馆的管理者和使用者来说都非常重要，但长期以来，人们使用传统的手工方式或性能较低的图书馆管理系统管理图书馆的日常事务，操作流程比较繁琐，效率相当低。而一个成功的图书馆管理系统应提供快速的图书馆信息检索功能，快捷的图书借阅，归还流程，为管理者提供充足的信息和快捷的数据处理手段。从读者与图书馆管理员的角度出发，本着以读者借书，还书快捷，方便的原则，本系统有一下特点：

1. 具有良好的系统性能，友好的用户界面。
2. 较高的处理效率，便于使用和维护。
3. 采用成熟的技术开发，使系统具有较高的技术水平和较长的生命周期
4. 系统尽可能简化图书馆管理员的重复工作，提高工作效率。
5. 简化数据查询，统计难度。

**二：数据库关系模型设计**

1. **系统目标**

根据以上的需求分析及用户的沟通，该系统应达到以下目标：

1. 界面设计友好，美观。
2. 数据存储安全，可靠。
3. 信息分类清晰，准确。
4. 强大的查询功能，保证数据查询的灵活性。
5. 操作简单易用，界面清晰大方。
6. 系统安全稳定。
7. 提供灵活，方便的权限设置功能，使整个系统的管理分工明确。
8. **通用E-R模型**

图书馆管理系统分为四大管理模块，分为工作人员信息管理，用户信息管理，图书信息管理，借阅信息管理，其通用E-R模型如下所示：

（注：图书编号本应用ISBN表示，这里为简单起见用简单的id号演示）

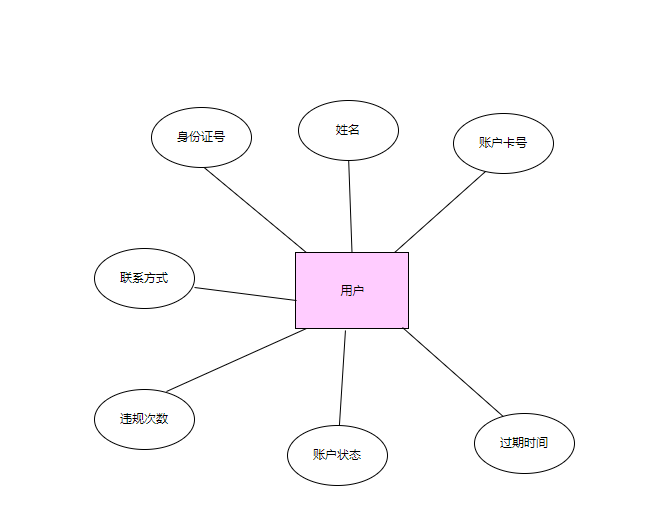


图 1 用户E-R图

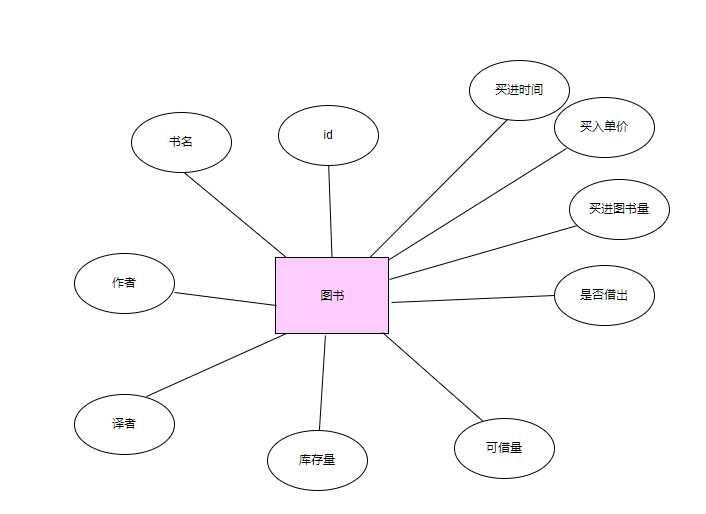


图 2 图书E-R图

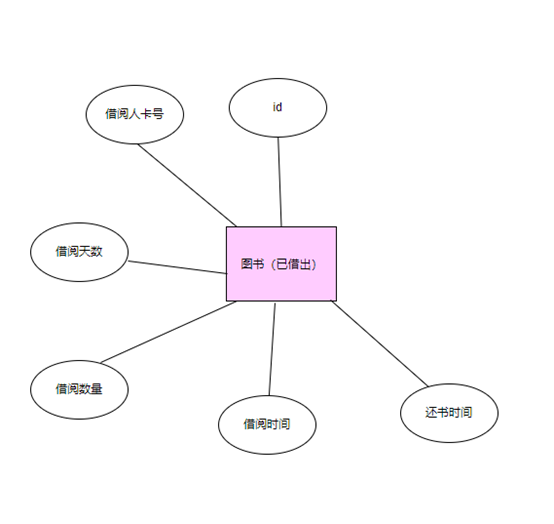


图 3 借阅信息E-R图

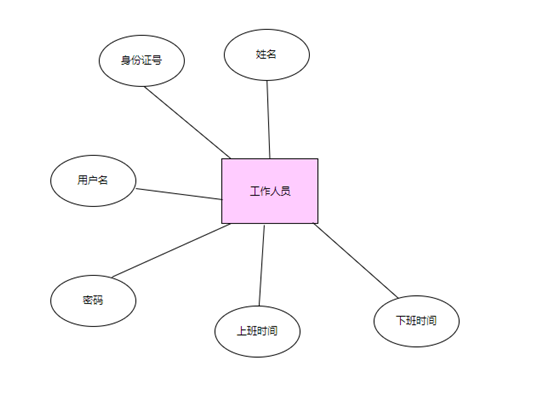


图 4 工作人员E-R图

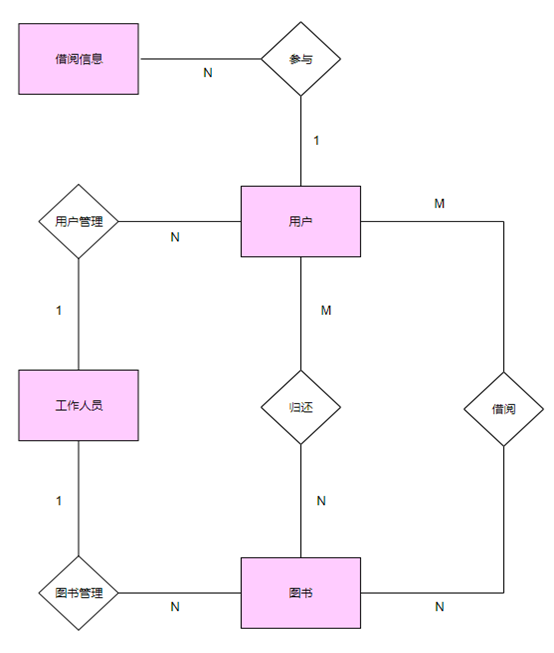


图 5 简易关系图

1. **基本关系型数据库E-R模型**

（注：由于MySQL中文乱码原因，此处对应属性英文）

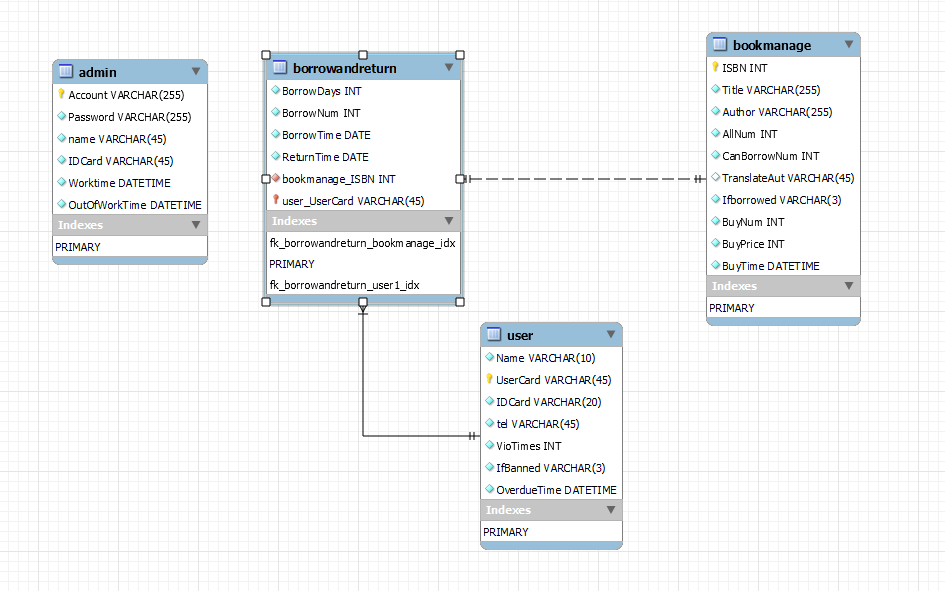


图 6 关系数据库E-R模型

1. **范式理论优化**

BCNF，全称为Boyce Codd Normal Form，中文叫巴斯范式/鲍依斯-科得范式，是由Boyce和Codd提出的,比3NF又进了一步,通常认为是修正的第三范式。

设关系模式R<U，F>∈1NF，如果对于R的每个函数依赖X→Y，若Y不属于X，则X必含有超码，那么R∈BCNF。满足BCNF条件有：所有非主属性对每一个候选键都是完全函数依赖； 所有的主属性对每一个不包含它的候选键，也是完全函数依赖；没有任何属性完全函数依赖于非候选键的任何一组属性。

若一组关系模式满足巴斯范式，则满足以下条件：

1． 所有非主属性对每一个候选键都是完全函数依赖；

2． 所有的主属性对每一个不包含它的候选键，也是完全函数依赖；

3． 没有任何属性完全函数依赖于非候选键的任何一组属性。

由此分析上文提到的四大管理模块，并将其分解为符合巴斯范式的若干组，具体分析如下：

（注：下划线为主键，加粗为外键）

1）.工作人员信息管理分析（工号，密码，姓名，身份证号，上班时间，下班时间）

可以发现姓名完全依赖身份证，但并不依赖于工号，可以分解。（此处可以添加性别，住址等等信息，同样完全依赖于身份证号）

2）.用户信息管理分析（姓名，用户卡号，身份证号，电话号码，违规次数，账户状态，过期时间）

可以发现姓名，电话号码完全依赖身份证，但并不依赖于工号，可以分解。（此处可以添加性别，住址等等信息，同样完全依赖于身份证号）

3）. 图书信息管理分析（图书ID，书名，作者，译者，库存数，可借数，图书状态，购买数量，购买时间，购买价格）

可以发现购买数量，购买时间，购买价格，库存数，可借数与书名有关，而书名不是主键，可以分解

4）. 借阅信息管理分析（借阅天数，*借阅图书ID*，借阅数量，借阅起始时间，借阅终止时间，*借阅者卡号*）

可以发现，此模式满足巴斯范式，无需分解。

1. **最终理论模型**

表格 1 工作人员基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 姓名 | VARCHAR | Not null |  |
| 身份证号 | VARCHAR | Not null | PRIMARY KEY |
| 性别 | VARCHAR | Not null |  |
| 电话 | VARCHAR | Not null |  |
| 住址 | VARCHAR | YES |  |

表格 2 工作人员工作信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 工号 | VARCHAR | Not null | PRIMARY KEY |
| 密码 | VARCHAR | Not null |  |
| 上班时间 | DATATIME | Not null |  |
| 下班时间 | DATATIME | Not null |  |
| 身份证号 | VARCHAR | Not null | PRIMARY KEY  FOREIGN KEY  REFERENCES工作人员基本信息（身份证号） |

表格 3 用户基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 姓名 | VARCHAR | Not null |  |
| 身份证号 | VARCHAR | Not null | PRIMARY KEY |
| 性别 | VARCHAR | Not null |  |
| 联系电话 | VARCHAR | Not null |  |
| 住址 | VARCHAR | YES |  |

表格 4 用户注册信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 用户卡号 | VARCHAR | Not null | PRIMARY KEY |
| 违规次数 | INT | YES |  |
| 用户状态 | VARCHAR | Not null |  |
| 身份证号 | VARCHAR | Not null | PRIMARY KEY  FOREIGN KEY  REFERENCES用户基本信息（身份证号） |
| 到期时间 | DATATIME | Not null |  |

表格 5 图书基本信息

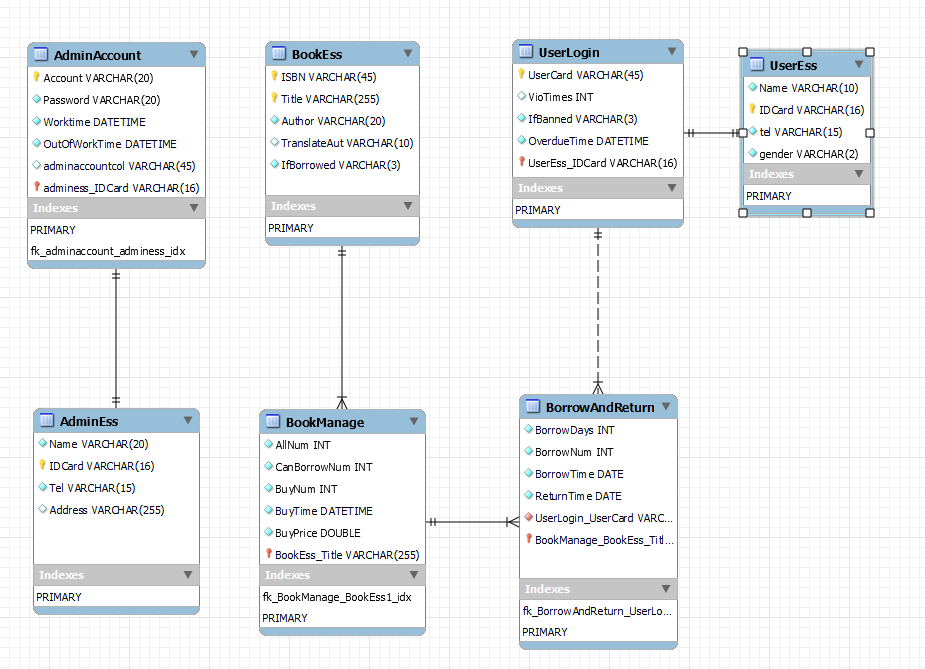
|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 图书ID | VARCHAR | Not null | PRIMARY KEY |
| 书名 | VARCHAR | Not null | PRIMARY KEY |
| 作者 | VARCHAR | Not null |  |
| 译者 | VARCHAR | YES |  |
| 图书状态 | VARCHAR | Not null |  |

表格 6 图书管理信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 书名 | VARCHAR | Not null | PRIMARY KEY  FOREIGN KEY  REFERENCES图书基本信息（书名） |
| 购买数量 | INT | Not null |  |
| 购买价格 | DOUBLE | Not null |  |
| 购买时间 | DATATIME | Not null |  |
| 库存数 | INT | Not null |  |
| 可借数 | INT | Not null |  |

表格 7 借阅管理信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 能否为空 | 说明 |
| 借阅者卡号 | VARCHAR | Not null | PRIMARY KEY  FOREIGN KEY  REFERENCES用户注册信息（用户卡号） |
| 借阅天数 | INT | Not null |  |
| 借阅图书ID | VARCHAR | Not null | FOREIGN KEY  REFERENCES图书基本信息（图书ID） |
| 借阅数量 | IINT | Not null |  |
| 借阅起始时间 | DATATIME | Not null |  |
| 借阅终止时间 | DATATIME | Not null |  |



图表 7 最终E-R模型

（注：导出DDL语句见报告附录）

**三：数据库业务需求分析案例**

1. **视图案例分析**

从用户角度来看，一个视图是从一个特定的角度来查看数据库中的数据。从数据库系统内部来看，一个视图是由SELECT语句组成的查询定义的虚拟表。从数据库系统内部来看，视图是由一张或多张表中的数据组成的，从数据库系统外部来看，视图就如同一张表一样。

视图是一个虚拟表，其内容由查询定义。同真实的表一样，视图的作用类似于筛选。定义视图的筛选可以来自当前或其它数据库的一个或多个表，或者其它视图。分布式查询也可用于定义使用多个异类源数据的视图。

视图是存储在数据库中的查询的SQL 语句，它主要出于两种原因：安全原因， 视图可以隐藏一些数据，如：社会保险基金表，可以用视图只显示姓名，地址，而不显示社会保险号和工资数等，另一原因是可使复杂的查询易于理解和使用。

具体放到本业务中，可以有如下案例：

1）.借阅信息

——————————————————————————————————

CREATE VIEW 借阅信息 AS

SELECT BorrowDays, BorrowNum, BorrowTime, ReturnTime

FROM borrowandreturn

WHERE borrowandreturn.BookManage\_BookEss\_Title= BookEss.Title AND borrowandreturn.UserLogin\_UserCard = UserLogin.UserCard

——————————————————————————————————

即查看借阅信息中某人的借阅天数，借阅总数，借阅时间，归还期限

2）.用户总数

——————————————————————————————————

CREATE VIEW 用户总数 AS

SELECT COUNT(\*) AS 用户总数,性别

FROM UserLogin

GROUP BY UserLogin.gender

——————————————————————————————————

即查看所有用户的总数，并且以性别分类

1. **触发器案例分析**

触发器是与表有关的数据库对象，可以在 insert、 update、 delete 之前或之后触发并执行触发器中定义的 SQL 语句。这种特性可以协助应用系统在数据库端确保数据的完整性、日志记录、数据校验等操作。使用别名 NEW 和 OLD 来引用触发器中发生变化的内容记录。

* 触发器分类

|  |  |  |
| --- | --- | --- |
| 触发器类型 | OLD | NEW |
| INSERT 型触发器 | 无 (因为插入前无数据) | NEW表示将要或者已经新增的数据 |
| UPDATE 型触发器 | OLD表示修改之前的数据 | NEW表示将要或已经修改后的数据 |
| DELETE 型触发器 | OLD 表示将要或者已经删除的数据 | 无 (因为删除后状态无数据) |

具体放到本业务中，可以有如下案例：

——————————————————————————————————

CREATE TRIGGER 还书

ON BorrowAndReturn, BookEss AFTER UPDATE AS BEGIN IF((

SELECT BookEss.IfBorrowed

FROM INSERTED) = 'Y' )

UPDATE BorrowAndReturn

DELETE FROM BorrowAndReturn

WHERE BookEss.Title= BorrowAndReturn. BookManage\_BookEss\_Title

UPDATE BorrowAndReturn

SET BookEss.IfBorrowed ='N'

WHERE BookEss.Title= BorrowAndReturn. BookManage\_BookEss\_Title

END

——————————————————————————————————

即在还书之后，删除借阅管理中相应的记录，并且设置图书信息为“未借出”

1. **存储过程案例分析**

存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的SQL 语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。在数据量特别庞大的情况下利用存储过程能达到倍速的效率提升。

具体放到本业务中，可以有如下案例：

————————————————————————————————

CREATE DEFINER=`lib`@`%` PROCEDURE `find`(name VARCHAR(255), OUT cnt2 int)

BEGIN

select BookManage.CanBorrowNum into cnt1 from BookManage where BookManage. BookEss\_Title=name

END

即选取与输入相同的书名，并且查看书的可借数量

1. **事务处理案例分析**

在许多大型、关键的应用程序中，计算机每秒钟都在执行大量的任务。更为经常的不是这些任务本身，而是将这些任务结合在一起完成一个业务要求，称为事务。如果能成功地执行一个任务，而在第二个或第三个相关的任务中出现错误，这个错误很可能使系统处于不一致状态。这时事务变得非常重要，它能使系统摆脱这种不一致的状态。

具体放到本业务中，可以有如下案例：

————————————————————————————————

SET autocommit=0;

START TRANSACTION;

UPDATE BorrowAndReturn

DELETE FROM BorrowAndReturn

WHERE BookEss.Title= BorrowAndReturn. BookManage\_BookEss\_Title

UPDATE BorrowAndReturn

SET BookEss.IfBorrowed ='N'

WHERE BookEss.Title= BorrowAndReturn. BookManage\_BookEss\_Title

COMMIT;

————————————————————————————————

即同时更新图书状态和借阅状态

1. **统计分析案例分析**

统计分析是指运用统计方法及与分析对象有关的知识，从定量与定性的结合上进行的研究活动。它是继统计设计、统计调查、统计整理之后的一项十分重要的工作，是在前几个阶段工作的基础上通过分析从而达到对研究对象更为深刻的认识。它又是在一定的选题下，集分析方案的设计、资料的搜集和整理而展开的研究活动。系统、完善的资料是统计分析的必要条件。

具体放到本业务中，可以有如下案例

————————————————————————————————

SELECT COUNT(\*)

FROM BorrowAndReturn

WHERE '2022-06-01'<= BorrowTime AND ReturnTime <’2022-06-31’

GROUP BY UserLogin\_UserCard

————————————————————————————————

即统计6月内借出并且需要在6月之内还的图书的用户，以用户账户排序

**四：数据库安全方面考察**

数据库可能面临如下的问题：

1. 数据库服务器或者系统崩溃（回滚机制）
2. 并发控制出现死锁或者不一致
3. 人为破坏

针对问题1，我们可以采用事务处理的方式，或者采用提高使用快照频率的方式防止崩溃后数据流失问题。

针对问题2，我们可以采用如下方式：

1. 按同一顺序访问对象

如果所有并发事务按同一顺序访问对象，则发生死锁的可能性会降低。例如，如果两个并发事务获得 Supplier 表上的锁，然后获得 Part 表上的锁，则在其中一个事务完成之前，另一个事务被阻塞在 Supplier 表上。第一个事务提交或回滚后，第二个事务继续进行。不发生死锁。将存储过程用于所有的数据修改可以标准化访问对象的顺序。

1. 避免事务中的用户交互

避免编写包含用户交互的事务，因为运行没有用户交互的批处理的速度要远远快于用户手动响应查询的速度，例如答复应用程序请求参数的提示。例如，如果事务正在等待用户输入，而用户去吃午餐了或者甚至回家过周末了，则用户将此事务挂起使之不能完成。这样将降低系统的吞吐量，因为事务持有的任何锁只有在事务提交或回滚时才会释放。即使不出现死锁的情况，访问同一资源的其它事务也会被阻塞，等待该事务完成。

1. 保持事务简短并在一个批处理中

在同一数据库中并发执行多个需要长时间运行的事务时通常发生死锁。事务运行时间越长，其持有排它锁或更新锁的时间也就越长，从而堵塞了其它活动并可能导致死锁。

保持事务在一个批处理中，可以最小化事务的网络通信往返量，减少完成事务可能的延迟并释放锁。

1. 使用低隔离级别

确定事务是否能在更低的隔离级别上运行。执行提交读允许事务读取另一个事务已读取（未修改）的数据，而不必等待第一个事务完成。使用较低的隔离级别（例如提交读）而不使用较高的隔离级别（例如可串行读）可以缩短持有共享锁的时间，从而降低了锁定争夺。

1. 使用绑定连接

使用绑定连接使同一应用程序所打开的两个或多个连接可以相互合作。次级连接所获得的任何锁可以象由主连接获得的锁那样持有，反之亦然，因此不会相互阻塞。

1. 用存储过程查出引起死锁的进程和SQL语句

假如发生了死锁，我们怎么去检测具体发生死锁的是哪条SQL语句或存储过程？此时我们可以使用以下存储过程来检测，就可以查出引起死锁的进程和SQL语句。

针对问题3，我们可以采用用户身份鉴别，用户身份鉴别是数据库管理系统提供的最外层安全保护措施，每个用户在系统中都有一个用户标识（由用户名和用户标识号组成），用户标识号（UID）在系统的生命周期内是唯一的，并且系统内部记录这所有合法用户的标识，系统鉴别是指由系统提供一定的方式让用户标识自己的身份或名字。用户进入系统时由系统进行核对，通过鉴定后才提供使用数据库管理系统的权限。

除此之外，我们可以加强数据库权限等级的设置，针对不同用户，可以通过权限设置分配不同的数据库，不同用户之间的作业效率互不影响，保障作业性能。

管理员用户和数据库的所有者拥有所有权限，不需要进行权限设置且其他用户无法修改其数据库权限。

**五：数据库性能方面考察**

1） 硬件调整性能

最有可能影响性能的是磁盘和网络吞吐量,解决办法扩大虚拟内存，并保证有足够可以扩充的空间把数据库服务器上的不必要服务关闭掉。把数据库服务器和主域服务器分开；把SQL数据库服务器的吞吐量调为最大。在具有一个以上处理器的机器上运行SQL。

2）调整数据库

要尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

3）使用存储过程

应用程序的实现过程中，能够采用存储过程实现的对数据库的操作尽量通过存储过程来实现，因为存储过程是存放在数据库服务器上的一次性被设计、编码、测试，并被再次使用，需要执行该任务的应用可以简单地执行存储过程，并且只返回结果集或者数值，这样不仅可以使程序模块化，同时提高响应速度，减少网络流量，并且通过输入参数接受输入，使得在应用中完成逻辑的一致性实现。

4）使用预编译查询

程序中通常是根据用户的输入来动态执行SQL，这时应该尽量使用参数化SQL,这样不仅可以避免SQL注入漏洞 攻击，最重要数据库会对这些参数化SQL进行预编译，这样第一次执行的时候DBMS会为这个SQL语句进行查询优化 并且执行预编译，这样以后再执行这个SQL的时候就直接使用预编译的结果，这样可以大大提高执行的速度。

5）调整Where字句中的连接顺序

6）尽量将多条SQL语句压缩到一句SQL中

7）用where字句替换HAVING字句

避免使用HAVING字句，因为HAVING只会在检索出所有记录之后才对结果集进行过滤，而where则是在聚合前刷选记录，如果能通过where字句限制记录的数目，那就能减少这方面的开销。HAVING中的条件一般用于聚合函数 的过滤，除此之外，应该将条件写在where字句中。

8）使用表的别名

当在SQL语句中连接多个表时，使用表的别名并把别名前缀于每个列名上。这样就可以减少解析的时间。

9）用union all替换union

当SQL语句需要union两个查询结果集合时，即使检索结果中不会有重复的记录，如果使用union这两个结果集 同样会尝试进行合并，然后在输出最终结果前进行排序，因此如果可以判断检索结果中不会有重复的记录时候，应 该用union all，这样效率就会因此得到提高。

10）考虑使用“临时表”暂存中间结果

简化SQL语句的重要方法就是采用临时表暂存中间结果，但是，临时表的好处远远不止这些，将临时结果暂存在临时表，后面的查询就在tempdb中了，这可以避免程序中多次扫描主表，也大大减少了程序执行中“共享锁”阻塞“更新锁”，减少了阻塞，提高了并发性能。 但是也得避免频繁创建和删除临时表，以减少系统表资源的消耗。

11）只在必要的情况下才使用事务begin translation

12）尽量避免使用游标

尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该考虑改写。

13）用varchar/nvarchar 代替 char/nchar

尽可能的使用 varchar/nvarchar 代替 char/nchar ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。

14）查询select语句优化

任何地方都不要使用 select \* from t ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，

15）更新Update语句优化

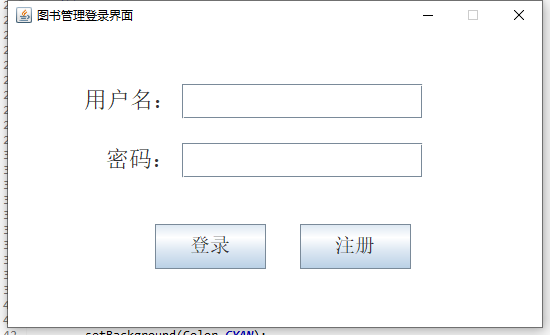
如果只更改1、2个字段，不要Update全部字段，否则频繁调用会引起明显的性能消耗，同时带来大量日志

16）删除Delete语句优化语句

17）插入Insert语句优化

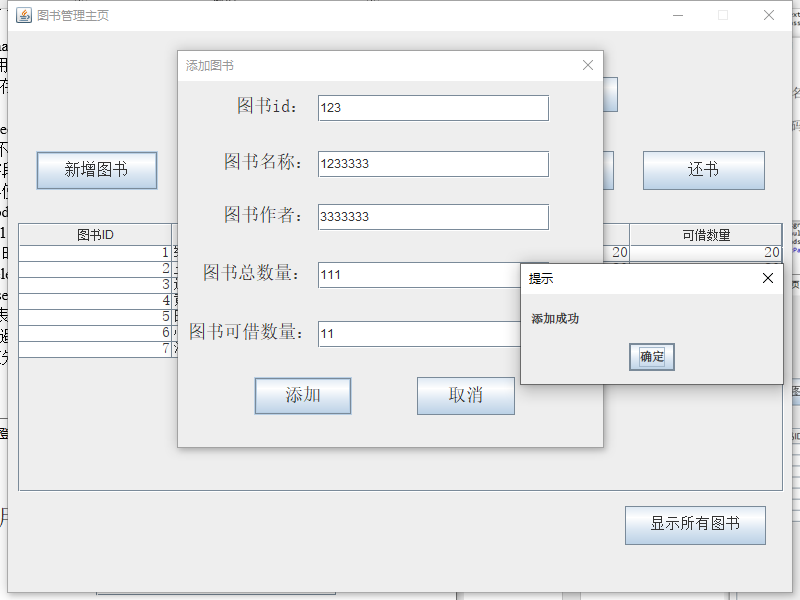
在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

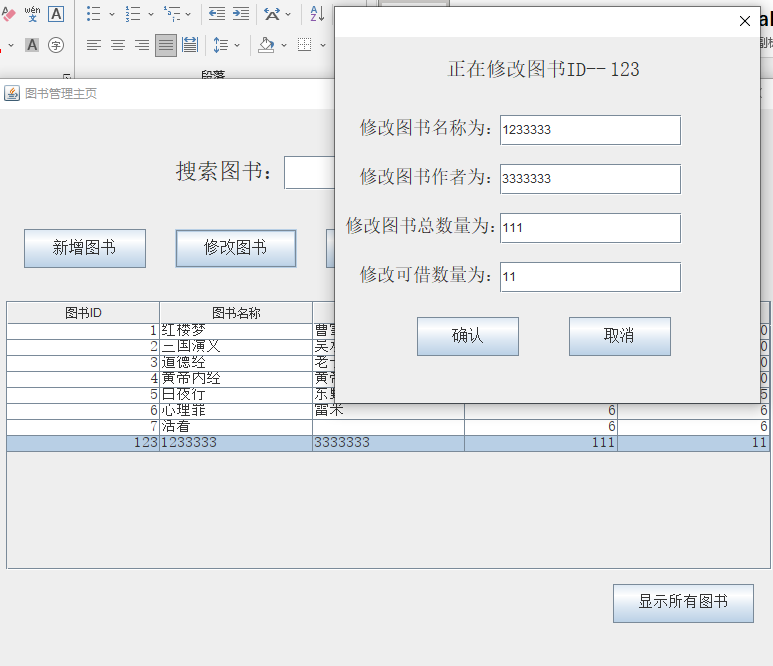
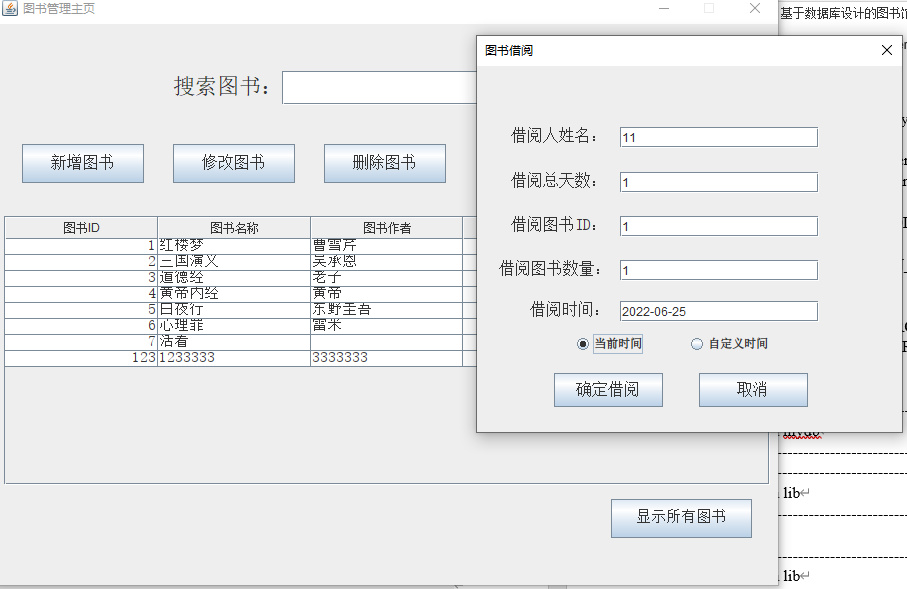
**六：数据库简易GUI实现**









**七：小结**

通过本次数据库业务实现，我初步学会了使用JBDC操作数据库的方式，学会了如何从一个业务的需求入手，首先分析出业务中实体的属性，将其对应到E-R图中，然后对应到关系型数据库中，采用范式理论分解，达到理论最优（一定程度上保障了安全性），通过对使用案例中视图，触发器，事务处理等等的分析，初步学会了数据库的建立及管理。

参考文献：

* [1]贾疏桐.数字图书馆建设中的安全保障策略研究[J].中国管理信息化,2020,23(04):197-198.
* [2]刘杰逾. SQL数据库性能优化必要性及其优化策略分析[J]. 电脑编程技巧与维护,2021,(07):104-105.
* [3]贾疏桐. 数字图书馆建设中的安全保障策略研究[J]. 中国管理信息化,2020,23(04):197-198.
* [4]张毅. 基于数据库设计的图书馆管理系统[J]. 电脑知识与技术,2019,15(31):6-7.
* [5]吕晓敏,李君艺. 基于SQL Server的高校图书馆数据库设计与实现[J]. 长春师范学院学报,2013,32(08):190-193+117.

附录：DDL生成语句

-- MySQL Script generated by MySQL Workbench

-- Fri Jun 24 23:22:12 2022

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema lib

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema lib

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `lib` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;

USE `lib` ;

-- -----------------------------------------------------

-- Table `lib`.`AdminEss`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`AdminEss` ;

CREATE TABLE IF NOT EXISTS `lib`.`AdminEss` (

`Name` VARCHAR(20) NOT NULL,

`IDCard` VARCHAR(16) NOT NULL,

`Tel` VARCHAR(15) NOT NULL,

`Address` VARCHAR(255) NULL,

PRIMARY KEY (`IDCard`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `lib`.`AdminAccount`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`AdminAccount` ;

CREATE TABLE IF NOT EXISTS `lib`.`AdminAccount` (

`Account` VARCHAR(20) NOT NULL,

`Password` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4\_0900\_ai\_ci' NOT NULL,

`Worktime` DATETIME NOT NULL,

`OutOfWorkTime` DATETIME NOT NULL,

`adminaccountcol` VARCHAR(45) NULL,

`adminess\_IDCard` VARCHAR(16) NOT NULL,

PRIMARY KEY (`Account`, `adminess\_IDCard`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `lib`.`BookEss`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`BookEss` ;

CREATE TABLE IF NOT EXISTS `lib`.`BookEss` (

`ISBN` VARCHAR(45) NOT NULL,

`Title` VARCHAR(255) NOT NULL,

`Author` VARCHAR(20) NOT NULL,

`TranslateAut` VARCHAR(10) NULL DEFAULT NULL,

`IfBorrowed` VARCHAR(3) NOT NULL,

PRIMARY KEY (`ISBN`, `Title`))

ENGINE = InnoDB

AUTO\_INCREMENT = 465

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `lib`.`UserEss`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`UserEss` ;

CREATE TABLE IF NOT EXISTS `lib`.`UserEss` (

`Name` VARCHAR(10) NOT NULL,

`IDCard` VARCHAR(16) NOT NULL,

`tel` VARCHAR(15) NOT NULL,

`gender` VARCHAR(2) NOT NULL,

PRIMARY KEY (`IDCard`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `lib`.`UserLogin`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`UserLogin` ;

CREATE TABLE IF NOT EXISTS `lib`.`UserLogin` (

`UserCard` VARCHAR(45) NOT NULL,

`VioTimes` INT NULL,

`IfBanned` VARCHAR(3) NOT NULL,

`OverdueTime` DATETIME NOT NULL,

`UserEss\_IDCard` VARCHAR(16) NOT NULL,

PRIMARY KEY (`UserEss\_IDCard`, `UserCard`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `lib`.`BookManage`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`BookManage` ;

CREATE TABLE IF NOT EXISTS `lib`.`BookManage` (

`AllNum` INT NOT NULL,

`CanBorrowNum` INT NOT NULL,

`BuyNum` INT NOT NULL,

`BuyTime` DATETIME NOT NULL,

`BuyPrice` DOUBLE NOT NULL,

`BookEss\_Title` VARCHAR(255) NOT NULL,

PRIMARY KEY (`BookEss\_Title`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `lib`.`BorrowAndReturn`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lib`.`BorrowAndReturn` ;

CREATE TABLE IF NOT EXISTS `lib`.`BorrowAndReturn` (

`BorrowDays` INT NOT NULL,

`BorrowNum` INT NOT NULL,

`BorrowTime` DATE NOT NULL,

`ReturnTime` DATE NOT NULL,

`UserLogin\_UserCard` VARCHAR(45) NOT NULL,

`BookManage\_BookEss\_Title` VARCHAR(255) NOT NULL,

PRIMARY KEY (`BookManage\_BookEss\_Title`))

ENGINE = InnoDB

AUTO\_INCREMENT = 56

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;