Database-System Experiment-12

21307289 刘森元

1. 实验目的

通过实验使学生加深对数据完整性的理解，学会创建和使用触发器。

2. 实验环境

Macbook Pro, 14 inchs, 2021

Apple M1 Pro

macOS Sonoma 14.1.1

psql (PostgreSQL) 15.4 (Homebrew)

3. 实验内容

课内实验

1) 为 Worker 表（参照实验 11）建立触发器 T1，当插入或是更新表中数据时，保证所操作的记录的 sage 值大于 0。

```sql
CREATE OR REPLACE FUNCTION check_sage() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Sage <= 0 THEN
        RAISE EXCEPTION 'Sage must be greater than 0';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_sage_trigger
BEFORE INSERT OR UPDATE ON Worker
FOR EACH ROW
EXECUTE FUNCTION check_sage();
```

创建表 Worker

```
CREATE TABLE Worker (
    Number char(5),
    Name char(8) CONSTRAINT U1 UNIQUE,
    Sex char(1),
    Sage int CONSTRAINT U2 CHECK (Sage <= 28),
    Department char(20),
    CONSTRAINT PK_Worker PRIMARY KEY (Number)
);
```

可见 `Worker` 表已成功创建

```
school=# \d Worker
                 数据表 "public.worker"
    栏位    |     类型      | 校对规则 |  可空的  | 预设
------------+---------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u2" CHECK (sage <= 28)
```

建立触发器

```
CREATE OR REPLACE FUNCTION check_sage() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Sage <= 0 THEN
        RAISE EXCEPTION 'Sage must be greater than 0';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_sage_trigger
BEFORE INSERT OR UPDATE ON Worker
FOR EACH ROW
EXECUTE FUNCTION check_sage();
```

可见触发器 `T1` 已成功创建

```
school=# \d Worker
                 数据表 "public.worker"
    栏位    |      类型      | 校对规则 |  可空的   | 预设
------------+----------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u2" CHECK (sage <= 28)
触发器:
    check_sage_trigger BEFORE INSERT OR UPDATE ON worker FOR EACH ROW EXECUTE
FUNCTION check_sage()
```

2) 为 Worker 表建立触发器 T2，禁止删除标号为 00001 的 CEO。

```
CREATE OR REPLACE FUNCTION forbid_delete_ceo() RETURNS TRIGGER AS $$
BEGIN
    IF OLD.Number = '00001' THEN
        RAISE EXCEPTION 'CEO cannot be deleted';
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER forbid_delete_ceo_trigger
BEFORE DELETE ON Worker
FOR EACH ROW
EXECUTE FUNCTION forbid_delete_ceo();
```

创建触发器 T2

```
CREATE OR REPLACE FUNCTION forbid_delete_ceo() RETURNS TRIGGER AS $$
BEGIN
    IF OLD.Number = '00001' THEN
        RAISE EXCEPTION 'CEO cannot be deleted';
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER forbid_delete_ceo_trigger
BEFORE DELETE ON Worker
FOR EACH ROW
EXECUTE FUNCTION forbid_delete_ceo();
```

可见触发器 T2 已成功创建

```
school=# \d Worker
                 数据表 "public.worker"
    栏位    |      类型      | 校对规则 |  可空的   | 预设
------------+----------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u2" CHECK (sage <= 28)
触发器:
    check_sage_trigger BEFORE INSERT OR UPDATE ON worker FOR EACH ROW EXECUTE
FUNCTION check_sage()
    forbid_delete_ceo_trigger BEFORE DELETE ON worker FOR EACH ROW EXECUTE
FUNCTION forbid_delete_ceo()
```

3) Worker 表中的人员的编号是不可改变的，创建触发器 T3 实现更新中编号的不可改变性。

创建触发器 T3

```
CREATE OR REPLACE FUNCTION forbid_update_number() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Number <> OLD.Number THEN
        RAISE EXCEPTION 'Number cannot be changed';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER forbid_update_number_trigger
BEFORE UPDATE ON Worker
FOR EACH ROW
EXECUTE FUNCTION forbid_update_number();
```

可见触发器 T3 已成功创建

```
school=# \d Worker
                 数据表 "public.worker"
    栏位     |      类型      | 校对规则 |  可空的   | 预设
------------+----------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u2" CHECK (sage <= 28)
触发器:
    check_sage_trigger BEFORE INSERT OR UPDATE ON worker FOR EACH ROW EXECUTE
FUNCTION check_sage()
    forbid_delete_ceo_trigger BEFORE DELETE ON worker FOR EACH ROW EXECUTE
FUNCTION forbid_delete_ceo()
    forbid_update_number_trigger BEFORE UPDATE ON worker FOR EACH ROW EXECUTE
FUNCTION forbid_update_number()
```

4) 演示违反 T1 触发器的约束的插入操作。

演示插入数据

```
INSERT INTO Worker (Number, Name, Sex, Sage, Department) VALUES ('00002', 'John',
'M', -5, 'IT');
```

有错误信息

```
ERROR:  Sage must be greater than 0
背景:  PL/pgSQL function check_sage() line 4 at RAISE
```

可知触发器 T1 成功生效

5) 演示违反 T1 触发器的约束的更新操作。

演示更新数据

```
UPDATE Worker SET Sage = -10 WHERE Number = '00002';
```

有错误信息

```
ERROR:  Sage must be greater than 0
背景:  PL/pgSQL function check_sage() line 4 at RAISE
```

可知触发器 T1 成功生效

6) 演示违反 T2 触发器的约束的删除操作。

演示删除操作

```
DELETE FROM Worker WHERE Number = '00001';
```

有错误信息

```
ERROR:  CEO cannot be deleted
背景:  PL/pgSQL function forbid_delete_ceo() line 4 at RAISE
```

可知触发器 T2 成功生效

7) 演示违反 T3 触发器的约束的更新操作。

演示更新操作

```
UPDATE Worker SET Number = '00003' WHERE Number = '00002';
```

有错误信息

```
ERROR:  Number cannot be changed
背景:  PL/pgSQL function forbid_update_number() line 4 at RAISE
```

可知触发器 T3 成功生效

8) 演示 INSTEAD OF 触发器在不可更新视图上的运用。

创建视图 WorkerView

```
CREATE VIEW WorkerView AS SELECT * FROM Worker WHERE Department = 'IT';

CREATE OR REPLACE FUNCTION instead_of_trigger_function() RETURNS TRIGGER AS $$
BEGIN
    RAISE EXCEPTION 'WorkerView is not updatable';
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER instead_of_trigger
INSTEAD OF INSERT OR UPDATE OR DELETE ON WorkerView
FOR EACH ROW
EXECUTE FUNCTION instead_of_trigger_function();
```

```
school=# \d workerview
               视图 "public.workerview"
    栏位    |      类型      | 校对规则 | 可空的 | 预设
------------+---------------+----------+--------+------
 number     | character(5)  |          |        |
 name       | character(8)  |          |        |
 sex        | character(1)  |          |        |
 sage       | integer       |          |        |
 department | character(20) |          |        |
触发器:
    instead_of_trigger INSTEAD OF INSERT OR DELETE OR UPDATE ON workerview FOR EACH ROW EXECUTE FUNCTION instead_of_trigger_function()
```

进行插入操作

```
INSERT INTO WorkerView (Number, Name, Sex, Sage, Department) VALUES ('00003',
'Carol', 'F', 15, 'IT');
```

有错误信息

```
ERROR:  WorkerView is not updatable
背景:  PL/pgSQL function instead_of_trigger_function() line 3 at RAISE
```

UPDATE/DELETE 操作同理

自我实践

1) 建立一个在 Worker 表上的触发器 T4，要求插入记录的 sage 值必须比表中已记录的最大 sage 值大。

```
CREATE OR REPLACE FUNCTION check_max_sage() RETURNS TRIGGER AS $$
DECLARE
    max_sage INT;
BEGIN
    SELECT MAX(Sage) INTO max_sage FROM Worker;
    IF NEW.Sage <= max_sage THEN
        RAISE EXCEPTION 'Sage must be greater than the maximum sage value in the
table';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_max_sage_trigger
BEFORE INSERT ON Worker
FOR EACH ROW
EXECUTE FUNCTION check_max_sage();
```

2) 建立一个在 Worker 表上的触发器 T5，要求当更新一个记录的时候，表中记录的 sage 值要比老记录的 sage 值大，因为一般工资级别只能升不能降。

```
CREATE OR REPLACE FUNCTION check_sage_increase() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Sage < OLD.Sage THEN
        RAISE EXCEPTION 'Sage cannot be decreased';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_sage_increase_trigger
BEFORE UPDATE ON Worker
FOR EACH ROW
EXECUTE FUNCTION check_sage_increase();
```

4. 实验心得

本次实验主要是通过创建和使用触发器来加深对数据完整性的理解。在实验中，我按照题目要求创建了多个触发器，并进行了相应的演示和测试。总的来说，本次实验通过创建和使用触发器，加深了我对数据完整性和约束的理解。触发器是数据库中强大的工具，可以在数据操作过程中自动执行特定的逻辑，保证数据的一致性和有效性。掌握触发器的使用方法对于数据库开发和管理非常重要。