目录

题目背景

- SQL是什么
  - SQL，指结构化查询语言，全称是 Structured Query Language。
  - SQL 可以让你访问和处理数据库。
- SQL能做什么
  - SQL可以面向数据库查询
  - SQL可以向数据库插入新的记录
  - SQL可以从数据库中删除记录
  - SQL可以修改数据库
  - ……

项目简介

- 在**Linux**下实现命令行交互的简易SQL——mySQL
- 具体的，项目二要求实现：从命令行输入一条SQL语句（指令），解析该语句，执行相应的数据库操作，返回对应的结果。
- 要求实现的指令有：
  - CREATE TABLE
  - DROP TABLE
  - INSERT INTO
  - DELETE
  - SELECT
  - 授权命令: GRANT
  - 收回权限命令: REVOKE

- 要求实现的功能
  - 登录管理：访问数据库需要密码
  - 权限管理：除非经过授权(grant)，否则用户不能访问其他用户创建的table，权限也可以取消

项目要求

- 需要在Linux下完成本次实验
- 数据库中所有TABLE均以TXT文件的形式保存在本地，所有对数据库的修改需要更新到数据库文件中
- 还需要维护一个文件表示所有合法的用户名列表和对应的密码
- 此外，还需要维护一个(些)文件辅助记录每张表上各种操作的权限

项目要求

- 运行你的程序，应该首先进入命令行界面，然后输入mySQL进入到数据库程序，开始执行指令，直到输入quit指令，退出mySQL程序。注意，这里的“~$”以及“(mysql)==>”都是由你的程序输出，并非系统自带的软件或函数功能。

```
~$ mySQL
(mysql)==>login:abc
(mysql)==>password:*******
(mysql)==>quit
```

CREATE

- CREATE TABLE 语句，格式为：
  - CREATE TABLE name (column1,column2,…,columnT) TO file
    - 创建一个TABLE，TABLE的名字是name，共有T列，其中columni为第i列的属性名
    - 所有属性用括号包含，不同的属性名以逗号隔开
    - file为TABLE在本地存储的文件名
    - 例如 CREATE TABLE Student (学号,姓名,专业) TO student.txt
  - CREATE TABLE name FROM filename
    - 从一个已经存在的数据库文件中读取数据创建TABLE
    - 例如 CREATE TABLE Student FROM student.txt

CREATE

```
(mysql)==> CREATE TABLE Student (学号, 姓名, 专业) TO Student.txt
(mysql)==> quit
$ cat Student.txt
ID      学号    姓名    专业
```

- 如上所示，创建一个名为Student的表格，生成Student.txt文件。
- ID列是系统自动添加的。
- 需要注意的是，一个数据库文件不能被多个TABLE共享，创建TABLE的时候不能和已经在数据库中的文件冲突。

DROP和TABLE LIST

- DROP TABLE 语句，格式为：
  - DROP TABLE name
    - 从数据库中删除名为name的TABLE
    - 对应的文件也删掉

- TABLE LIST 语句，格式为：
  - TABLE LIST
    - 为了方便查看，使用TABLE LIST打印当前用户所能访问的表和权限

DROP和TABLE LIST

```
(mysql1)==> CREATE TABLE Student (学号, 姓名, 专业) TO Student.txt
(mysql1)==> CREATE TABLE Lecture (课程编号, 课程名称, 任课老师) TO Lecture.txt
```

```
(mysql)==> TABLE LIST
   total:2
      Student: (3,0) [学号, 姓名, 专业] DROP, INSERT, DELETE, SELECT [OWNER]
      Lecture: (3,0) [课程编号, 课程名称, 任课老师] DROP, INSERT, DELETE, SELECT [OWNER]
(mysql)==>
```

TABLE LIST展示当前用户所能访问的表集合，包括表格的名称，长度（列，行），属性列表以及操作权限

```
(mysql)==> DROP TABLE Lecture
(mysql)==> TABLE LIST
   total:1
      Student: (3,0) [学号, 姓名, 专业] DROP, INSERT, DELETE, SELECT [OWNER]
(mysql)==>
```

删除Lecture表，所以只剩下Student

INSERT

- INSERT INTO 语句，格式为：
  - INSERT INTO name VALUES (value1,value2,…,valueT)
    - 向TABLE name里插入一行，共T个属性的值，T应该与name的列数一致
    - 所有属性值用括号包含，不同的属性值以英文逗号隔开
    - 例如 INSERT INTO Student VALUES (170000001,王二小,计算机科学与技术)
  - INSERT INTO name (column1,column2,…) VALUES (value1,value2,…)
    - 向TABLE name里插入一行，但是仅指定的列有值，value与column对应
    - 缺省的列应设置为默认值
    - 例如 INSERT INTO Student (学号,姓名) VALUES (170000001,王二小)

INSERT

```
(mysql)==> INSERT INTO Student VALUES (170000001, 王二小, 计算机科学与技术)
(mysql)==> INSERT INTO Student (学号, 姓名) VALUES (170000002, 陈独秀)
(mysql)==> quit
$cat Student.txt
ID      学号    姓名    专业
1       170000001       王二小  计算机科学与技术
2       170000002       陈独秀
```

缺省的属性默认为空

- 向Student插入记录，对应文件中也插入记录

DELETE

- DELETE 语句，格式为：
  - DELETE FROM name WHERE column = value
    - 从TABLE name里删除若干行
    - 删除的行满足条件 column = value
    - 例如 DELETE FROM name WHERE 姓名 = 王二小
  - DELETE * FROM name
    - 从TABLE name里删除所有行
    - 注意这里与DROP TABLE的区别是保留TABLE的结构，没有删除TABLE

DELETE

```
(mysql)==> INSERT INTO Student VALUES (170000001, 王二小, 计算机科学与技术)
(mysql)==> INSERT INTO Student (学号, 姓名) VALUES (170000002, 陈独秀)
(mysql)==> quit
$cat Student.txt
ID      学号            姓名    专业
1       170000001       王二小  计算机科学与技术
2       170000002       陈独秀
$mySQL
(mysql)==> DELETE FROM Student WHERE 姓名 = 王二小
(mysql)==> quit
$cat Student.txt
ID      学号            姓名    专业
1       170000002       陈独秀
$
```

删除一条记录

注意ID的变化

SELECT

- SELECT 语句，格式为：
  - SELECT column1,column2,… FROM name
    - 从TABLE name里选择若干列展示
    - 不同列以逗号‘,’隔开
    - 例如 SELECT 学号,姓名 FROM Student
  - SELECT * FROM name
    - 从TABLE name里选择所有列展示，即展示整个TABLE
    - 例如 SELECT * FROM Student

SELECT

```
(mysql)==> SELECT 学号, 姓名 FROM Student
学号              姓名
170000001         王二小
170000002         陈独秀
(mysql)==> SELECT * FROM Student
ID      学号              姓名     专业
1       170000001         王二小   计算机科学与技术
2       170000002         陈独秀
(mysql)==>
```

查询表中记录的若干属性，*代表查看全部属性

SELECT

- SELECT 语句增加关键字，格式为：
  - SELECT DISTINCT column1,column2,… FROM name
    - 在TABLE name中，一列可能会有重复的值
      DISTINCT关键字表示只展示不同的值
    - 例如 SELECT DISTINCT 专业 FROM Student
  - SELECT * FROM name ORDER BY column1,column2,… ASC|DESC
    - 对返回的查询结果按某些列进行排序展示
    - 如果排序的条件有多列，以逗号‘,’隔开， ASC表示升序，DESC表示降序
    - 例如 SELECT * FROM Student ORDER BY 学号 ASC

SELECT

```
(mysql)==> SELECT * FROM Student ORDER BY 学号 DESC
ID      学号              姓名    专业
3       170000003         张三    软件工程
2       170000002         陈独秀
1       170000001         王二小  计算机科学与技术

(mysql)==> SELECT DISTINCT 专业 FROM Student
专业
计算机科学与技术
软件工程
(mysql)==>
```

按学号降序排列

查看不同的专业

SELECT

- SELECT 语句增加关键字，格式为：
  - SELECT column1,column2,… FROM name WHERE column = value
    - 在TABLE name中，选择若干列进行展示
    - 在这些列中，只展示满足条件 column = value的行
    - 例如 SELECT 专业 FROM Student WHERE 姓名 = 王二小
  - SELECT column1,column2,… FROM name TO file
    - 将查询的结果写入文件file中
    - 写入文件需要保持TABLE的结构，即能以写入后的file生成新的TABLE（CREATE TABLE table_name FROM file）
    - 例如 SELECT * FROM Student WHERE 专业 = 计算机 TO 计算机系学生名单.txt

SELECT

按姓名查询

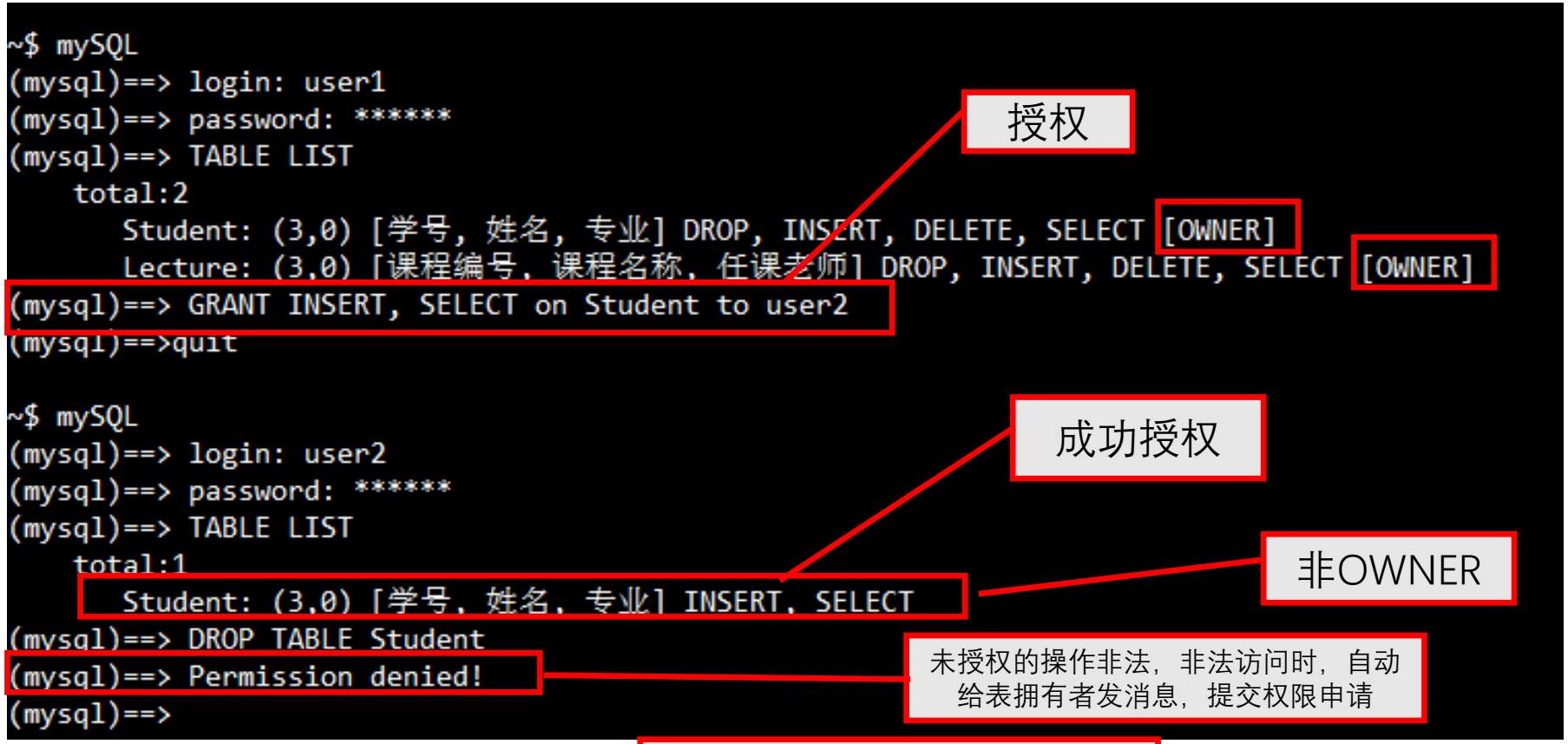
```
(mysql)==> SELECT * FROM Student WHERE 姓名 = 陈独秀
ID      学号            姓名    专业
2       170000002       陈独秀
(mysql)==> SELECT * FROM Student WHERE 姓名 = 陈独秀 TO Class1.txt
(mysql)==> quit
$ cat Class1.txt
ID      学号            姓名    专业
2       170000002       陈独秀
```

将查询结果保存到指定文件

GRANT

- GRANT 语句用于授予权限
  - GRANT <权限列表> on <表名> to <用户列表>
- <权限列表>
  - DROP : 删除表的权限
  - INSERT : 插入行的权限
  - DELETE : 删除行的权限
  - SELECT : 查询的权限
- <用户列表>:
  - 用户名
  - public 所有合法用户持有所授权限
- 权限的授予者必须已经持有相应的权限，权限可以传递

GRANT


~$ mySQL
(mysql)==> login: user1
(mysql)==> password: ******
(mysql)==> TABLE LIST
total:2
Student: (3,0) [学号, 姓名, 专业] DROP, INSERT, DELETE, SELECT [OWNER]
Lecture: (3,0) [课程编号, 课程名称, 任课老师] DROP, INSERT, DELETE, SELECT [OWNER]
(mysql)==> GRANT INSERT, SELECT on Student to user2
(mysql)==>quit
~$ mySQL
(mysql)==> login: user2
(mysql)==> password: ******
(mysql)==> TABLE LIST
total:1
Student: (3,0) [学号, 姓名, 专业] INSERT, SELECT
(mysql)==> DROP TABLE Student
(mysql)==> Permission denied!
(mysql)==>
授权
成功授权
非OWNER
未授权的操作非法，非法访问时，自动给表拥有者发消息，提交权限申请


~$ mySQL
(mysql)==> login: user1
(mysql)==> password: ******
(mysql)==> [System information] user2 requested DROP TABLE permission on table Student at 14:00 April 9, grant permission? (Y/N)
(mysql)==> Y
(mysql)==> DROP TABLE permission on table Student granted to user2 successfully
User1下次登录时，自动报告权限申请

GRANT


~$ mySQL
(mysql)==> login: user1
(mysql)==> password: ******
(mysql)==> TABLE LIST
total:2
Student: (3,0) [学号, 姓名, 专业] DROP, INSERT, DELETE, SELECT [OWNER]
(mysql)==> GRANT INSERT on Student to user2
(mysql)==>quit
~$ mySQL
(mysql)==> login: user2
(mysql)==> password: ******
(mysql)==> TABLE LIST
total:1
Student: (3,0) [学号, 姓名, 专业] INSERT
(mysql)==> GRANT INSERT on Student to user3
(mysql)==> quit
~$ mySQL
(mysql)==> login: user3
(mysql)==> password: ******
(mysql)==> TABLE LIST
total:1
Student: (3,0) [学号, 姓名, 专业] INSERT
(mysql)==> quit
User1是拥有者
权限可以传递

REVOKE

- REVOKE 语句用于收回授权
  - REVOKE <权限列表> on <表名> from <用户列表>
- Example:
  - REVOKE SELECT on Student from U1, U2, U3
- <权限列表> 可以是 all ， 表示收回被收回人持有的所有权限
- 某一用户拥有权限的前提是，授予他权限的人仍有对应的权限
- 如果同一权限由不同的授权人两次授予同一用户， 用户在一次回收后仍保持授权

REVOKE

```
~$ mySQL
(mysql)==> login: user1
(mysql)==> password: ******
(mysql)==> TABLE LIST
    total:2
       Student: (3,0) [学号, 姓名, 专业] DROP, INSERT, DELETE, SELECT [OWNER]
(mysql)==> GRANT INSERT on Student to user2
(mysql)==>quit

~$ mySQL
(mysql)==> login: user2
(mysql)==> password: ******
(mysql)==> TABLE LIST
    total:1
       Student: (3,0) [学号, 姓名, 专业] INSERT
(mysql)==> GRANT INSERT on Student to user3
(mysql)==> quit

~$ mySQL
(mysql)==> login: user3
(mysql)==> password: ******
(mysql)==> TABLE LIST
    total:1
       Student: (3,0) [学号, 姓名, 专业] INSERT
(mysql)==> quit
```

User1是拥有者

权限可以传递

REVOKE

```
~$ mySQL
(mysql)==> login: user1
(mysql)==> password: ******
(mysql)==> REVOKE INSERT on Student from user2
(mysql)==>quit

~$ mySQL
(mysql)==> login: user2
(mysql)==> password: ******
(mysql)==> TABLE LIST
(mysql)==> quit

~$ mySQL
(mysql)==> login: user3
(mysql)==> password: ******
(mysql)==> TABLE LIST
(mysql)==> quit
```

取消了user2的授权
等于自动取消了
user3的授权

- 某一用户拥有权限的前提是，授予他权限的人仍有对应的权限

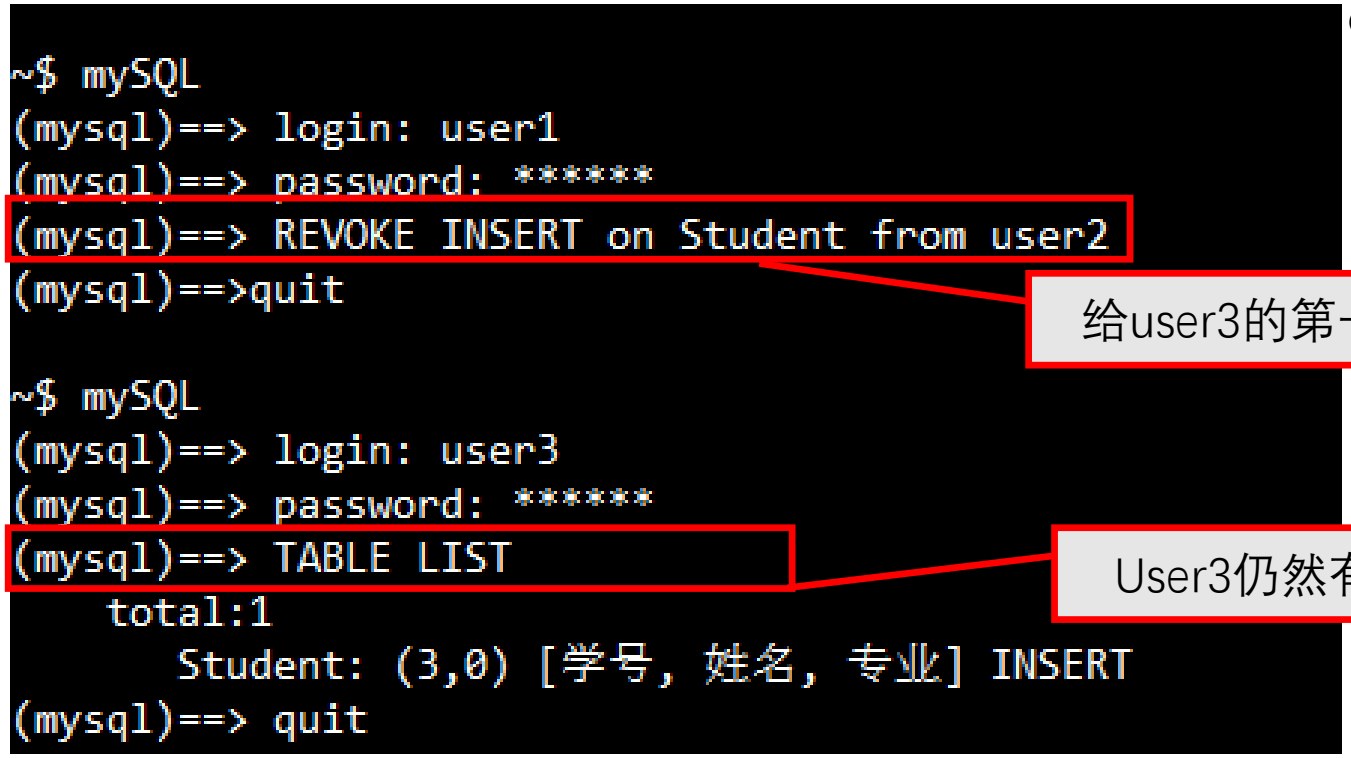
REVOKE

```
~$ mySQL
(mysql)==> login: user1
(mysql)==> password: ******
(mysql)==> TABLE LIST
    total:2
       Student: (3,0) [学号, 姓名, 专业] DROP, INSERT, DELETE, SELECT [OWNER]
(mysql)==> GRANT INSERT on Student to user2
(mysql)==> GRANT INSERT on Student to user3
(mysql)==>quit

~$ mySQL
(mysql)==> login: user2
(mysql)==> password: ******
(mysql)==> TABLE LIST
    total:1
       Student: (3,0) [学号, 姓名, 专业] INSERT
(mysql)==> GRANT INSERT on Student to user3
(mysql)==> quit

~$ mySQL
(mysql)==> login: user3
(mysql)==> password: ******
(mysql)==> TABLE LIST
    total:1
       Student: (3,0) [学号, 姓名, 专业] INSERT
(mysql)==> quit
```

给user3的第一次授权

给user3的第二次授权

REVOKE



给user3的第一次REVOKE

User3仍然有INSERT权限

- 如果同一权限由不同的授权人两次授予同一用户， 用户在一次回收后仍保持授权

题目要求

- 严格按照描述的指令格式，实现基本的指令功能，但是为了美观的虚线框可以省略或者自行设计修改，命令执行效果不得更改
- 命令行界面应简洁、美观，便于操作
- 充分考虑数据库操作中的非法操作，给出合理清晰的错误反馈
- 设计的程序应当分模块，各模块功能明确，有良好的代码风格。

格式约定

- CREATE TABLE Student (学号, 姓名, 专业) TO Student.txt

英文逗号后跟空格

- INSERT INTO Student VALUES (170000001, 王二小, 计算机科学与技术)
- INSERT INTO Student (学号, 姓名) VALUES (170000001, 王二小)
- DELETE FROM Student WHERE 姓名 = 王二小
- SELECT * FROM Student ORDER BY 学号 DESC

等号左右两边有空格

- SELECT * FROM Student WHERE 姓名 = 陈独秀 TO temp.txt
- GRANT INSERT, SELECT on Student to user2
- SELECT * FROM Student WHERE 学号 < (SELECT MAX (学号) FROM STUDENT)

符号左右两边有空格

1. 在前面你应该可以注意到，我们在多条语句中使用了WHERE关键字，它后面的内容表示一个条件判断，除了已经实现的‘=’判断外，还有
‘>’，‘<’，‘!=’，‘>=’，‘<=’，‘BETWEEN’，‘LIKE’，‘IN’等等，挑选你感兴趣的实现。

2. 更复杂的功能，函数。在某些数据库文件中，一些属性和数值有关，你可以对这些数值类的属性进行一些操作，例如求平均数、计数、求最大最小值等。

语句 SELECT MAX(学号) FROM Student 表示查询TABLE Student中学号最大的那一个。

更多的函数和功能自由发挥，函数返回格式和结果展示自行合理设计。

3. 语句的复合。实现了函数之后，我们就可以将一些函数作为WHERE关键字的判断条件，例如

   SELECT * FROM Student WHERE 学号 < (SELECT MAX(学号) FROM STUDENT)

4. 将多条语句按顺序写进文件，称为一个事务文件，直接运行文件,执行文件中所有操作
6. 打印系统权限图：显示表上权限的传递过程
7. 任何你想到的，好玩的功能，炫酷的界面，增强系统鲁棒性的方法，提高运行效率的算法等等

Thank you!

Q&A