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（介绍实验目的）

|  |
| --- |
| 1. 熟悉SQL语言，熟练掌握关系数据库系统的使用 2. 学习简单数据库系统的设计方法，包括数据库概要设计、逻辑设计 3. 能够向数据库中添加一定规模的数据 4. 能够对数据库性能进行简单的评估与优化 |

二、实验环境（介绍实验使用的硬件设备、软件系统、开发工具等）

|  |
| --- |
| 本实验采用MySQL关系数据库，使用Python语言进行开发。用户界面部分则使用 PyQt5进行生成，在测试、数据导入、可视化过程中，使用了Navicat数据库管理工具。 |

三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **实验内容概述**   本次实验中，我完成了一个用于学校管理的综合性系统，系统包括老师，学生，教室等实体，可以实现成绩录入，显示课程排班，教学任务安排等功能。在完成数据库的概念性设计后，我使用PyQt5完成了用户界面的设计，并使其能够与数据库进行交互。   1. **实验内容1: 数据库设计**   **2.1需求分析**  预想的该系统具有的功能如下：   1. 学生选课 2. 学生成绩录入 3. 学生成绩查询 4. 课程成绩查询 5. 教学任务安排 6. 教室安排   根据这些功能，需要设计以下9个实体，分别是：院系，班级，学生，老师，成绩，课程，课程的教学任务，课程的教室安排，教室。关于它们的具体定义以及约束以表的方式列举如下：  表 1：Student   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | s\_id | char(9) | 主键 | 学号 | | cl\_id | char(6) | 外键，NOT NULL | 班号 | | s\_name | varchar(12) | NOT NULL | 学生姓名 | | s\_sex | char(2) | NOT NULL | 学生性别 | | s\_age | int | NOT NULL | 学生年龄 | | s\_tel | char(11) | NOT NULL | 学生联系方式 |   表 2：Teacher   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | t\_id | char(8) | 主键 | 教师工号 | | d\_id | char(4) | 外键，NOT NULL | 专业代码 | | t\_name | varchar(12) | NOT NULL | 教师姓名 | | t\_sex | char(2) | NOT NULL | 教师性别 | | t\_age | int | NOT NULL | 教师年龄 | | t\_tel | char(11) | NOT NULL | 教师联系方式 |   表 3：Department   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | d\_id | char(4) | 主键 | 专业代码 | | d\_name | varchar(16) | NOT NULL | 专业名称 |   表 4：Class   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | cl\_id | char(6) | 主键 | 班号 | | t\_id | char(8) | NOT NULL | 班主任工号 | | d\_id | char(4) | 外键，NOT NULL | 专业代码 |   表 5：Course   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | c\_id | char(6) | 主键 | 课程号 | | c\_name | varchar(20) | NOT NULL | 课程名称 | | d\_id | char(4) | 外键，NOT NULL | 开课专业 | | credit | float | NOT NULL | 课程学分 | | hours | int | NOT NULL | 课程学时 |   表 6：Classroom   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | r\_id | varchar(8) | 主键 | 教室代号 | | r\_addr | varchar(20) | NOT NULL | 教室地点 |   表 7：sc (StudentCourse)   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | s\_id | char(9) | 主键，外键 | 学号 | | c\_id | char(6) | 主键，外键 | 课程号 | | score | float | NOT NULL | 课程成绩 | | date | date | NOT NULL | 成绩录入时间 |   表 8：tt (TeachingTask)   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | t\_id | char(8) | 主键，外键 | 班主任工号 | | c\_id | char(6) | 主键，外键 | 课程号 | | start\_week | int |  | 开课周数 | | end\_week | int |  | 结课周数 |   表 9：ttr (TeachingTaskClassroom)   |  |  |  |  | | --- | --- | --- | --- | | **列名** | **数据类型** | **完整性约束** | **含义** | | c\_id | char(6) | 主键，外键 | 课程号 | | t\_id | char(8) | 主键，外键 | 班主任工号 | | r\_id | varchar(8) | 主键，外键 | 教室代号 | | dayno | int | NOT NULL | 一周中第几天 | | classno | int | NOT NULL | 课程节数 |   在我所想的设定中，包含了各种联系：班级与班主任的联系属于一对一联系，班级与学生的联系属于一对多联系，而教室与教学任务安排属于多对多联系。  **2.2系统ER图**  由前述，整个管理系统包含 9 个实体和 10 个联系，具体的系统ER图如下所示：    **2.3数据库索引**  按照实际经验，针对经常修改，查看的实体：Student, Teacher, ttr分别建立了索引，以优化数据库的性能。具体而言，对 student 表中的 s\_name 属性、s\_sex 属性和 s\_tel 属性创建了索引；对 teacher 表中的t\_name 属性创建了索引；对 ttr 表中的 classNo 属性创建了索引。  在Navicat中查看创建好的索引，如图所示：    **2.4数据库外模式**  在该管理系统中，创建视图（外模式）主要是满足成绩查询等功能的要求，同时在查询过程中方便得到复杂的统计结果（如平均成绩等）。目前，我为该管理系统设计了如下三个视图：   1. studStat   包括如下信息：学生学号，学生姓名，以及该学生目前修读的总学分，目前的平均学分绩。该视图只用于查询，其构造语句如图所示：     1. studNoScoreCourse   包括如下信息：学生学号，学生姓名，目前暂无成绩的课程门数，没出成绩的课程总学分，该视图只用于查询，其构造语句如图所示：     1. courseStat   包括如下信息：课程的课程号、课程名称，选修该课程的学生成绩的最高分、最低分以及平均分，该视图只用于查询，其构造语句如图所示：    **2.5批量数据生成与添加**  为了测试数据库性能，需要向数据库中注入一定规模的数据。由于在现实场景中，学生和成绩这两个实体的数量往往多于其余实体的数量，因此在本实验中，我通过Python程序分别生成了1000个Student和Score实体，将其保存在.xls文件中，在需要时可以写入数据库，便于后续进行数据库的性能测试。  生成的数据格式如下：    图 1：生成的Student实体    图 2：生成的sc实体  在生成完毕后，可以通过Navicat数据库管理软件进行数据写入。具体流程如下：          导入完成后，效果如下：     1. **数据库应用开发**   **3.1应用界面简介**  按照前述的思想使用PyQt5设计管理系统，完成后得到sms.ui文件。使用pyuic5将.ui文件转化为.py文件，再另外编写一个main函数即可启动程序界面。  下面按顺序来介绍应用界面的每个部分。   1. **菜单栏**   菜单栏可以选择具体执行操作的两部分，分别是基础信息部分和其他功能部分。如图所示。     1. **基础信息部分**   在打开程序时，系统默认进入的部分也就是学生的基础信息部分。如图所示（默认表部分内容为空，需要在SELECT操作下点击“确认”才能显示数据内容）：    基础信息部分主要完成对实体数据的查看，另外我完成了对实体的增，删，查三种功能的支持。如上图所示，可以在对应的框中输入数据，以便完成增，删，查三种功能。具体如何执行将在3.2部分详细阐述。   1. **其他功能部分**   其他部分包含了完成五种不同功能的模块：学生选课，学生成绩录入，显示学生成绩，显示某门课程的所有学生成绩，以及教学任务安排和教室安排。其界面如图所示（以教室安排为例）：    关于该部分的具体功能演示，放在3.2部分详细阐述。  **3.2基本功能与功能测试**  以下具体讲解该系统实现的功能。   1. **基本查询功能**   用户在界面中选择进行SELECT操作，在输入任意查询条件的组合后，系统后端会根据按键反馈，组合成相应的SQL查询语句，执行后，系统返回MySQL数据库中满足条件的全部元组，同时在Python后端中打印输出用户所执行的操作，如图所示，用户在教师页面选中SELECT操作并点击确认，前端与后端界面的显示如下：       1. **模糊查询**   在基本的查询功能之外，用户也可以输入“%”表示通配符，进行模糊匹配查询。以学生信息查询为例，如图所示，学号部分输入“1001%”，系统会查询所有学号以“1001%”开头的元组并返回：      当输入的信息在表中没有与其匹配的元组时，系统则会提示报错：    另外，在点击“清除”按钮后，可以清除该页面所有输入内容和显示区内容。   1. **插入（增加）功能**   插入功能允许用户向表中插入一行数据，插入的数据需满足数据库的完整性约束，如主键约束、外键约束（不能插入外键表中不存在的内容）、非空约束（不允许为空的属性都不能插入为空）等等，若插入的数据不满足完整性约束，将弹出提示信息。  在本系统中，插入功能一般要求所有输入框或者选择框都有相应的内容，（因为大部分属性都有NOT NULL的约束）。以教学任务安排表tt为例，选择教师编号“10012005”和课程号“110001”，设置开课星期为 4，结课星期为 12，点击确认，提示插入成功，如下图所示。    改而选中SELECT进行查询，同样能查到对应的结果。    若是不满足插入条件，如：某一栏为空，则会弹窗显示错误提示，如图所示：    另外，为了部分地解决插入过程中可能出现的问题，同时提升程序的用户友好度，我在部分页面将输入框更改为了选择框，这样就有效地满足了外键约束（不会出现没有的选项），例如在教学任务安排的界面中，课程号和教师编号就是由选择框直接选择，而不是由用户输入，这样可以有效地降低输入错误率。  在代码方面，插入的代码会相对复杂一些。与查询和删除不一样，插入过程中要求每一项都是确认的，要罗列插入值，所以查询中的 where 条件查询函数是不可复用的，因此对于每一种插入，都西药重新书写SQL插入语句的生成函数。以insert\_tt\_sql()函数为例，当判断 index = 2，即需要进行插入操作时，执行插入函数insert()，在每次执行该 sql 语句之后都需要进行“commit”操作。其具体的代码如下：     1. **删除功能**   与查询相似，用户可以通过完整的属性值来删除某一特定元组，也可以只输入部分属性的值来删除所有满足该条件的元组（包括模糊匹配功能）。因此，此处删除所使用的 where 条件可以复用查询的 where 条件。若用户的删除操作破坏了完整性约束，则系统会报错，主要体现在外键约束带来的错误，例如当某个教室被安排了教学任务删除该教室就会报错。对于删除操作，在用户选择执行时，首先会提示确认是否要删除，若选择 Yes，则运行删除 SQL语句，并检测用户想要执行的删除是否能够正确执行。若是不满足外键约束，则弹出错误提示。如图所示：       1. **连接查询**   连接查询其实在实验中的许多部分都有所体现，最典型的就是相对复杂的查询功能，如学生成绩单和课程成绩统计。这两项除了需要创建视图来获取统计数据外，都需要对sc、student、course 三个表进行连接来得到更详细的信息。这两个功能的实现代码如下图所示：      这两部分的具体运行结果如图：       1. **嵌套查询**   嵌套查询在复杂功能中也具有很高的重要性。如，在查询班级情况时，为防止班级因为存在学生而被漏统计，就将满足条件的班级元组作为表A，与student表中的班级-学生数统计结果进行外连接，再在“班级学生数”这一项中用“0”来代替全部的null。查询全部的班级及其相关信息，其运行结果如图所示，这里有2个班级的学生人数为0。     1. **分组查询**   如上述嵌套查询一样，对于分组查询，在实验中查询班级时，引入班级学生数的概念，首先要将学生按照班级进行划分并且用“count(\*)”进行统计，具体来说就是获得相应班级的学生数。例子如嵌套查询所示。  **3.4系统优化**  正如前文所述，我为比较常被修改的实体类中的特定属性增加了索引。如： student表中的s\_name属性、s\_sex属性和s\_tel属性创建了索引；teacher表中的t\_name属性创建了索引；ttr表中的classNo属性创建了索引。索引方式均为BTREE。  为测试创建索引对性能的优化，在MySQL中对student表的s\_name属性进行索引前后，测试如下语句的执行时间：  SELECT s\_name from student;  在未创建索引时，结果如图所示:    创建索引后，结果如图：    可以看出，创建索引对提升查询速度产生了一定效果，但由于数据总量仍不够大，因此效果不算明显。 |

四、实验结论（总结实验发现及结论）

|  |
| --- |
| 1.通过本次实验，我强化掌握了数据建模的思想和方法，并体会到良好的数据库设计对后续数据库开发的重要作用。  2.更加熟练地掌握了SQL语言的使用。包括数据库的建立、视图和索引的建立、表的建立、插入、删除、查询等方面，同时我也更加掌握到了数据库完整性约束的作用和重要性（尤其是外键约束）。  3.初步掌握了 python 环境下的嵌入 SQL 语言的使用和动态构建 SQL 语句的方法，构建数据库应用界面的过程也让我熟悉了 PyQt5的使用。  4.了解了数据库对于大规模系统构建的重要性。以此实验构建的数据库，后续可以进行丰富的拓展，这让我体会到了软件构造的复杂性与乐趣。与此同时，系统中涉及数据库的部分也让我知道要实现更加复杂的功能，还需要进一步对数据库进行探索和学习。 |