CS33503数据库系统实验

实验检查记录

|  |  |  |  |
| --- | --- | --- | --- |
| 实验结果的正确性(60%) |  | 表达能力(10%) |  |
| 实验过程的规范性(10%) |  | 实验报告(20%) |  |
| 加分(5%) |  | 总成绩(100%) |  |

实验报告

一、实验目的

|  |
| --- |
| 1. 学会正确运用概念数据库设计方法，正确使用实体-联系图(ER图)表示概念数据模型。 2. 学会正确运用逻辑数据库设计方法，在概念数据模型的基础上，设计规范的关系数据库模式。 3. 学会正确运用物理数据库设计方法，根据工作负载，合理设计与调优数据库的存取方法与模式。 4. 掌握一种关系数据库管理系统(RDBMS)的使用方法，使用SQL创建、更新和查询关系数据库。 5. 掌握数据库系统应用开发方法。 |

二、实验环境

|  |
| --- |
| 硬件设备：Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz  软件系统：Windows 11 22H2、Ubuntu 20.04.4 LTS、MySQL Server version: 8.0.28-  0ubuntu0.20.04.3  开发工具：Visual Studio Code 1.65.2 |

三、实验过程

|  |
| --- |
| **数据库设计和应用开发：**   1. 需求分析   在本实验中，我们计划实现一个书评网站的后台数据管理系统。我们计划实现以下的功能：   * + 管理员和用户分别登陆，跳转到不同窗口。为了简化，登陆时只输入账号名。管理员的账号名为r00t；用户的账号名为任意字符，自动注册。   + 管理员可以按照标题、作者、译者、出版社、出版时间、价格关键字查找图书（属性为空表示此项没有约束），可以添加、修改、删除图书内容。   + 管理员可以按照被评论图书标题、评论用户名来查找评论。也可删除评论。   + 管理员可以按照用户名查找用户，当用户名为空时显示所有用户。也可以删除用户。   + 用户可以按照标题、作者、译者、出版社、出版时间关键字查找图书。   + 用户可以按照被评论图书标题、评论用户名来查找评论。   + 用户可以赞评论。   + 用户可以按照书名给图书评分和评论。   + 用户可以按照用户名搜索、关注、取消关注其他用户。注意，用户名不能为空，即用户不能得到所有用户列表。   + 图书评分由用户的评分实时计算。   + 评论的获赞数实时计算。   + 用户的关注者数量和关注数量实时计算。  1. 概念数据库设计   我们根据需求设计了相应的概念数据库。ER图如下所示：  D:\Dedsecr\HIT-DataBase\lab\lab3\ER.drawio.png   1. 逻辑数据库设计   由ER模型转换的关系数据库模式如下：   * Book(id, title, time, price) * Press(id, name, location) * Author(id, name, country, gender) * Translator(id, name, country, gender) * Reviewer(id, name, gender) * Publish(book\_id, press\_id) * WriteBook(book\_id, author\_id) * TranslateBook(book\_id, translator\_id) * Review(review\_id, book\_id, reviewer\_id, time, rating, body) * LikeReview(reviewer\_id, review\_id) * Follow(fed\_id, fing\_id)   我们在MySQL中建立数据库后导出关系模式ER图如下图所示，下面的图展示了我们设计的逻辑数据库的结构：     1. 物理数据库设计   为了简化模型，也为了加快索引速度，我们对Book、Press、Author、Translator和Reviewer的title或name键设置为UNIQUE KEY，即值唯一，这样在搜索及查询时能够根据具体实际来有效地加快查询速度。  我们根据用户需求，分别设计了3个视图，分别为：   * 图书视图，属性包括图书ID，图书标题，作者姓名，译者姓名，出版社名称，出版时间，价格，和用户评分（实时计算）。 * 评论视图，属性包括评论ID，被评论图书标题，评论用户名，评论时间，评分，评论内容，和获赞数（实时计算）。 * 用户关注情况视图，属性包括用户ID，用户名，用户性别，被关注数（实时计算），关注数（实时计算）。  1. 数据库建立   我们如下的SQL语句来建立数据库以及视图：   1. DROP DATABASE IF EXISTS BookReview; 2. CREATE DATABASE BookReview; 3. USE BookReview; 4. CREATE TABLE Book ( 5. `id` INT AUTO\_INCREMENT, 6. `title` VARCHAR(128) NOT NULL, 7. `time` YEAR, 8. `price` DECIMAL(8, 2), 9. PRIMARY KEY (`id`), 10. UNIQUE KEY `title\_uni` (`title`) 11. ); 12. CREATE TABLE Press ( 13. `id` INT AUTO\_INCREMENT, 14. `name` VARCHAR(128) NOT NULL, 15. `location` VARCHAR(128), 16. PRIMARY KEY (`id`), 17. UNIQUE KEY `name\_uni` (`name`) 18. ); 19. CREATE TABLE Author ( 20. `id` INT AUTO\_INCREMENT, 21. `name` VARCHAR(128) NOT NULL, 22. `country` VARCHAR(128), 23. `gender` ENUM('F', 'M', 'UNK') DEFAULT 'UNK', 24. PRIMARY KEY (`id`), 25. UNIQUE KEY `name\_uni` (`name`) 26. ); 27. CREATE TABLE Translator ( 28. `id` INT AUTO\_INCREMENT, 29. `name` VARCHAR(128) NOT NULL, 30. `country` VARCHAR(128), 31. `gender` ENUM('F', 'M', 'UNK') DEFAULT 'UNK', 32. PRIMARY KEY (`id`), 33. UNIQUE KEY `name\_uni` (`name`) 34. ); 35. CREATE TABLE Reviewer( 36. `id` INT AUTO\_INCREMENT, 37. `name` VARCHAR(128) NOT NULL, 38. `gender` ENUM('F', 'M', 'UNK') DEFAULT 'UNK', 39. PRIMARY KEY(`id`), 40. UNIQUE KEY `name\_uni` (`name`) 41. ); 42. CREATE TABLE Publish( 43. `book\_id` INT, 44. `press\_id` INT, 45. PRIMARY KEY(`book\_id`, `press\_id`), 46. FOREIGN KEY(`book\_id`) REFERENCES Book(`id`) ON DELETE CASCADE, 47. FOREIGN KEY(`press\_id`) REFERENCES Press(`id`) ON DELETE CASCADE 48. ); 49. CREATE TABLE WriteBook( 50. `book\_id` INT, 51. `author\_id` INT, 52. PRIMARY KEY(`book\_id`, `author\_id`), 53. FOREIGN KEY(`book\_id`) REFERENCES Book(`id`) ON DELETE CASCADE, 54. FOREIGN KEY(`author\_id`) REFERENCES Author(`id`) ON DELETE CASCADE 55. ); 56. CREATE TABLE TranslateBook( 57. `book\_id` INT, 58. `translator\_id` INT, 59. PRIMARY KEY(`book\_id`, `translator\_id`), 60. FOREIGN KEY(`book\_id`) REFERENCES Book(`id`) ON DELETE CASCADE, 61. FOREIGN KEY(`translator\_id`) REFERENCES Translator(`id`) ON DELETE CASCADE 62. ); 63. CREATE TABLE Review( 64. `review\_id` INT AUTO\_INCREMENT, 65. `book\_id` INT, 66. `reviewer\_id` INT, 67. `time` DATE NOT NULL, 68. `rating` INT NOT NULL CHECK (rating >= 1 AND rating <= 5), 69. `body` TEXT NOT NULL, 70. PRIMARY KEY(`review\_id`), 71. FOREIGN KEY(`book\_id`) REFERENCES Book(`id`) ON DELETE CASCADE, 72. FOREIGN KEY(`reviewer\_id`) REFERENCES Reviewer(`id`) ON DELETE CASCADE 73. ); 74. CREATE TABLE LikeReview( 75. `reviewer\_id` INT, 76. `review\_id` INT, 77. PRIMARY KEY(`reviewer\_id`, `review\_id`), 78. FOREIGN KEY(`reviewer\_id`) REFERENCES Reviewer(`id`) ON DELETE CASCADE, 79. FOREIGN KEY(`review\_id`) REFERENCES Review(`review\_id`) ON DELETE CASCADE 80. ); 81. CREATE TABLE Follow( 82. `fed\_id` INT, 83. `fing\_id` INT, 84. PRIMARY KEY(`fed\_id`, `fing\_id`), 85. FOREIGN KEY(`fed\_id`) REFERENCES Reviewer(`id`) ON DELETE CASCADE, 86. FOREIGN KEY(`fing\_id`) REFERENCES Reviewer(`id`) ON DELETE CASCADE 87. ); 88. CREATE VIEW view\_books AS SELECT Book.id, Book.title, Author.name AS author, Translator.name AS translator, Press.name AS press, Book.time, Book.price, AVG(Review.rating) FROM Book LEFT JOIN WriteBook ON WriteBook.book\_id=Book.id LEFT JOIN Author ON Author.id=WriteBook.author\_id LEFT JOIN TranslateBook ON TranslateBook.book\_id=Book.id LEFT JOIN Translator ON Translator.id=TranslateBook.translator\_id LEFT JOIN Publish ON Publish.book\_id=Book.id LEFT JOIN Press ON Press.id=Publish.press\_id LEFT JOIN Review ON Review.book\_id=Book.id GROUP BY Book.id, Book.title, author, translator, press, Book.time, Book.price; 89. CREATE VIEW view\_reviews AS SELECT Review.review\_id, Book.title AS title, Reviewer.name AS reviewer, Review.time, Review.rating, Review.body, COUNT(LikeReview.reviewer\_id) AS like\_num FROM Review LEFT JOIN Reviewer ON Reviewer.id=Review.reviewer\_id LEFT JOIN Book ON Book.id=Review.book\_id LEFT JOIN LikeReview ON LikeReview.review\_id=Review.review\_id GROUP BY Review.review\_id; 90. *-- for view view\_follow* 91. CREATE VIEW view\_followed\_num AS SELECT fed\_Reviewer.id AS fed\_id, fed\_Reviewer.name AS fed\_name, fed\_Reviewer.gender AS fed\_gender, COUNT(fing\_Reviewer.name) AS fed\_num FROM Follow RIGHT JOIN Reviewer AS fed\_Reviewer ON fed\_Reviewer.id=Follow.fed\_id LEFT JOIN Reviewer AS fing\_Reviewer ON fing\_Reviewer.id=Follow.fing\_id GROUP BY fed\_Reviewer.name; 92. *-- for view view\_follow* 93. CREATE VIEW view\_following\_num AS SELECT fing\_Reviewer.id AS fing\_id, fing\_Reviewer.name AS fing\_name, fing\_Reviewer.gender AS fing\_gender, COUNT(fed\_Reviewer.name) AS fing\_num FROM Follow RIGHT JOIN Reviewer AS fing\_Reviewer ON fing\_Reviewer.id=Follow.fing\_id LEFT JOIN Reviewer AS fed\_Reviewer ON fed\_Reviewer.id=Follow.fed\_id GROUP BY fing\_Reviewer.name; 94. CREATE VIEW view\_follow AS SELECT fed\_id AS id, fed\_name AS name, fed\_gender AS gender, fed\_num, fing\_num FROM view\_followed\_num FULL JOIN view\_following\_num ON fed\_id=fing\_id; 95. 向数据库中添加大量数据   图书和评论的数据来自全国高校数据驱动创新研究大赛 (北京大学)提供的《豆瓣读书top250图书信息与热门评论数据集》（<https://opendata.pku.edu.cn/dataset.xhtml?persistentId=doi:10.18170/DVN/X20PS1>）。我们将得到的数据进行数据清洗，得到更为规范的数据。对于Book和Review的所有属性，我们都可以数据中获得，我们直接添加即可。对于Press、Author、Translator和Reviewer的部分属性（一项或者两项）则无法从数据中获得，对于有默认值的属性，我们将其设置为了默认值；对于没有默认值的属性，我们将其设置为空。而对于用户之间的关注关系，我们则没有导入数据。   1. 根据应用需求进行数据库应用系统设计   我们基于PyQt5实现了数据库应用的GUI界面，代码在./gui下，下面我们将简要介绍各个界面的设计和功能。  **登录界面：**    对于登录界面，我们只做了一个用户名的输入框和登录按钮。若用户输入“r00t”，并点击登录，则进入管理员界面；若用户输入其他用户名，并点击登录，我们自动为其注册，进入用户界面。  **管理员界面：**        管理员界面分为三个页面，包括管理图书，管理评论和管理用户。  管理图书页面拥有标题、作者、译者、出版社、出版时间、和价格的输入框，还有用于搜索、添加、修改和删除图书信息的按钮。管理员可以按照这些关键字查找图书，若属性为空表示此项没有约束。管理员可以通过这些输入框来添加和修改图书信息。管理员也可以通过这些输入框来条件地筛选并删除图书信息。  管理评论页面拥有图书标题和评论用户名的两个输入框，还有用于搜索和删除评论的按钮。管理员可以按照这些关键字搜索并查看相关评论，若有不当评论，则还可以通过删除按钮来删除这些评论。  管理用户界面拥有用户名的输入框，还有用于搜索和删除用户的按钮。当用户名为空时，点击搜索按钮，则会显示所有用户列表。管理员也可以按照用户名来搜索并筛选用户，也可以按照用户名来删除用户。  **用户界面：**        用户界面分为三个页面，包括浏览图书，评论和关注用户。  浏览图书页面拥有标题、作者、译者、出版社和出版时间的输入框，还有用于搜索图书信息的按钮。用户可以按照这些关键字查找并浏览图书信息，若属性为空表示此项没有约束。若全部属性为空则展示全部的图书列表。  评论页面有三个功能区，从上到下分别为赞评论、浏览评论、发表评论。用户可以通过输入评论的ID，并点击赞评论按钮来给评论点赞。图书标题和评论用户的输入框可以让用户按照这些筛选项来筛选想要的评论。最下面的图书标题、评分和评论输入框可以用于用户的评论的发表。  关注用户界面拥有用户昵称的输入框，还有搜索、关注和取消关注的按钮。用户可以按照用户昵称来搜索用户。也可以点击关注按钮来关注其他用户，点击取消关注按钮来取消关注其他用户。  **实验结果及演示：**   1. 索引效果演示：   我们用下面的例子来展示我们在表Reviewer上建立name的UNIQUE INDEX的效果      上图为建立前的查询耗时，下图为建立索引后的查询耗时。我们可以看到，增加索引后，进行同样的查询，查询时间从0.007秒降到了0.0003秒，提升了20倍以上。   1. 系统功能演示：   **管理员界面：**  查找作者为“刘慈欣”的图书：    添加标题为“三体4”的图书    修改标题为“三体4”的图书价格为514元    删除标题为“三体4”的图书    查找对标题为“三体”的图书的评论    删除用户yutori对标题为“三体”的图书的评论    查找昵称为lecia的用户    删除昵称为lecia的用户    **用户“user1”界面：**  （下面只展示管理员界面没有的功能）  对ID为1的评论点赞    对标题为“三体”的图书添加一个评论    关注昵称为eiris的用户    取消关注昵称为eiris的用户 |

四、实验结论

|  |
| --- |
| 本次实验的工作量很大，但是难度并不高。主要需要注意的是从零开始的选题和对于ER图的设计。关系数据库模式的实现，后端代码，和前端代码的书写较为繁琐。同时，通过这此实验也让我熟悉了数据库应用系统设计的整个pipeline，包括熟悉需求，画ER图，转换为关系数据库模式等等。我也在这次试验中第一次用到了Python的GUI库PyQt5，第一次实现了自己的GUI界面。 |