数据库系统课程实验报告

实验名称:	触发器
实验日期:	2025.5.9
实验地点:	西部片区 4 号楼 104
提交日期:	2025.5.16

学号:	37220232203786
姓名:	潘腾凯
专业年级:	软工 2023 级
学年学期:	2024-2025 学年第二学期

1. 实验目的

- 理解和掌握 Mysql8.4 触发器的作用和工作原理
  - AFTER/BEFORE 触发器
  - 行级触发器
- 熟练掌握 Mysql8.4 触发器的设计方法
- 熟练掌握 Mysql8.4 触发器创建、 删除与查看的方法

2. 实验内容和步骤

1.关于 mysql8.4 触发器的两点知识:

（1） mysql8.4 只支持行级触发器， 不支持语句级触发器

（2） mysql8.4 使用修饰词 old 和 new 来代表修改前后的值，具体使用如下:

Trigger Event	OLD	NEW
INSERT	No	Yes
UPDATE	Yes	Yes
DELETE	Yes	No

2.Before 触发器的创建与验证

创建触发器的语法

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
FOR EACH ROW
```

BEGIN

-- Trigger body (SQL statements)

END; \g --触发器代码最后以\g 结尾，表示执行代码，如要中断执行，同时按 CTRL+C

（1） 创建部门表 dept(dno, dname)，其中， dno 为部门号， 定长为 2 的字符型，主码； dname 为部门名， 最大长度为 20 的变长字符型，非空

如下：

```
mysql> CREATE TABLE dept
(dno char(2) PRIMARY KEY,
dname varchar(20) NOT NULL);
Query OK, 0 rows affected (0.07 sec)

mysql> desc dept
    -> ;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| dno   | char(2)     | NO   | PRI | NULL    |       |
| dname | varchar(20) | NO   |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.08 sec)
```

（2） 创建 Teacher 表： Teacher(ID, job, Sal, dno)，其中， ID 为教工号，定长为 5 的字符型，主码； job 为职称，最大长度为 20 的变长字符型，非空； sal 为工资， 总长度为 7 的数值型，其中保留两位小数； dno 为部门号，定长为 2 的字符型，外码，引用 dept 表中的主码 dno

如下：

```
at line 3
mysql> CREATE TABLE teacher
        (ID CHAR(5) PRIMARY KEY,
        job VARCHAR(20) NOT NULL,
        sal DECIMAL(7,2),
        dno CHAR(2),
        FOREIGN KEY (dno) REFERENCES dept(dno));
Query OK, 0 rows affected (0.06 sec)

mysql> desc teacher;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| ID    | char(5)      | NO   | PRI | NULL    |       |
| job   | varchar(20)  | NO   |     | NULL    |       |
| sal   | decimal(7,2) | YES  |     | NULL    |       |
| dno   | char(2)      | YES  | MUL | NULL    |       |
+-------+--------------+------+-----+---------+-------+
4 rows in set (0.08 sec)
```

（3） 为 dept 表增加实验数据：（‘01’,’CS’),(‘02’,’SW’),(‘03’,’MA’)； 为 Teacher 表增加实验数据：（‘10001’,’教授’,3800,’01’),(‘10002’,’教授’,4100,’02’),(‘10003’,’副教授’,3500,’01’),(‘10004’,’助理教授’,3000,’03’）

如下：

```
mysql> INSERT INTO dept (dno, dname) VALUES
('01', 'CS'),
('02', 'SW'),
('03', 'MA');
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Teacher (ID, job, Sal, dno) VALUES
('10001', '教授', 3800.00, '01'),
('10002', '教授', 4100.00, '02'),
('10003', '副教授', 3500.00, '01'),
('10004', '助理教授', 3000.00, '03');
Query OK, 4 rows affected (0.04 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

（4） 在 Teacher 表上创建一个名为 tri_INSERT_SAL 的 BEFORE 行级触发器以实现如下完整性规则： 如果新增数据中教授的工资低于 4000 元，则自动改为 4000 元

注： 本步骤使用了 IF 语句， trigger body 的格式：

if（condition） then set new.sal =4000; --设置字段的值使用 SET…=;

end if;

--格式已经给出， 主要工作是将 condition 替换为具体的表达式

如下：

```
mysql> DELIMITER $$

CREATE TRIGGER tri_INSERT_SAL
BEFORE INSERT ON Teacher
FOR EACH ROW
BEGIN
    IF (NEW.job = '教授' AND NEW.Sal < 4000) THEN
        SET NEW.Sal = 4000;
    END IF;
END$$

DELIMITER ;
Query OK, 0 rows affected (0.04 sec)
```

因为触发器体内部使用分号（‘;‘），所以用 DELIMITER 临时改变 MySQL 的语句结束符为’$$’来避免冲突。

（5） 验证 tri_INSERT_SAL 触发器是否正常工作：

A.插入两条新数据(‘10005’,’教授’,3999,’02’)， (‘10006’,’教授’,4001,’03’)

B.发布查询语句： select * from teacher; 观察触发器是否按预期执行成功

```
mysql> INSERT INTO Teacher (ID, job, Sal, dno) VALUES
('10005', '教授', 3999, '02'),
('10006', '教授', 4001, '03');
Query OK, 2 rows affected (0.04 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM teacher;
+-------+----------+---------+-----+
| ID    | job      | sal     | dno |
+-------+----------+---------+-----+
| 10001 | 教授     | 3800.00 | 01  |
| 10002 | 教授     | 4100.00 | 02  |
| 10003 | 副教授   | 3500.00 | 01  |
| 10004 | 助理教授 | 3000.00 | 03  |
| 10005 | 教授     | 4000.00 | 02  |
| 10006 | 教授     | 4001.00 | 03  |
+-------+----------+---------+-----+
6 rows in set (0.09 sec)
```

触发器按预期执行成功

3.查看触发器

命令： show triggers…\g

```
SHOW TRIGGERS
    [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```

SHOW TRIGGERS lists the triggers currently defined for tables in a database (the default database unless a FROM clause is given). This statement returns results only for databases and tables for which you have the TRIGGER privilege. The LIKE clause, if present, indicates which table names (not trigger names) to match and causes the statement to display triggers for those tables. The WHERE clause can be given to select rows using more general conditions, as discussed in Section 28.8, "Extensions to SHOW Statements".

- show triggers\g ---显示当前数据库的所有触发器信息，包括名称和代码

```
+----------------+--------+---------+------------------------------------------------------------+
| Trigger        | Event  | Table   | Statement                                                  | Timing | Created                | sql_mode
                                                                | Definer              | character_set_client | collation_connection | Database Collation |
+----------------+--------+---------+------------------------------------------------------------+
| tri_INSERT_SAL | INSERT | teacher | BEGIN
   IF (NEW.job = '教授' AND NEW.Sal < 4000) THEN
      SET NEW.Sal = 4000;
   END IF;
END | BEFORE | 2025-05-16 11:34:06.60 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | pantengkai@localhost | utf8mb4              | utf8mb4_0900_ai_ci   | u
tf8mb4_0900_ai_ci |
+----------------+--------+---------+------------------------------------------------------------+
1 row in set (0.07 sec)
```

- show triggers [from | in] db_name \g ---显示 db_name 数据库里的触发器信息

```
mysql> show triggers in exp8;
+----------------+--------+---------+------------------------------------------------------------+
| Trigger        | Event  | Table   | Statement                                                  | Timing | Created                | sql_mode
                                                                | Definer              | character_set_client | collation_connection | Database Collation |
+----------------+--------+---------+------------------------------------------------------------+
| tri_INSERT_SAL | INSERT | teacher | BEGIN
   IF (NEW.job = '教授' AND NEW.Sal < 4000) THEN
      SET NEW.Sal = 4000;
   END IF;
END | BEFORE | 2025-05-16 11:34:06.60 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | pantengkai@localhost | utf8mb4              | utf8mb4_0900_ai_ci   | u
tf8mb4_0900_ai_ci |
+----------------+--------+---------+------------------------------------------------------------+
1 row in set (0.09 sec)
```

- like ‘pattern’ --这里的 pattern 是指表名， 非触发器名

特别注意：执行详细用法见： mysql8.4 手册的第 15 章之 15.7.7.40

SHOW TRIGGERS Statement

```
mysql> show triggers in exp8
like 'teacher';
+----------------+--------+---------+----------------------------------
-----------------------------------------------------------+------------
| Trigger        | Event  | Table   | Statement
                                                          | Definer
+----------------+--------+---------+----------------------------------
-----------------------------------------------------------+------------
| tri_INSERT_SAL | INSERT | teacher | BEGIN
    IF (NEW.job = '教授' AND NEW.Sal < 4000) THEN
        SET NEW.Sal = 4000;
    END IF;
END | BEFORE | 2025-05-16 11:34:06.60 | ONLY_FULL_GROUP_BY,STRICT_TRANS
tf8mb4_0900_ai_ci |
+----------------+--------+---------+----------------------------------
-----------------------------------------------------------+------------
1 row in set (0.10 sec)
```

4.触发器实现外码约束

（1） 设计触发器自动维持以下表间的外码约束： 删除 dept 表中 dno 的数据（如 dno=’03’） 后， teacher 表上引用该数据的记录也被自动删除。

如下：

```
mysql> DELIMITER $$

CREATE TRIGGER tri_delete_cascade_dept_teacher
AFTER DELETE ON dept
FOR EACH ROW
BEGIN
    DELETE FROM Teacher WHERE dno = OLD.dno;
END$$

DELIMITER ;
Query OK, 0 rows affected (0.03 sec)
```

Trigger 成功创建：

```
| Trigger                        | Event  | Table   | Statement
                                                                | Definer
+--------------------------------+--------+---------+----------------------------
-------------------------------------------------------------------------+--------
| tri_delete_cascade_dept_teacher | DELETE | dept    | BEGIN
    DELETE FROM Teacher WHERE dno = OLD.dno;
END                                          | AFTER  | 2025-05-17 11:16:11.73 | O
```

（2）验证（1）的效果。发布语句：

delete from dept where dno=’03’;\g

select * from teacher;\g --观察 dno=’03’的记录是否被删除， 从而验证触发器是否正常工作

需要注意的是，如果已定义了 teacher.dno 引用 dept.dno（即外码），那么触发器虽然工作，但因有外码约束，所以删除 teacher 表的相应数据失败（系统将会提示）

删除失败：

```
mysql> DELETE FROM dept
WHERE dno='03';
1451 - Cannot delete or update a parent row: a foreign key constraint fails (`exp8`.`teacher`, CONSTRAINT `t
eacher ibfk 1` FOREIGN KEY (`dno`) REFERENCES `dept` (`dno`))
```

5.触发器实现审计功能

设有选课表 SC(Sno,Cno,grade)： sno： 学号， 定长为 9 的字符型，非空； cno： 课程号， 定长为 3 的字符型，非空； grade： 课程成绩， 长度为 3 的整型，约束：介于 0 和 100 之间

样本数据集为： {(‘202315121’,’1’,92), (‘202315121’,’2’,85), (‘202315121’,’3’,88), (‘202315122’,’2’,90), (‘202315122’,’3’,80)}

要求设计一个触发器实现审计日志记录：当对表 SC 的 Grade 属性进行修改时，若分数增加了 10%及其以上，则将此次操作登记到下表中： SC_U(Sno, Cno, Oldgrade, Newgrade)，其中各字段含义和数据类型： sno 同 SC.sno， cno 同 SC.cno， 修改前成绩 oldgrade 和修改后成绩 newgrade 同 SC.grade。操作步骤如下：

（1） 创建 SC 表： SC(sno, cno, grade)

```
mysql> CREATE TABLE SC (
    Sno CHAR(9) NOT NULL,
    Cno CHAR(3) NOT NULL,
    Grade INT(3) CHECK (Grade BETWEEN 0 AND 100),
    PRIMARY KEY(Sno, Cno)
);
Query OK, 0 rows affected (0.03 sec)

mysql> desc sc
    -> ;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| Sno   | char(9) | NO   | PRI | NULL    |       |
| Cno   | char(3) | NO   | PRI | NULL    |       |
| Grade | int     | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
3 rows in set (0.09 sec)
```

（2） 插入样本数据

```
mysql> INSERT INTO SC (Sno, Cno, Grade) VALUES
('202315121', '1', 92),
('202315121', '2', 85),
('202315121', '3', 88),
('202315122', '2', 90),
('202315122', '3', 80);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

（3） 创建 SC_U 表： SC_U(Sno, Cno, Oldgrade, Newgrade)

```
mysql> CREATE TABLE SC_U (
    Sno CHAR(9) NOT NULL,
    Cno CHAR(3) NOT NULL,
    Oldgrade INT(3) CHECK (Oldgrade BETWEEN 0 AND 100),
    Newgrade INT(3) CHECK (Newgrade BETWEEN 0 AND 100)
);
Query OK, 0 rows affected (0.03 sec)

mysql> desc sc_u;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| Sno      | char(9) | NO   |     | NULL    |       |
| Cno      | char(3) | NO   |     | NULL    |       |
| Oldgrade | int     | YES  |     | NULL    |       |
| Newgrade | int     | YES  |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
4 rows in set (0.04 sec)
```

（4） 在 SC 表上创建一个名为 tri_update_sc 的 AFTER 行级触发器

```
mysql> DELIMITER $$

CREATE TRIGGER tri_update_sc
AFTER UPDATE ON SC
FOR EACH ROW
BEGIN
    IF NEW.Grade > OLD.Grade * 1.1 THEN
        INSERT INTO SC_U (Sno, Cno, Oldgrade, Newgrade)
        VALUES (OLD.Sno, OLD.Cno, OLD.Grade, NEW.Grade);
    END IF;
END$$

DELIMITER ;
Query OK, 0 rows affected (0.01 sec)
```

（5） 验证 tri_update_sc 触发器是否正常工作?

A. update sc set grade =94 where sno='202315121' and cno='2';

B. select * from sc_u;

```
mysql> UPDATE SC SET Grade = 94 WHERE Sno='202315121' AND Cno='2';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM SC_U;
+-----------+-----+----------+----------+
| Sno       | Cno | Oldgrade | Newgrade |
+-----------+-----+----------+----------+
| 202315121 | 2   |       85 |       94 |
+-----------+-----+----------+----------+
1 row in set (0.06 sec)
```

正常工作

6.删除触发器

删除命令： DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name;

- 删除所有的触发器
- 注意： 如果删除触发器所在的基本表，那么触发器将被自动删除

而无需显式删除

```
mysql> DROP TRIGGER IF EXISTS tri_INSERT_SAL;
DROP TRIGGER IF EXISTS tri_delete_cascade_dept_teacher;
DROP TRIGGER IF EXISTS tri_update_sc;
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

3. 实验总结

3.1 完成的工作

完成了 mysql84 各种功能（插入，删除，更新）触发器的创建、验证和删除。

3.2 对实验的认识

1.认识：

深入理解触发器的工作机制：触发器是响应特定事件（如 INSERT, UPDATE, DELETE）而自动执行的存储过程。实验中创建了不同类型的触发器（如 BEFORE INSERT, AFTER DELETE, AFTER UPDATE），以满足不同的业务规则需求。

学会了如何编写触发器：包括条件判断（如 IF 语句）、对新旧数据的访问（如 NEW 和 OLD 关键字）等技巧。

学会了如何使用触发器维护数据的一致性：例如，在更新成绩时如果分数增加了超过 10%，则将更改前后的信息记录到审计表中；或者确保教授的工资不低于某个标准值。

增强了对数据安全性的认识：通过触发器可以在一定程度上防止不合法的数据修改或删除操作，从而保护数据库的安全性和完整性。

2.思考题

2.1 如何实现在删除 dept 表中 dno 为’01’的数据后自动将 teacher 表中引用了该值（即’01’） 的 dno 字段值设置为 null? 请验证。

先删除原有外键约束:

```
mysql> ALTER TABLE Teacher DROP FOREIGN KEY teacher_ibfk_1;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

再添加 on delete set null 的约束

```
mysql> ALTER TABLE Teacher
ADD CONSTRAINT fk_dno
FOREIGN KEY (dno) REFERENCES dept(dno) ON DELETE SET NULL;
Query OK, 6 rows affected (0.07 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

验证:

```
mysql> DELETE FROM dept WHERE dno = '01';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM teacher;
+-------+----------+---------+------+
| ID    | job      | sal     | dno  |
+-------+----------+---------+------+
| 10001 | 教授     | 3800.00 | NULL |
| 10002 | 教授     | 4100.00 | 02   |
| 10003 | 副教授   | 3500.00 | NULL |
| 10004 | 助理教授 | 3000.00 | 03   |
| 10005 | 教授     | 4000.00 | 02   |
| 10006 | 教授     | 4001.00 | 03   |
+-------+----------+---------+------+
6 rows in set (0.07 sec)
```

3.3 遇到的困难及解决方法

无。