数据库实验报告

实验人： 凌国明 学号： 21307077 日 期： 2023年12月8日

院（系）： 计算机学院 专业（班级）： 计算机科学与技术

实验题目： 实验5：数据库设计实验

一. 实验目的

掌握数据库设计基本方法及数据库设计工具。

二. 实验内容和要求

掌握数据库设计基本步骤，包括数据库概念结构设计、逻辑结构设计，物理结构设计，数据库模式 SQL 语句生成。能够使用数据库设计工具进行数据库设计。

三. 实验重点和难点

实验重点：概念结构设计、逻辑结构设计。

实验难点：逻辑结构设计。逻辑结构设计虽然可以按照一定的规则从概念结构转换而来，但是由于概念结构通常比较抽象，较少考虑更多细节，因此转换而成的逻辑结构还需要进一步调整和优化。逻辑结构承接概念结构和物理结构，处于核心地位，因而是数据库设计的重点，也是难点。

四. 实验工具

- Navicat （参阅：http://www.yaotu.net/biancheng/774.html ）
- PowerDesigner、ERwin、Office Visio、亿图图示、starUML、Visual Studio Code 等数据库设计工具
- MySQL、SQL Server

五. 实验过程

设计一个应用数据库。请使用 Navicat、PowerDesigner 或者 ERwin 等数据库设计工具设计该数据库。

(1)数据库概念结构设计

先识别出系统中的实体。然后根据实际语义，分析实体之间的联系，确定实体之间一对一，一对多和多对多联系。据此，绘制实体-联系图(E-R 图)。

设计如下（参阅实验教材）：

设计一个学校图书借阅管理系统，根据图书借阅相关信息，设计实体如下：

图书(Book):属性包括书号、书名、作者、出版社等。

类别(Category):独立实体，属性包括类别 ID、类别名称等。

读者(Reader): 包含读者 ID、姓名、联系方式等属性。

借阅证(Borrowing Card): 包含借阅证 ID、有效期等属性。

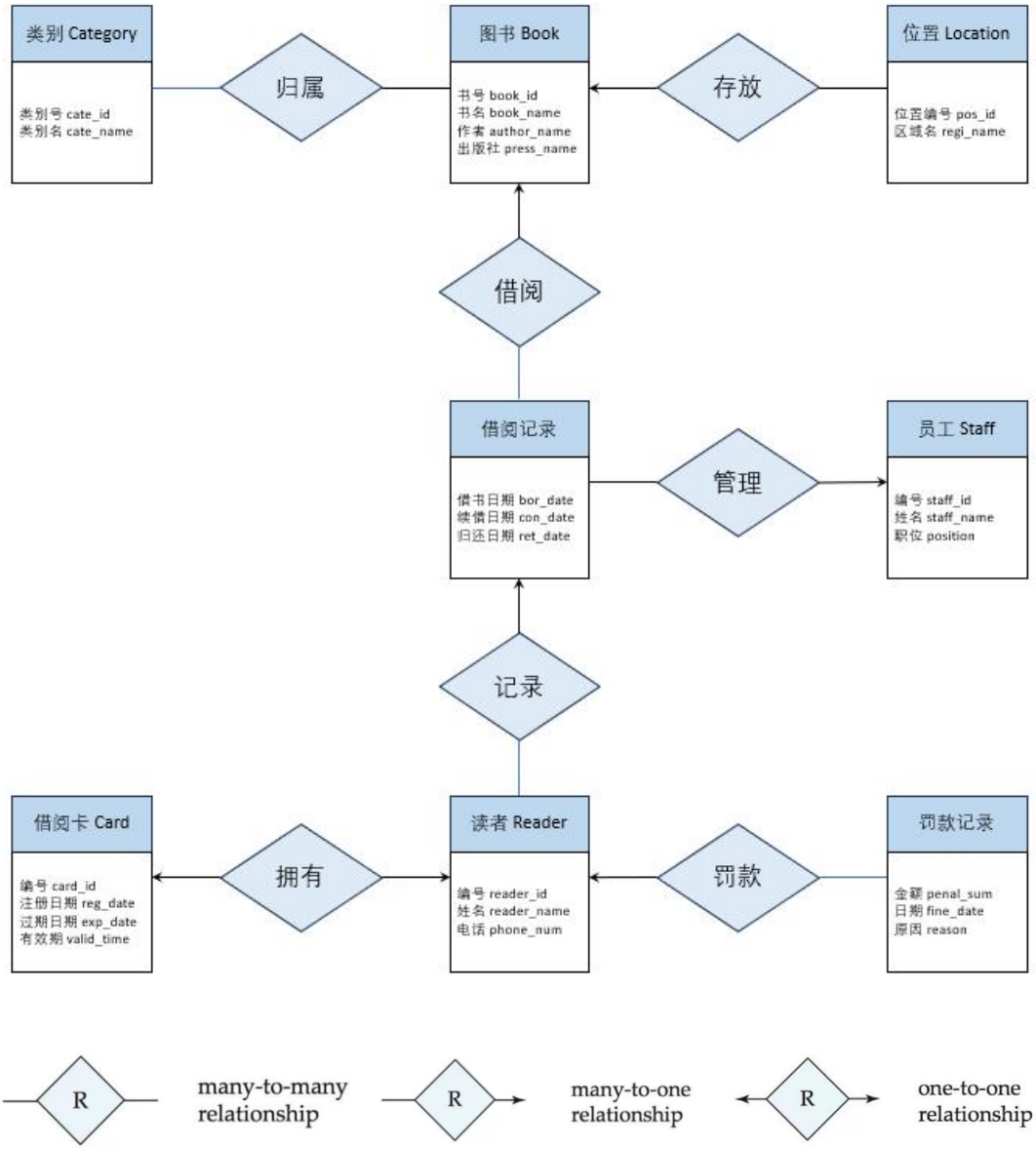
借阅记录(Borrowing Record): 包含借书日期、续借日期、归还日期等属性。

罚款记录(Fine Record): 记录罚款的详细信息。

员工(Staff): 包含员工 ID、姓名、职位等属性。

位置(Location): 图书的具体存放位置，包含位置 ID、区域等属性。

根据现实生活中各实体之间的联系，我设计了一组联系集来关联实体集。



归属(Book - Category)：一个图书可以对应多个类别，一个类别可以包含多个图书。关系类型：M:N。

借阅(Borrowing Record - Book)：一个借阅记录对应一个图书，一个图书可以有多个借阅记录。关系类型：1:N。

拥有(Reader - Borrowing Card)：一个读者拥有一个借阅证，一个借阅证对应一个读者。关系类型：1:1。

罚款(Reader - Fine Record)：一个读者可以有多个罚款记录，一个罚款记录对应一个读者。关系类型：1:N。

记录(Borrowing Card - B.. Record)：一个借阅证对应多个借阅记录，一个借阅记录对应一个借阅证。关系类型：1:N。

管理(Staff - Borrowing Record)：一个员工可以管理多个借阅记录，一个借阅记录由一个员工管理。关系类型：1:N。

存放(Book - Location)：一个图书存放在一个位置，一个位置可以有多个图书。关系类型：1:N。

这里为了方便向逻辑结构转换，统一采用二元关系去表达实体之间的联系。

(2)数据库逻辑结构设计

按照数据库设计原理中概念结构转化成逻辑结构的规则，每个实体转换成一个关系，多对多的联系也转换成一个关系。从而，根据上述 E-R 图设计数据库逻辑结构。

设计如下（参阅实验教材）：

图书(Book) 表： 书号 (主键) 书名 作者 出版社 是否在馆	类别(Category) 表： 类别 ID (主键) 类别名称	读者(Reader) 表： 读者 ID (主键) 姓名 联系方式
借阅证(Borrowing Card) ： 借阅证 ID (主键) 有效期 读者 ID (外键，引用读者表的读者 ID)	位置(Location) 表： 位置 ID (主键) 区域 书号 (外键,引用图书表的书号)	罚款记录(Fine Record) 表： 罚款记录 ID (主键) 详细信息 读者 ID (外键，引用读者表的读者 ID)
员工(Staff) 表： 员工 ID (主键) 姓名 职位	借阅记录(Borrowing Record) 表： 借阅记录 ID (主键) 借书日期 续借日期 归还日期 书号 (外键,引用图书表的书号) 借阅证 ID (外键，引用借阅证表的借阅证 ID) 员工 ID (外键，引用员工表的员工 ID)	图书与类别关系(处理 M:N 关系)： 书号 (外键,引用图书表的书号) 类别 ID (外键，引用类别表的类别 ID)

(3)数据库物理结构设计

数据库物理结构首先根据逻辑结构自动转换生成，然后根据应用需求设计数据库的索引结构、存储结构。

设计如下（参阅实验教材）：

以下是索引结构的设计：

图书(Book)表： 对书号建立主键索引，因为它是唯一标识。 对书名、作者和出版社建立非聚集索引，以加快搜索和查询速度。	类别(Category)表： 对类别 ID 建立主键索引。 对类别名称建立非聚集索引。	读者(Reader)表： 对读者 ID 建立主键索引。 对姓名建立非聚集索引，以便快速检索读者信息。
借阅证(Borrowing Card)表： 对借阅证 ID 建立主键索引。 对读者 ID 建立外键索引。	借阅记录(Borrowing Record)表： 对借阅记录 ID 建立主键索引。 对书号、借阅证 ID 和员工 ID 建立外键索引，以加快关联查询。	罚款记录(Fine Record)表： 对罚款记录 ID 建立主键索引。 对读者 ID 建立外键索引。
员工(Staff)表： 对员工 ID 建立主键索引。	位置(Location)表： 对位置 ID 建立主键索引。 对书号建立外键索引。	图书与类别关系(Book-Category)表： 对书号和类别 ID 建立复合索引。

以下是存储结构的设计

图书(Book)表： 使用 B-tree 结构存储书号索引，以优化查询和范围搜索。 书名、作者和出版社字段可使用全文索引以优化文本搜索。	类别(Category)和读者(Reader)表： 类别 ID 和读者 ID 使用 B-tree 索引。 类别名称和读者姓名可使用散列索引以加快等值查询。
借阅记录(Borrowing Record)表： 使用 B-tree 索引来存储借阅记录 ID，书号和借阅证 ID。 考虑到借书和还书操作频繁，可以使用聚集索引来优化插入和更新性能。	位置(Location)表： 使用 B-tree 索引来存储位置 ID。 书号索引应考虑空间效率，因为一个位置可能存放多本书。

(4) 数据库模式 SQL 语句生成

生成 MySQL 或 SQL Server 数据库管理系统的 SQL 语句。

设计如下（参阅实验教材）：

```sql
-- 创建图书(Book)表
CREATE TABLE Book (
    book_id VARCHAR(20) PRIMARY KEY,
    title VARCHAR(100),
    author VARCHAR(100),
    publisher VARCHAR(100),
    is_borrowed BOOLEAN DEFAULT FALSE
);
-- 为图书表的书名、作者和出版社创建非聚集索引
CREATE INDEX idx_book_title ON Book (title);
CREATE INDEX idx_book_author ON Book (author);
CREATE INDEX idx_book_publisher ON Book (publisher);
```

```sql
-- 创建类别(Category)表
CREATE TABLE Category (
    category_id INT PRIMARY KEY,
    name VARCHAR(50)
);
-- 为类别名称创建非聚集索引
CREATE INDEX idx_category_name ON Category (name);

-- 创建读者(Reader)表
CREATE TABLE Reader (
    reader_id INT PRIMARY KEY,
    name VARCHAR(100),
    contact_info VARCHAR(100)
);
-- 为读者姓名创建非聚集索引
CREATE INDEX idx_reader_name ON Reader (name);

-- 创建借阅证(Borrowing Card)表
CREATE TABLE BorrowingCard (
    card_id INT PRIMARY KEY,
    valid_until DATE,
    reader_id INT,
    FOREIGN KEY (reader_id) REFERENCES Reader(reader_id)
);

-- 创建员工(Staff)表
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(100)
);

-- 创建借阅记录(Borrowing Record)表
CREATE TABLE BorrowingRecord (
    record_id INT PRIMARY KEY,
    borrow_date DATE,
    renew_date DATE,
    return_date DATE,
    book_id VARCHAR(20),
    card_id INT,
    staff_id INT,
    FOREIGN KEY (book_id) REFERENCES Book(book_id),
    FOREIGN KEY (card_id) REFERENCES BorrowingCard(card_id),
    FOREIGN KEY (staff_id) REFERENCES Staff(staff_id)
);

-- 创建罚款记录(Fine Record)表
CREATE TABLE FineRecord (
    fine_id INT PRIMARY KEY,
    details VARCHAR(255),
    reader_id INT,
    FOREIGN KEY (reader_id) REFERENCES Reader(reader_id)
);
```

```sql
-- 创建位置(Location)表
CREATE TABLE Location (
    location_id INT PRIMARY KEY,
    area VARCHAR(100),
    book_id VARCHAR(20),
    FOREIGN KEY (book_id) REFERENCES Book(book_id)
);

-- 创建图书与类别关系(Book-Category)表
CREATE TABLE BookCategory (
    book_id VARCHAR(20),
    category_id INT,
    FOREIGN KEY (book_id) REFERENCES Book(book_id),
    FOREIGN KEY (category_id) REFERENCES Category(category_id),
    PRIMARY KEY (book_id, category_id)
);
```

然后创建触发器，在修改借阅记录时，更新图书的状态

```sql
-- 创建触发器：借书时自动更新图书的is_borrowed字段
DELIMITER $$
CREATE TRIGGER trigger_borrow_book
AFTER INSERT ON BorrowingRecord
FOR EACH ROW
BEGIN
    UPDATE Book SET is_borrowed = TRUE WHERE book_id = NEW.book_id;
END$$
DELIMITER ;

-- 创建触发器：还书时自动更新图书的is_borrowed字段
DELIMITER $$
CREATE TRIGGER trigger_return_book
AFTER UPDATE ON BorrowingRecord
FOR EACH ROW
BEGIN
    IF NEW.return_date IS NOT NULL THEN
        UPDATE Book SET is_borrowed = FALSE WHERE book_id = NEW.book_id;
    END IF;
END$$
DELIMITER ;
```

最后创建视图，查询各种图书的书号、书名、总数和在册数（未被借出的图书数）

```sql
-- 创建视图：查询各种图书的书号、书名、总数和在册数（未被借出的图书数）
CREATE VIEW view_book_details AS
SELECT book_id, title, COUNT(*) AS total_count, SUM(NOT is_borrowed) AS available_count
FROM Book
GROUP BY book_id, title;
```

Book 表：

名	类型	长度	小数点	不是 null	虚拟	键
book_id	varchar	20		☑	☐	1
title	varchar	100		☐	☐	
author	varchar	100		☐	☐	
publisher	varchar	100		☐	☐	
is_borrowed	tinyint	1		☐	☐	

BookCategory 表：

名	类型	长度	小数点	不是 null	虚拟	键
book_id	varchar	20		☑	☐	1
category_id	int			☑	☐	2

BorrowingCard 表：

名	类型	长度	小数点	不是 null	虚拟	键
card_id	int			☑	☐	1
valid_until	date			☐	☐	
reader_id	int			☐	☐	

BorrowingRecord 表

字段 索引 外键 检查 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
record_id	int			☑	☐	1
borrow_date	date			☐	☐	
renew_date	date			☐	☐	
return_date	date			☐	☐	
book_id	varchar	20		☐	☐	
card_id	int			☐	☐	
staff_id	int			☐	☐	

Category 表：

名	类型	长度	小数点	不是 null	虚拟	键
category_id	int			☑	☐	1
name	varchar	50		☐	☐	

FineRecord 表：

名	类型	长度	小数点	不是 null	虚拟	键
fine_id	int			☑	☐	1
details	varchar	255		☐	☐	
reader_id	int			☐	☐	

Location 表：

名	类型	长度	小数点	不是 null	虚拟	键
location_id	int			☑	☐	🔑 1
area	varchar	100		☐	☐	
book_id	varchar	20		☐	☐	

Reader 表：

名	类型	长度	小数点	不是 null	虚拟	键
reader_id	int			☑	☐	🔑 1
name	varchar	100		☐	☐	
contact_info	varchar	100		☐	☐	

Staff 表：

名	类型	长度	小数点	不是 null	虚拟	键
staff_id	int			☑	☐	🔑 1
name	varchar	100		☐	☐	
position	varchar	100		☐	☐	

BorrowingRecord 表中的触发器：

字段 | 索引 | 外键 | 检查 | 触发器 | 选项 | 注释 | SQL 预览

名	触发	插入	更新	删除
trigger_borrow_boc	AFTER	☑	☐	☐
rigger_return_book	AFTER	☐	☑	☐

trigger_borrow_book：

定义

```
1 BEGIN
2     UPDATE Book SET is_borrowed = TRUE WHERE book_id = NEW.book_id;
3 END
```

trigger_return_book：

定义

```
1 BEGIN
2     IF NEW.return_date IS NOT NULL THEN
3         UPDATE Book SET is_borrowed = FALSE WHERE book_id = NEW.book_id;
4     END IF;
5 END
```

六. 与实验结果相关的文件

init.sql：SQL 语句初始化数据库（建表，创建触发器，创建视图）

library.sql：数据库的转储（library 数据库的结构和数据）

七. 实验总结

在这次的数据库设计实验中，我从 ER 图的分析开始，逐步深入到逻辑结构的设计，再到物理结构（包括索引和存储结构）的实现，最后通过编写 SQL 语句完成了整个数据库的构建。这个过程不仅加深了我对数据库设计原理的理解，也提升了我的实际操作能力。

首先，ER 图为我提供了一个清晰的视角来观察和理解实体间的关系。通过将每个实体转化为一个关系表，并为多对多关系建立单独的关联表，我得以在逻辑层面上确立了数据的结构。这个步骤对于后续的物理设计至关重要，因为它直接影响了数据的存储和检索效率。

进入物理设计阶段，我特别关注于索引结构和存储结构的优化。通过为主键和频繁查询的字段创建索引，我能显著提升查询速度。特别是在面对大量数据和复杂查询时，合理的索引策略显得尤为重要。此外，考虑到数据的访问模式和存储效率，我选择了适合各种场景的索引类型，如 B-tree 和全文索引。

编写 SQL 语句是实践中最具挑战性的部分。我不仅需要确保语句的正确性和高效性，还要考虑数据完整性和安全性。创建触发器和视图使我能够自动化一些常规任务，如更新图书的借阅状态，同时也使得复杂的查询变得简单。通过这个过程，我深刻体会到了 SQL 语言的强大和灵活性。

总的来说，这次实验是一次宝贵的学习经历。这次实验我不仅更深入地理解了数据库设计的理论知识，而且我还通过实践学会了如何将理论应用于实际的数据库构建中。