实验3 SQL数据完整性

实验目的:

1. 熟悉通过SQL进行数据完整性控制的方法

实验平台:

1. 操作系统：Windows 10
2. 数据库管理系统：MySQL 8.0.28

实验内容和要求:

1. 定义若干表，其中包括 primary key, foreign key 和 check 的定义。

```
CREATE TABLE depart_info
            (
             DepartNo   INT(10),
             DepartName VARCHAR(25),
             PRIMARY KEY (DepartNo)
            );
CREATE TABLE gender_info
            (
             GenderNo INT(10),
             Gender   VARCHAR(10),
             PRIMARY KEY (GenderNo),
             CHECK    (GenderNo <= 2)
            );
CREATE TABLE member_info
            (
             ID       INT(10),
             Name     VARCHAR(20),
             DepartNo INT(10),
             GenderNo INT(10),
             PRIMARY KEY (ID),
             FOREIGN KEY (DepartNo) REFERENCES
depart_info(DepartNo),
             FOREIGN KEY (GenderNo) REFERENCES
gender_info(GenderNo),
             CHECK    (GenderNo <= 2)
            );
```

2. 向表中插入数据，考察 primary key 如何控制实体完整性。

```
INSERT gender_info
VALUES (1, "Male");

INSERT gender_info
VALUES (1, "Female");
```

```
mysql> INSERT gender_info
    -> VALUES (1, "Male");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT gender_info
    -> VALUES (1, "Female");
ERROR 1062 (23000): Duplicate entry '1' for key 'gender_info.PRIMARY'
```

3. 删除被引用表中的行，考察 foreign key 中 on delete 子句如何控制参照完整性。

```
#此时，引用表中存在一条 GenderNo = 2 的记录
DELETE FROM gender_info
WHERE       GenderNo = 2;
```

```
mysql> DELETE FROM gender_info
    -> WHERE GenderNo = 2;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fail
s (`db03`.`member_info`, CONSTRAINT `member_info_ibfk_2` FOREIGN KEY (`GenderNo`) REFER
ENCES `gender_info` (`GenderNo`))
```

4. 修改被引用表中的行的 primary key， 考察 foreign key 中 on update 子句如何控制参照完整性。

```
#此时，引用表中存在一条 GenderNo = 2 的记录
UPDATE gender_info
SET    GenderNo = 0
WHERE  GenderNO = 2;
```

```
mysql> UPDATE gender_info
    -> SET    GenderNo = 0
    -> WHERE  GenderNO = 2
    -> ;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
 (`db03`.`member_info`, CONSTRAINT `member_info_ibfk_2` FOREIGN KEY (`GenderNo`) REFEREN
CES `gender_info` (`GenderNo`))
```

5. 修改或向表中插入数据，考察 check 子句如何控制校验完整性。

```
#表member_info的chek子句为：check(GenderNo <= 2)，其中GenderNo为第四
个字段
INSERT member_info
VALUES (3, "Caron", 3, 3);
```

```
mysql> INSERT member_info
    -> VALUES (3, "Caron", 3, 3);
ERROR 3819 (HY000): Check constraint 'member_info_chk_1' is violated.
```

6. 定义一个 assertion， 并通过修改表中数据断言如何控制数据完整性。

```
CREATE ASSERTION g_check
CHECK (NOT EXISTS
       (SELECT *
        FROM member_info
        WHERE DepartNo = 1
          AND GenderNo = 2
       )
       );
```

#MySQL 不支持 Assertion 语句。

7. 定义一个 trigger， 并通过修改表中数据考察触发器如何起作用。

```
#TRIGGER trig - 若插入的 ID < 0, 则将ID 设为 0
DELIMITER &&    #将结束符定义为&&
CREATE TRIGGER trig BEFORE INSERT
ON member_info FOR EACH ROW
BEGIN
    IF(NEW.ID < 0) THEN
        SET NEW.ID = 0;
    END IF;
END &&
DELIMITER ;    #将结束符重置为 ;
```

```
mysql> DELIMITER &&
mysql> CREATE TRIGGER trig BEFORE INSERT
    -> ON member_info FOR EACH ROW
    -> BEGIN
    -> IF(NEW.ID < 0) THEN
    -> SET NEW.ID = 0;
    -> END IF;
    -> END &&
Query OK, 0 rows affected (0.05 sec)

mysql> DELIMITER ;
mysql> INSERT member_info
    -> VALUES (-1, "Grace", 2, 2);
Query OK, 1 row affected (0.03 sec)

mysql> SELECT *
    -> FROM member_info;
+----+-------+----------+----------+
| ID | Name  | DepartNo | GenderNo |
+----+-------+----------+----------+
|  0 | Grace |        2 |        2 |
|  1 | Sam   |        1 |        1 |
|  2 | Amy   |        1 |        2 |
+----+-------+----------+----------+
3 rows in set (0.00 sec)
```

如图，INSERT 语句中 ID = -1 的记录被 TRIGGER 更改为 0 后插入表中。

```
#TRIGGER depart_delete 实现表 depart_info 与 member_info 间的级联删除
DELIMITER &&
CREATE TRIGGER depart_delete BEFORE DELETE #在删除depart_info的相关记录前
ON depart_info FOR EACH ROW
BEGIN
    DELETE FROM member_info                #先删除参照表member_info中的相应记录
    WHERE  DepartNo = OLD.DepartNo;
END &&
DELIMITER ;
```

```
mysql> DELIMITER &&
mysql> CREATE TRIGGER depart_delete BEFORE DELETE
    -> ON depart_info FOR EACH ROW
    -> BEGIN
    -> DELETE FROM member_info
    -> WHERE  DepartNo = OLD.DepartNo;
    -> END &&
Query OK, 0 rows affected (0.05 sec)

mysql> DELIMITER ;
mysql> DELETE FROM depart_info
    -> WHERE DepartNo = 1;
Query OK, 1 row affected (0.06 sec)

mysql> SELECT *
    -> FROM member_info;
+----+-------+----------+----------+
| ID | Name  | DepartNo | GenderNo |
+----+-------+----------+----------+
|  0 | Grace |        2 |        2 |
+----+-------+----------+----------+
1 row in set (0.00 sec)
```

如图，在删除被参照表depart_info中 DepartNo = 1 的记录前，member_info 中所有 DepartNo = 1 的记录都被删除，保持了参照完整性。

实验心得:

- TRIGGER 大大降低了级联删除/更新操作的繁琐程度

  不会因为MySQL的报错才想起来没删除参照表/被参照表的相关数据。

  只要对其中某一张表进行操作，“啪”的一下很快就级联操作完毕了。

- 慎用TRIGGER

  有时候误操作了，比如在本实验样例中，可能本来想从 depart_info 中删除 DepartNo = 2 的记录，结果输成了 DepartNo = 1，一个回车下去，两张表都灰飞烟灭。

  只有这时候会觉得 MySQL 关于参照完整性的错误提示非常的友好。

- 记得删除弃用的TRIGGER

  周一做了一半的实验，周三捡起来DELETE一下，顺便手滑输错了参数。参照表里的数据在TRIGGER的给力操作下一去不复返，真好。

  决定在每次重新开始操作前都浏览一下之前定义了哪些TRIGGER，推出前把容易误伤友军的TRIGGER都删掉。