

实验 2 数据库安全性/完整性定义与检查

实验目的：

- 1) 熟悉通过 SQL 对数据库进行安全性控制的方法，其中包括自主存取控制实验和审计实验。
- 2) 熟悉通过 SQL 对数据库进行完整性控制的方法，其中包括实体完整性、参照完整性、用户自定义完整性。
- 3) 熟悉并掌握数据库触发器的设计和使用方法。

实验内容：

2.1.1 自主存取控制实验

1) 实验内容与要求：定义用户、角色，分配权限给用户、角色，回收权限，以相应的用户名登录数据库验证权限分配是否正确。选择一个应用场景，使用自主存取控制机制设计权限分配。可以采用两种方案：

方案一：采用 SYSTEM 超级用户登录数据库，完成所有权限分配工作，然后用相应用户名登陆数据库以验证权限分配正确性；

方案二：采用 SYSTEM 用户登陆数据库创建两个部门经理用户，并分配相应的权限，然后分别用两个经理用户名登陆数据库，创建相应部门的 USER, ROLE，并分配相应权限。验证权限分配之前，请备份好数据库；针对不同用户所具有的权限，分别设计相应的 SQL 语句加以验证。

2) 实验重难点：定义角色，分配权限和回收权限，实现权限的再分配与回收。

3) 实验样例：创建数据库 company，包含 salary 工资表、employee 员工表。

Salary (name, salarise)

| name 姓名 | Salarise 工资 |
|---------|-------------|
| 刘星 | 3000 |
| 刘晨 | 3000 |
| 张三 | 3000 |
| 李勇 | 5000 |
| 李四 | 3000 |
| 王明 | 3000 |

Employee (name, number, dept)

| Name 姓名 | Number 员工号 | Dept 职位 |
|---------|------------|--------------|
| 刘晨 | 1 | Zhigong (职工) |
| 刘星 | 2 | Zhigong |
| 张三 | 3 | Zhigong |
| 李四 | 4 | Zhigong |
| 王明 | 5 | Zhigong |
| 李勇 | 6 | Daiban (代班) |

4) 示例

使用 SYSTEM 超级用户登录数据库创建两个部门经理用户，分配相应权限，然后使用经理用户名登录数据库创建相应部门的 USER、ROLE，分配相应权限。

创建财务部经理管理工资表，拥有查询、删除、修改工资的权限，创建人事部经理管理员工表，拥有删除员工、添加员工、更新员工信息的权限，并赋予这两个用户创建用户的权限，先创建两个用户并给与他们创建用户的权限：

```
create user 'finance'@'%' identified by '111111';
grant create user on *.* to 'finance'@'%';
create user 'hr'@'%' identified by '222222';
grant create user on *.* to 'hr'@'%';
```

然后给与他们更新相应表的权限，创建两个角色，然后给这两个角色分别赋予其对应的权限，并将角色权限给用户，同时加上 with grant option 选项，方便其用户赋予员工查询权限。

```
create role frole;
grant select,delete,update,insert on table salary to frole;
grant frole to finance with admin option;
create role hrrole;
grant select,delete,update,insert on TABLE employee to hrrole;
grant hrrole to hr with admin option;
```

在 finance 财务部经理下创建员工用户，给与查询权限：

```
create user 'f_user'@'%' identified by '.....';
grant select on TABLE salary to f_user;
```

验证如下：可以看到用户 f_user 仅有查询权限，不具有修改权限。

The screenshot shows a MySQL command window with the following SQL commands and their results:

```
1 use company;
2 select * from salary;
3 delete from salary where name='刘星';
```

An error message is displayed: [42000][1142] DELETE command denied to user 'f_user'@'localhost' for table 'salary'.

The output window shows the result of the SELECT query:

| | name | salaries |
|---|------|----------|
| 1 | 刘星 | 3000 |
| 2 | 刘晨 | 3000 |
| 3 | 张三 | 3000 |
| 4 | 李勇 | 5000 |
| 5 | 李四 | 3000 |
| 6 | 王明 | 3000 |

2.1.2 审计实验

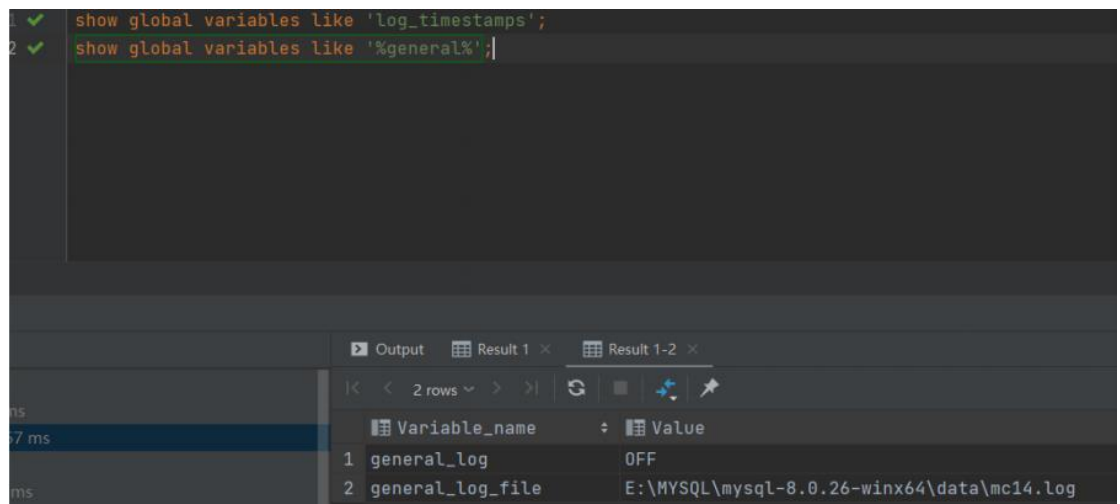
1) 实验内容与要求：掌握数据库审计的设置和管理方法，以便监控数据库操作，维护数据库安全。打开数据库审计开关，以具有审计权限的用户登录数据库，设置审计权限，然后以普通用户登录数据库，执行相应的 SQL 语句，验证审计设置是否有效，最后再以具有审计权限的用户登录数据库，查看是否存在相应的审计信息。

2) 实验重难点：合理的设置各种审计信息，一方面，为了保护系统重要的敏感数据，需要系统地设置各种审计信息，不能留有漏洞，以便随时监督系统使用情况，一旦出现问题，也便于追查；另一方面，审计信息设置过多，会严重影响数据库的使用性能，因此需要合理设置。

3) 示例

首先查看审计配置情况：

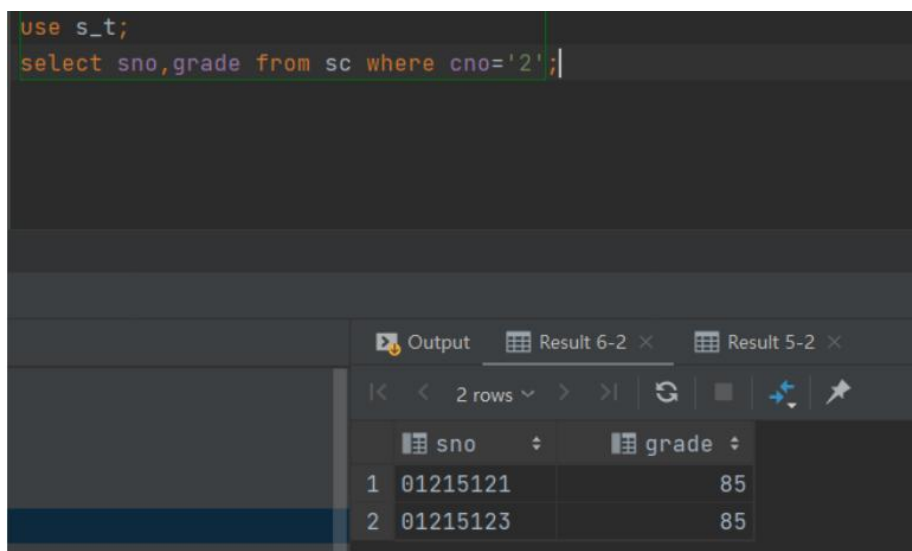
```
1 ✓ show global variables like 'log_timestamps';
2 ✓ show global variables like '%general%';
```



| Variable_name | Value |
|------------------|--|
| general_log | OFF |
| general_log_file | E:\MYSQL\mysql-8.0.26-winx64\data\mc14.log |

然后去对应地址找到审计文件。以没有审计权限的用户进行操作，以之前 S_T 数据库为例，查询 sc 表中选修 2 号课程的学生学号和成绩：

```
use s_t;
select sno,grade from sc where cno='2';
```



| sno | grade |
|----------|-------|
| 01215121 | 85 |
| 01215123 | 85 |

去查询相应位置的审计文件是否发生变化，然后再打开审计功能：

```
set global general_log =on;
```

再次执行 SQL 语句查询审计文件是否变化了。

2.2.1 实体完整性实验

1) 实验内容与要求：定义实体完整性，删除实体完整性。能够写出两种方式定义实体完整性的 SQL 语句：创建表时定义实体完整性、创建表后定义实体完整性。设计 SQL 语句验证完整性约束是否起作用。

2) 实验重难点：创建表时定义实体完整性，有多个候选码时实体完整性的定义。

3) 示例

创建一个学生表，属性包括：学号、姓名、性别、年龄、所在班级，其中学号、(姓名，所在班级) 为候选码，此处设计后者作为候选码，均不允许为空值，性别只能是男或女，年龄设置小于 25。

Student (sno, name, sex, age, class_num) , 其中 (name, class_num) 做为主码。

```
create table student (  
    sno char(12) not null unique,  
    name char(20) not null,  
    sex char(2) check (sex in ('男','女') ),  
    age int check ( age<=30 ),  
    class_num char(2) not null ,  
    constraint s_key primary key (name,class_num)  
    #主码有多个属性时只能定义在表级  
);
```

创建表之后定义完整性，限制班级号只能是 1-30:

```
alter table student add constraint c1 check ( class_num between 1 and 30);
```

验证完整性约束是否生效：插入一条学生记录，设置年龄为 31:

```
lab3> insert into student values (201901010102,'王阳','男',31,1)  
[2022-04-24 14:44:15] [HY000][3819] Check constraint 'student_chk_2' is violated.
```

会因为年龄不符合小于 25 的限制而插入失败。

2.2.2 参照完整性实验

1) 实验内容与要求：定义参照完整性，定义参照完整性的违约处理，删除参照完整性。写出两种方式定义参照完整性的 SQL 语句：创建表时定义参照完整性、创建表后定义参照完整性。

2) 实验重难点：创建表时定义参照完整性，参照完整性的违约处理定义。

3) 示例

创建一个班级表，属性包括姓名，学号，成绩，其中学号作为主码，姓名作为 student 表的外码。

创建表时定义参照完整性：

```
create table class (  
    sno char(12) primary key ,  
    name char(20) not null ,  
    grade int,  
    foreign key (name) references student(name)  
);
```

定义参照完整性的违约处理：

```
alter table class add constraint c1 foreign key (name) references student(name) on delete cascade ;|
```

删除 student 表中内容时级联删除 class 表中相应内容。请各位同学自行设计 SQL 语句进行验证违约处理是否生效。

删除参照完整性，再次删除 student 表中内容，无法删除的原因是默认的违约处理是禁止删除操作。

```
lab3> alter table class drop constraint c1  
[2022-04-24 15:18:43] completed in 211 ms  
lab3> insert into student values (201901010101, '李明', '男', 17, 1)  
[2022-04-24 15:18:53] 1 row affected in 156 ms  
lab3> insert into class values (201901010101, '李明', 90)  
[2022-04-24 15:18:58] 1 row affected in 172 ms  
lab3> delete from student where name='李明' and class_num='1'  
[2022-04-24 15:19:57] [23000][1451] Cannot delete or update a parent row: a foreign key constraint fails ('lab3'. 'class', COI  
[2022-04-24 15:19:57] [23000][1451] Cannot delete or update a parent row: a foreign key constraint fails ('lab3'. 'class', COI
```

2.2.3 用户自定义完整性实验

1) 实验内容与要求：针对具体应用语义，选择 NULL/NOT NULL、DEFAULT、UNIQUE、CHECK 等，定义属性上的约束条件。

2) 实验重难点：NULL/NOT NULL, DEFAULT, CHECK。

3) 示例

创建 source 课程记录表，属性包括学号，姓名，成绩，性别，所在小组，其中学号不能为空且唯一，姓名不能为空，成绩可以为空（期末考之后才会有成绩），性别不能为空，组号限制在 1-10，其中学号为主码。

```
lab3> create table source (  
    sno char(12) not null unique ,  
    name char(20) not null ,  
    grade int,  
    sex char(2) not null ,  
    zu_num char(1) check ( zu_num between 1 and 10),  
    primary key (sno)  
)  
[2022-04-24 15:29:31] completed in 546 ms
```


自行设计创建表之后定义完整性约束。设计 SQL 语句验证完整性约束：
插入时姓名为空会因为违反了完整性约束而无法插入。

```
lab3> insert into source values (201901010101,null,null,'男',1)
[2022-04-24 15:34:04] [23000][1048] Column 'name' cannot be null
```

2.3 触发器实验

- 1) 实验内容与要求：掌握数据库触发器的设计与使用方法，定义 BEFORE 触发器和 AFTER 触发器，能够理解不同类型触发器的作用和执行原理，验证触发器的有效性。
- 2) 实验重难点：利用触发器实现较为复杂的用户自定义完整性。
- 3) 示例

以 before 触发器为例,定义一个 teacher_salary 教师工资表,属性包括 name、job、salary，均不可为空，其中 name 作为主码。

```
lab3> create table t_s(
    name char(20) primary key ,
    job char(20) not null ,
    salary int not null
)
[2022-04-24 15:58:31] completed in 916 ms
```

Before 触发器定义如下:教授工资不得低于 5000,若低于则自动更改为 5000。

```
lab3> create trigger insert_or_update_salary
before insert on t_s
#referencing new row as newtuple
for each row
begin
    if (NEW.job='教授')and (NEW.salary<5000)
        then set NEW.salary=5000;
    end if;
end
[2022-04-24 16:08:17] completed in 132 ms
```

验证如下：

```
lab3> insert into t_s values ('王勇','教授',4500)
[2022-04-24 16:09:20] 1 row affected in 464 ms
```

| | name | job | salary |
|---|------|-----|--------|
| 1 | 王勇 | 教授 | 5000 |

触发器的执行是由触发事件激活, 执行顺序如下: 执行该表上的 before 触发器, 激活触发器上 sql 语句, 执行表上 after 触发器, 对于表上多个 before 触发器, 保持谁先创建谁执行原则, 也有的是按照触发器名字的排序执行。

实验要求:

- 1) 通过 SQL 对数据进行安全性/完整性控制。
- 2) 定义 BEFORE 触发器和 AFTER 触发器。
- 3) 能够理解不同类型触发器的作用和执行原理, 验证触发器的有效性。
- 4) 按以下要求填写实验报告, 记录所有的实验样例。

| | | | | |
|-------------------------------------|----|--|----|--|
| 《数据库系统概论》实验报告 | | | | |
| 题目: | 姓名 | | 日期 | |
| 实验环境: | | | | |
| 实验内容与完成情况: | | | | |
| 出现的问题: | | | | |
| 解决方案 (列出遇到的问题 and 解决办法, 列出没有解决的问题): | | | | |

学时分配: 1 学时