图书管理系统数据库设计说明书

目录

[一、数据库需求分析 1](#_Toc185959177)

[（一）项目背景 1](#_Toc185959178)

[（二）功能需求 2](#_Toc185959179)

[（三）性能需求 2](#_Toc185959180)

[二、数据库概念结构设计 3](#_Toc185959181)

[（一）实体E-R图 3](#_Toc185959182)

[（二）实体关系 3](#_Toc185959183)

[（三）系统E-R图 3](#_Toc185959184)

[三、数据库逻辑结构设计 4](#_Toc185959185)

[（一）将E-R图转换为关系模式 4](#_Toc185959186)

[（二）关系模式优化 4](#_Toc185959187)

[四、数据库物理结构设计 5](#_Toc185959188)

[（一）确定数据库的物理结构 5](#_Toc185959189)

[（二）设计索引 5](#_Toc185959190)

[（三）数据存储分配 5](#_Toc185959191)

[五、数据库操作代码部分 5](#_Toc185959192)

[（一）数据库创建代码 5](#_Toc185959193)

[（二）创建表代码 6](#_Toc185959194)

[（三）数据库的查询操作代码 7](#_Toc185959195)

[（四）视图创建代码 8](#_Toc185959196)

[（五）创建索引代码 9](#_Toc185959197)

[六、系统规范化设计 9](#_Toc185959198)

[（一）数据库表的规范化设计 10](#_Toc185959199)

[（二）数据库的完整性设计 10](#_Toc185959200)

[（三）数据库的维护和安全性设计 12](#_Toc185959201)

[七、总结 13](#_Toc185959202)

[（一）收获 13](#_Toc185959203)

[（二）不足和改进 14](#_Toc185959204)

一、数据库需求分析

（一）项目背景

在当今数字化时代，图书管理的高效性和便捷性对于学校、企业及各类机构至关重要。传统的图书管理方式难以满足大量图书的存储、检索、借阅及管理需求。因此，开发一个功能完备、用户友好的图书管理系统数据库成为必然。

（二）功能需求

1. \*\*用户管理\*\*

- 支持用户注册、登录及权限分配，区分图书管理员和普通用户。

- 普通用户可查询图书、借阅图书、查看借阅记录。

- 图书管理员负责图书入库、分类、修改、归还、管理借书记录等。

2. \*\*图书管理\*\*

- 实现图书信息的录入、修改、删除及查询，包括书号、书名、作者、价格、出版社、类别、库存等。

- 可按多种条件筛选图书，如书号、书名、作者等。

3. \*\*借阅管理\*\*

- 处理图书借阅、归还操作，自动更新库存及借阅记录。

- 支持查看所有用户借阅记录，对逾期归还进行罚款处理。

4. \*\*图书统计\*\*

- 按图书类别统计数量，为采购和资源配置提供依据。

（三）性能需求

1. 随着图书数量和用户量增加，系统应可扩展，保持高效性能。

2. 确保高可用性，全年稳定运行，减少维护影响。

3. 快速处理图书信息，保证检索、借阅、归还操作流畅。

二、数据库概念结构设计

（一）实体E-R图

1. \*\*图书实体\*\*

**图书**

2. \*\*用户实体\*\*

- 包含用户ID、姓名、性别、密码、专业等属性，分为管理员和普通用户。

**管理员**

**用户**

3. \*\*借阅记录实体\*\*

- 包括借书证号、书号、借阅者ID、借阅日期、应归还日期、当前状态、借还次数等属性。

**借阅记录**

（二）实体关系

1. 一个用户可借阅多本图书，一本图书可被多个用户借阅，形成多对多关系，通过借阅记录实体关联。

p

图书

m

n

借阅

用户

借阅记录

2. 图书管理员与图书管理操作（入库、修改、删除等）存在一对多关系。

n

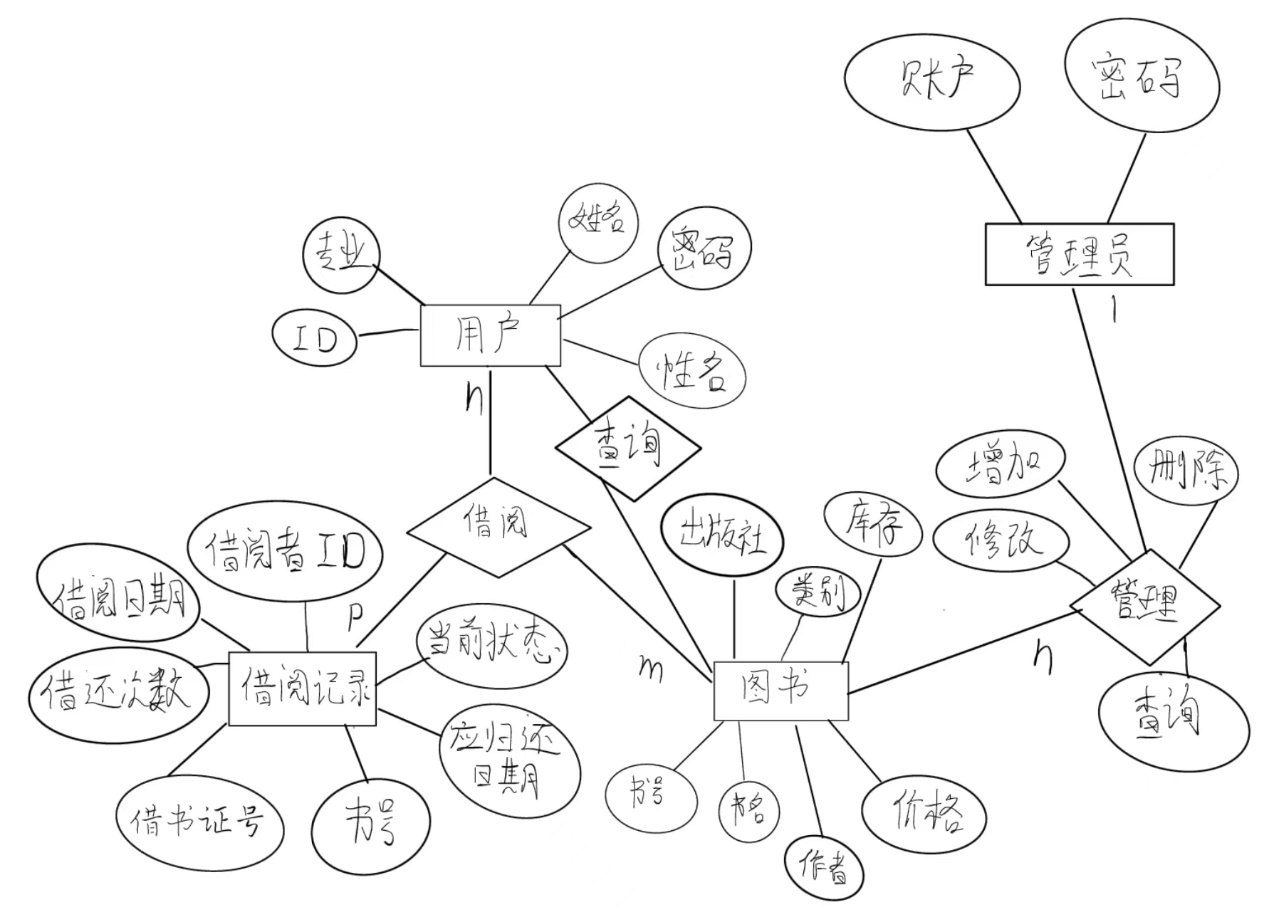
管理操作

图书

1

管理员

（三）系统E-R图

整合上述实体及关系，构建系统整体E-R图，明确各实体间联系及约束。

三、数据库逻辑结构设计

（一）将E-R图转换为关系模式

1. \*\*图书表（Book）\*\*

- （书号BookID，书名BookName，作者Author，价格Price，出版社Publisher，类别Category，库存Stock）

2. \*\*用户表（t\_user）\*\*

- （用户ID id，姓名name，性别sex，密码psw，专业dept）

3. \*\*借阅记录表（BorrowRecord）\*\*

- （借书证号BorrowerID，书号BookID，借阅者ID UserID，借阅日期BorrowDate，应归还日期DueDate，当前状态Status，借还次数BorrowCount）

4. \*\*管理员表（t\_admin）\*\*

- （账户id，密码psw）

（二）关系模式优化

对各关系模式进行分析，确保满足一定范式要求，减少数据冗余，保证数据完整性和一致性。例如，通过合理设置主键、外键及约束条件。

四、数据库物理结构设计

（一）确定数据库的物理结构

1. 选择合适的存储引擎，如SQL Server的默认存储引擎。

2. 确定数据文件和日志文件的存储位置、大小及增长方式。

（二）设计索引

1. 为经常用于查询条件和连接操作的字段创建索引，如图书表的书号BookID、用户表的用户id等，提高查询速度。

2. 考虑索引的维护成本，避免过度创建索引影响数据更新性能。

（三）数据存储分配

根据数据量和访问频率，合理分配表空间和数据页，优化数据存储。

五、数据库操作代码部分

（一）数据库创建代码

```sql

CREATE DATABASE BookDB;

GO

```

（二）创建表代码

1. \*\*创建图书表\*\*

```sql

CREATE TABLE Book (

BookID VARCHAR(20) NOT NULL PRIMARY KEY,

BookName VARCHAR(50) NOT NULL,

Author VARCHAR(50) NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

Publisher VARCHAR(50) NOT NULL,

Category VARCHAR(30) NOT NULL,

Stock INT NOT NULL

);

```

2. \*\*创建用户表\*\*

```sql

CREATE TABLE t\_user (

id VARCHAR(20) NOT NULL PRIMARY KEY,

name CHAR(2) NOT NULL,

sex VARCHAR(20) NOT NULL,

psw VARCHAR(20) NOT NULL,

dept VARCHAR(50) NOT NULL

);

```

3. \*\*创建借阅记录表\*\*

```sql

CREATE TABLE BorrowRecord (

RecordID INT NOT NULL,

BorrowerID VARCHAR(20) NOT NULL,

BookID VARCHAR(20) NOT NULL,

UserID VARCHAR(20) NOT NULL,

BorrowDate DATE NOT NULL,

DueDate DATE NOT NULL,

Status VARCHAR(20) NOT NULL,

Count INT NOT NULL,

PRIMARY KEY (RecordID),

FOREIGN KEY (BookID) REFERENCES

Book(BookID),

FOREIGN KEY (UserID) REFERENCES

t\_User(id)

);

```

（三）数据库的查询操作代码

1. \*\*查询指定图书信息\*\*

```sql

SELECT BookID, BookName, Author, Price, Publisher, Category, Stock

FROM Book

WHERE BookName = '三体';

```

2. \*\*查询用户借阅记录\*\*

```sql

SELECT BorrowerID, BookID, UserID, BorrowDate, DueDate, Status, Count

FROM BorrowRecord

WHERE BorrowerID = 'lyf';

```

（四）视图创建代码

1. \*\*创建可借阅图书视图\*\*

```sql

CREATE VIEW AvailableBooks

AS

SELECT BookID, BookName, Author, Price, Publisher, Category, Stock

FROM Book

WHERE Stock > 0;

```

2. \*\*创建逾期未还图书视图\*\*

```sql

CREATE VIEW OverdueBooks

AS

SELECT br.RecordID, br.BorrowerID, br.BookID, br.BorrowerName, br.BorrowDate, br.DueDate, br.Status, br.BorrowCount

FROM BorrowRecord br

WHERE br.Status = '未归还' AND br.DueDate < GETDATE();

```

（五）创建索引代码

1. \*\*为图书表书号创建唯一索引\*\*

```sql

CREATE UNIQUE INDEX IX\_Book\_BookID ON Book(BookID);

```

2. \*\*为用户表用户ID创建索引\*\*

```sql

CREATE INDEX IX\_User\_UserID ON User(id);

```

六、系统规范化设计

（一）数据库表的规范化设计

1. 分析各表结构，判断其范式级别。例如，图书表、用户表、借阅记录表均满足第三范式（3NF），消除了非主属性对码的部分函数依赖和传递函数依赖，保证了数据的规范性和完整性。

2. 若存在不满足范式要求的情况，进行适当的模式分解，确保数据冗余最小化，提高数据操作效率和一致性。

（二）数据库的完整性设计

1. \*\*实体完整性\*\*

- 在图书表、用户表、借阅记录表中，通过主键约束（如BookID、UserID等）确保实体的唯一性和可识别性，防止非法数据插入。

2. \*\*参照完整性\*\*

- 在借阅记录表中，通过外键约束（如BookID、UserID分别参照图书表和用户表的主键）维护表间数据的一致性和关联性，防止孤立数据出现。

3. \*\*用户自定义完整性\*\*

- 对用户表中的性别字段可定义约束，限制其取值范围为“男”或“女”；对图书库存字段可设置非负约束，确保数据符合实际业务规则。

4. \*\*创建触发器\*\*

- \*\*防止图书库存为负数\*\*

```sql

CREATE TRIGGER tr\_CheckStock

ON BookDB.Book

FOR UPDATE, INSERT

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE Stock < 0)

BEGIN

RAISERROR('图书库存不能为负数', 16, 1);

ROLLBACK TRANSACTION;

END

END;

```

- \*\*更新图书库存\*\*

```sql

CREATE TRIGGER tr\_UpdateStockOnBorrow

ON BookDB.BorrowRecord

FOR INSERT

AS

BEGIN

UPDATE BookDB.Book

SET Stock = Stock - 1

FROM inserted i

WHERE Book.BookID = i.BookID;

END;

```

- \*\*更新图书库存（归还操作）\*\*

```sql

CREATE TRIGGER tr\_UpdateStockOnReturn

ON BookDB.BorrowRecord

FOR UPDATE

AS

BEGIN

IF UPDATE(Status) AND EXISTS (SELECT 1 FROM inserted WHERE Status = '已归还')

BEGIN

UPDATE BookDB.Book

SET Stock = Stock + 1

FROM inserted i

WHERE Book.BookID = i.BookID;

END

END;

```

（三）数据库的维护和安全性设计

1. \*\*用户管理\*\*

- 创建不同角色用户，如借阅者、图书管理员，为其分配不同权限。

- 例如，借阅者具有查询图书、借阅图书、查看个人借阅记录权限；图书管理员可进行图书管理、借阅管理操作.

2. \*\*权限管理\*\*

- 通过GRANT和REVOKE语句为用户或角色授予和收回数据库对象的操作权限，如SELECT、INSERT、UPDATE、DELETE等，确保数据安全性和保密性。

3. \*\*数据备份与恢复\*\*

- 定期进行数据库全量和增量备份，制定备份策略，如每日全量备份，每小时增量备份。

- 在出现数据丢失或损坏时，可利用备份文件进行数据恢复，确保系统正常运行。

七、总结

（一）收获

1. 深入理解数据库设计流程，从需求分析、概念设计、逻辑设计到物理设计，构建完整数据库系统。

2. 掌握数据库设计工具和技术，如E-R图绘制、关系模式转换、SQL语言编写，提高实际操作能力。

3. 通过团队协作，学会分工合作、沟通协调，培养团队精神和项目管理能力。

（二）不足和改进

1. \*\*性能优化不足\*\*

- 部分复杂查询可能效率较低，可进一步优化查询语句、索引设计或数据库配置。

2. \*\*安全机制待完善\*\*

- 可加强用户认证和授权管理，采用加密技术保护敏感数据，防止数据泄露和恶意攻击。

3. \*\*系统可扩展性有限\*\*

- 考虑采用分布式数据库架构或云数据库服务，提高系统扩展性和容错性，适应未来业务增长。

4. \*\*用户体验需提升\*\*

- 优化界面设计，提供更友好操作提示和反馈，简化借阅流程，提高用户满意度。