**《数据库原理及应用》实验报告**

题目: 安全与完整性

院系：计算机科学与工程学院

班级：210402

姓名：谢鸿

学号：20212345

**实验报告（五）**

**一、实验题目**

实验五 安全与完整性

**二、实验目的**

1. 掌握 使⽤SQL语⾔对⽤户、⻆⾊的维护⽅法。

2. 掌握 使⽤SQL语⾔对权限的分配、回收操作。

3. 理解 约束的触发机制。

4. 掌握 使⽤触发器实现复杂完整性。

**三、实验内容与实现**

**【实验要求一及实现】**

1. 创建**用户us1和ut1**,自行指定密码

**【答】**

SQL语句：

1. create user us1 identified by 'us20212345';
2. create user ut1 identified by 'ut20212345';

数据视图：



2、创建**角色rt\_read、rt\_insert、rt\_update**。

**【答】**

SQL语句：

1. CREATE ROLE rt\_read;
2. CREATE ROLE rt\_insert;
3. CREATE ROLE rt\_update;

数据视图：



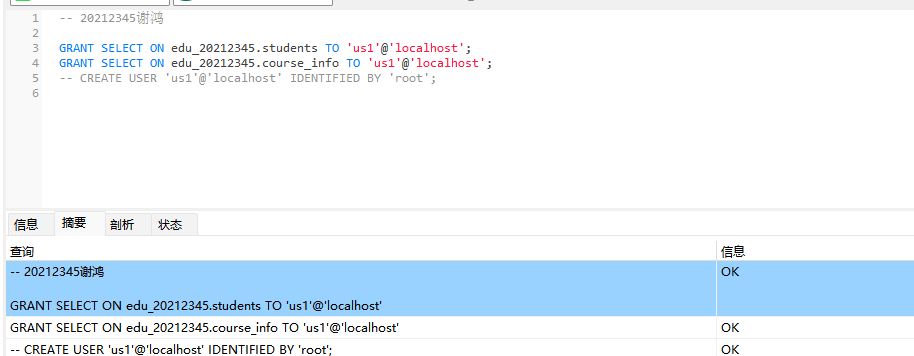
3、将**学生信息**与**开课信息**的**查询**权限授予us1。

**【答】**

SQL语句：

1. GRANT SELECT ON edu\_20212345.students TO 'us1'@'localhost';
2. GRANT SELECT ON edu\_20212345.course\_info TO 'us1'@'localhost';

数据视图：



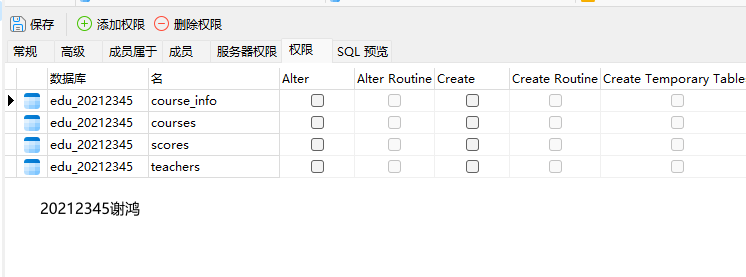
4、将**教师信息、课程信息、开课信息、成绩信息**的查询授权角色rt\_read。

**【答】**

SQL语句：

1. grant select on edu\_20212345.teachers to rt\_read;
2. grant select on edu\_20212345.courses to rt\_read;
3. grant select on edu\_20212345.course\_info to rt\_read;
4. grant select on edu\_20212345.scores to rt\_read;

数据视图：



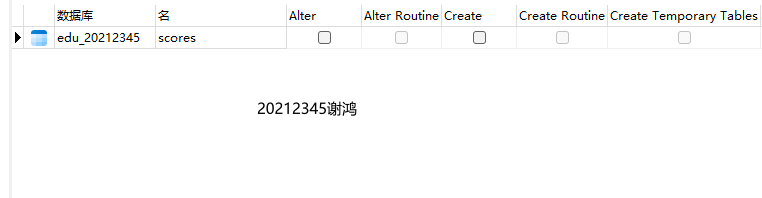
5、将**成绩信息**的插入权限授予角色rt\_insert。

**【答】**

SQL语句：

1. grant insert on edu\_20212345.scores to rt\_insert;

数据视图：



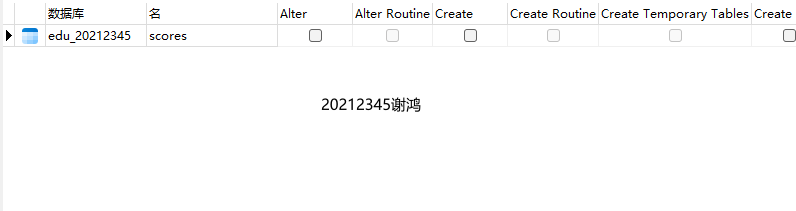
6、将**成绩信息**分数列的更新权限授予角色rt\_update。

**【答】**

SQL语句：

1. grant update on edu\_20212345.scores to rt\_update;

数据视图：



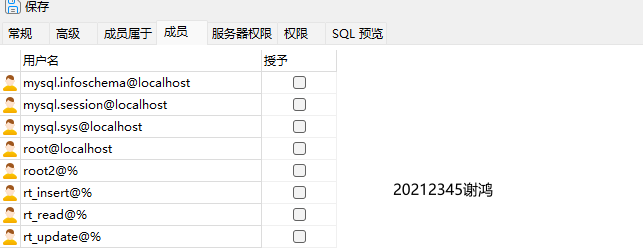
7、将用户ut1加入到角色rt\_read、rt\_insert和rt\_update中。

**【答】**

SQL语句：

1. GRANT rt\_read, rt\_insert, rt\_update TO ut1;

数据视图：



8、验证上述权限。

由于将用户ut1加入到角色rt\_read、rt\_insert和rt\_update中，验证角色授权时，登录ut1账户去验证。

(1)验证第4问将教师信息、课程信息、开课信息、成绩信息的查询授权角色rt\_read。

**【答】**

SQL语句：

1. show grants for ut1;

数据视图：



(2)验证第5问成绩信息的插入权限授予角色rt\_insert。

**【答】**

SQL语句：

1. INSERT INTO edu\_20212345.scores (student\_id, cs\_id, type, score) VALUES ('19040001', 12, '正考', 80.00);

数据视图：



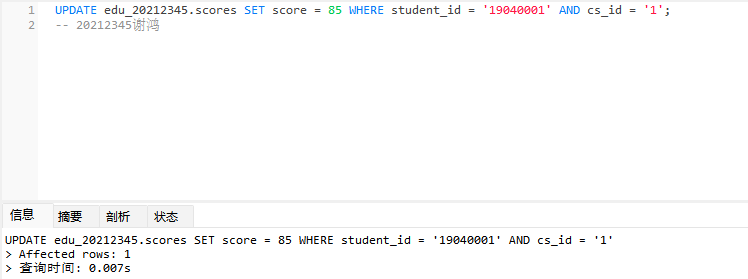
(3)验证第6问成绩信息分数列的更新权限授予角色rt\_update。

**【答】**

SQL语句：

1. UPDATE edu\_20212345.scores SET score = 85 WHERE student\_id = '19040001' AND cs\_id = '1';

数据视图：



9、如果要屏蔽教师查看到工资，需要在上述权限分配的基础上做哪些操作？请给出完整的操作过程。

**【答】**

1. #回收对于老师表的查看权限
2. REVOKE SELECT ON edu\_db.teachers FROM rt\_read;
3. #创建一个新的视图，从老师表中选择除工资列之外的其他列
4. CREATE VIEW vw\_teachers\_no AS SELECT id, name, sex, job\_title, dept\_id FROM teachers;
5. #授予 rt\_read 角色对该视图的查看权限
6. GRANT SELECT ON vw\_teachers\_no TO rt\_read;

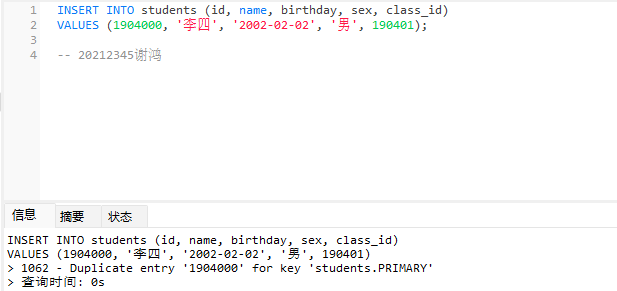


**【实验要求二及实现】**

1. 根据第一部分的结构，举例说明：
2. 主键约束的触发机制

**【答】**

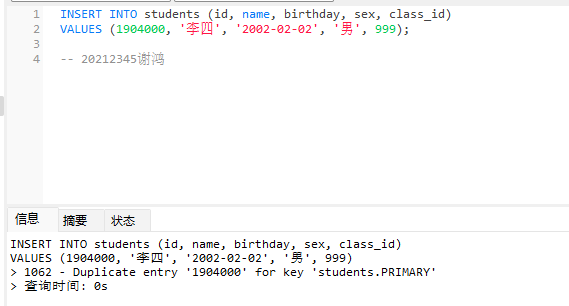
这个操作将失败，因为它试图使用一个已经存在的主键（学生编号为1904000）。数据库系统将拒绝这个插入操作并返回一个错误，提示主键约束被违反。这正是主键约束确保记录唯一性的方式。



1. 外键约束的触发机制

**【答】**

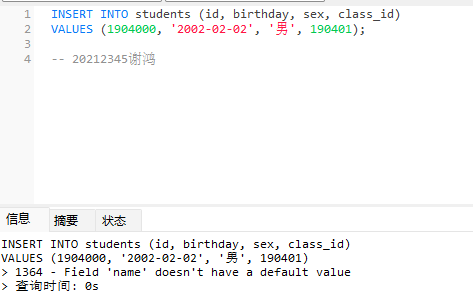
这个操作将失败，因为班级编号999在班级信息表中不存在。数据库系统将拒绝这个插入操作并返回一个错误，提示外键约束被违反。这正是外键约束确保数据完整性和一致性的方式。



1. 非空约束的出发机制

**【答】**

这个操作将失败，因为学生姓名（name）是非空的，但试图插入的记录未提供这个信息。数据库系统将拒绝这个插入操作并返回一个错误，提示非空约束被违反。这就是非空约束在实践中确保关键信息不被遗漏的方式。



1. 实现业务规则，并验证规则的有效性
2. 学生的年龄不低于15岁，不高于50岁。

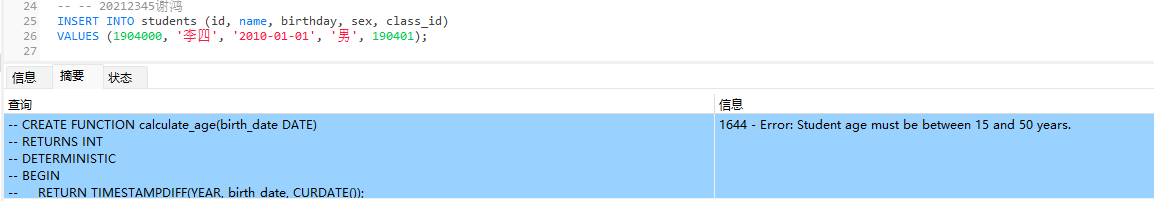
方法一：

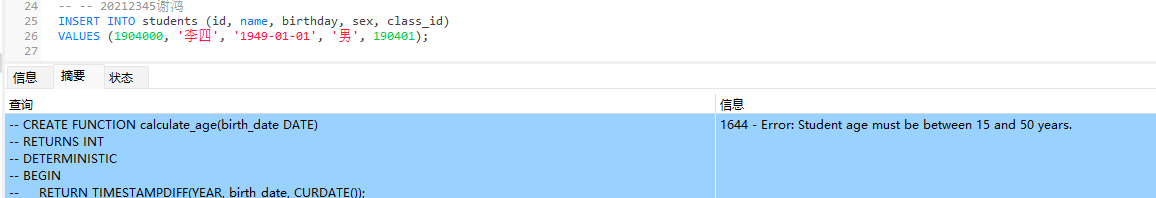
**【答】**

SQL语句：

1. CREATE FUNCTION calculate\_age(birth\_date DATE)
2. RETURNS INT
3. DETERMINISTIC
4. BEGIN
5. RETURN TIMESTAMPDIFF(YEAR, birth\_date, CURDATE());
6. END;
7. DELIMITER
8. CREATE TRIGGER check\_age\_before\_insert\_or\_update
9. BEFORE INSERT ON students
10. FOR EACH ROW
11. BEGIN
12. DECLARE age INT;
13. SET age = calculate\_age(NEW.birthday);
14. IF age < 15 OR age > 50 THEN
15. SIGNAL SQLSTATE '45000'
16. SET MESSAGE\_TEXT = 'Error: Student age must be between 15 and 50 years.';
17. END IF;
18. END;
19. DELIMITER ;

数据视图：





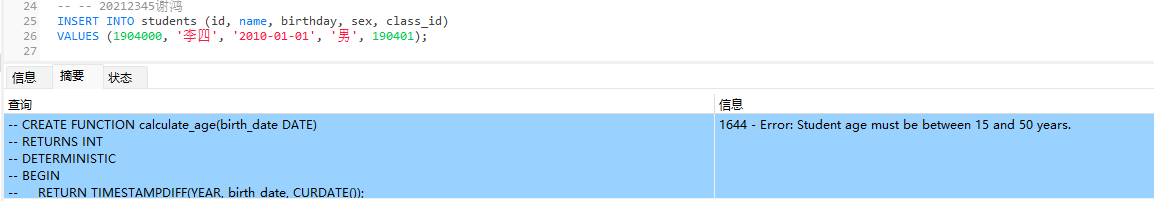
方法二：

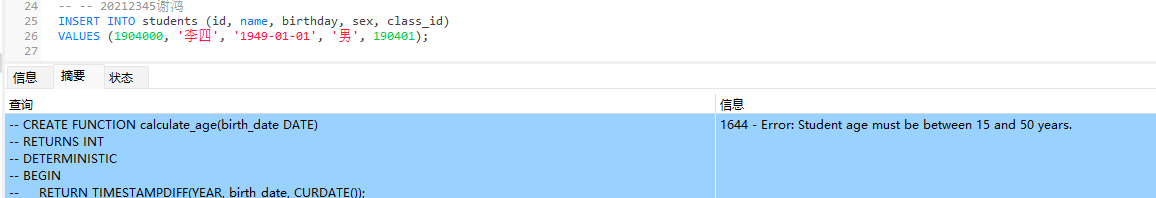
**【答】**

SQL语句：

1. DROP TRIGGER IF EXISTS tr\_add\_row\_check\_year\_2;
2. DELIMITER //
3. CREATE TRIGGER tr\_add\_row\_check\_year\_2
4. BEFORE INSERT
5. ON students
6. FOR EACH ROW
7. BEGIN
8. DECLARE year\_every INT DEFAULT 0;
9. SET year\_every = YEAR(NOW()) - YEAR(new.birthday);
10. IF (year\_every < 15) THEN
11. SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = '年龄不能小于15岁';
12. END IF;
13. IF (year\_every > 50) THEN
14. SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = '年龄不能大于50岁';
15. END IF;
16. END //
17. DELIMITER ;

数据视图：





1. 每学期学生选课的总学分不超过10分。

**【答】**

SQL语句：

1. DELIMITER //
2. CREATE TRIGGER check\_credit\_limit\_before\_insert
3. BEFORE INSERT ON scores
4. FOR EACH ROW
5. BEGIN
6. DECLARE total\_credits INT DEFAULT 0;
7. *-- 计算学生在同一学期的总学分*
8. SELECT SUM(c.credit) INTO total\_credits
9. FROM courses c
10. JOIN course\_info ci ON c.id = ci.course\_id
11. JOIN scores s ON ci.id = s.cs\_id
12. WHERE s.student\_id = NEW.student\_id AND ci.semester = (SELECT semester FROM course\_info WHERE id = NEW.cs\_id)
13. GROUP BY s.student\_id, ci.semester;
14. *-- 检查总学分是否超过10分*
15. IF total\_credits + (SELECT credit FROM courses WHERE id = (SELECT course\_id FROM course\_info WHERE id = NEW.cs\_id)) > 10 THEN
16. SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error: Total credits for a semester cannot exceed 10.';
17. END IF;
18. END;
19. //
20. DELIMITER ;

数据视图：



1. 记录成绩表被更改和被删除的操作记录流水(自定义流水表结构，要求记录修改的时间)。

**【答】**

SQL语句：

1. CREATE TABLE scores\_log (
2. audit\_id INT AUTO\_INCREMENT PRIMARY KEY,
3. operation\_type VARCHAR(10),  *-- 'UPDATE' 或 'DELETE'*
4. student\_id INT,
5. course\_id INT,
6. old\_score INT,  *-- 旧分数，仅在更新操作中使用*
7. new\_score INT,  *-- 新分数，仅在更新操作中使用*
8. operation\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  *-- 操作时间*
9. );

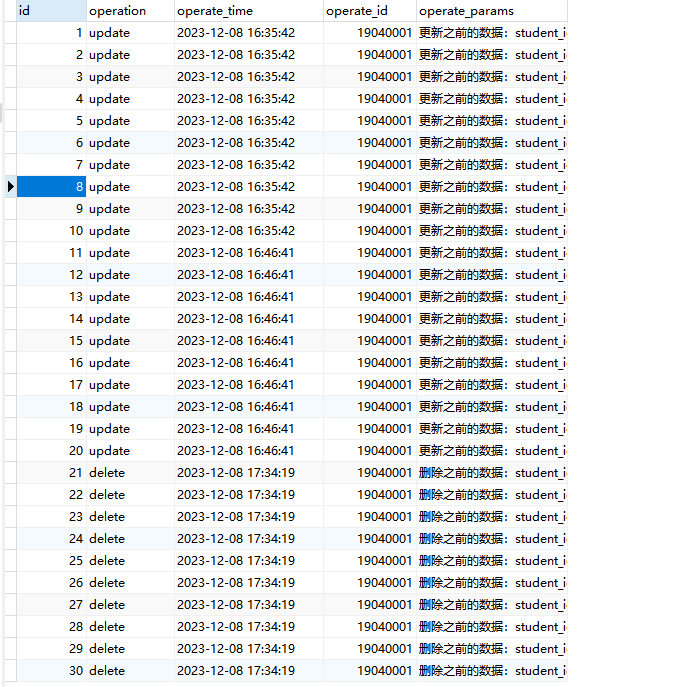
更新操作的触发器

1. CREATE TRIGGER tr\_update\_check\_scores
2. AFTER UPDATE ON scores FOR EACH ROW
3. BEGIN
4. INSERT INTO scores\_log(operation\_type, student\_id, course\_id, old\_score, new\_score, operation\_time)
5. VALUES('UPDATE', NEW.student\_id, NEW.cs\_id, OLD.score, NEW.score, now());
6. END;

删除操作的触发器

1. CREATE TRIGGER tr\_delete\_check\_scores
2. AFTER DELETE ON scores FOR EACH ROW
3. BEGIN
4. INSERT INTO scores\_log(operation\_type, student\_id, course\_id, old\_score, operation\_time)
5. VALUES('DELETE', OLD.student\_id, OLD.cs\_id, OLD.score, now());
6. END;

数据视图：



**四、实验问题总结与心得**

使用SQL语言进行用户和角色维护、权限分配和回收以及触发器实现复杂完整性的操作是非常重要的数据库管理技能。在进行这些实验过程中，我深入了解了以下几个方面：用户和角色维护：通过SQL语言，我学会了创建和管理用户和角色。这包括创建新用户、修改用户信息、重置密码、删除用户以及创建和分配角色。这些操作使我能够更好地管理数据库的访问权限，并确保只有授权的用户可以访问敏感数据。权限的分配和回收：使用SQL语言，我学会了授予和回收用户或角色的特定权限。这包括授予SELECT、INSERT、UPDATE、DELETE等操作表的权限，以及授予创建表、修改表结构和执行存储过程等高级权限。同时，我也学会了回收用户或角色的权限，以限制其对数据库的访问和操作。

约束的触发机制：在实验中，我深入理解了约束的触发机制。通过在数据库表上定义各种约束（如主键、外键、唯一约束、检查约束等），可以保证数据的完整性和一致性。当违反约束条件时，触发器会自动触发，并执行相应的操作，如拒绝插入、更新或删除操作，并返回错误信息。触发器实现复杂完整性：使用触发器，我能够实现更复杂的完整性约束。通过在数据库表上创建触发器，可以在特定事件（如插入、更新、删除）发生时执行自定义的操作。这使我能够实施更高级的数据验证和业务逻辑，确保数据的完整性和一致性。