触发器

触发器

触发器会在指定的数据库事件发生时自动执行函数。

注意事项:

1.仅支持普通行存表

2.如果同一个事件定义多个相同类型触发器 这按照触发器名称字母顺序触发

触发器函数

```
CREATE OR REPLACE FUNCTION 触发器名称()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
        <标准SQL或过程化SQL>
RETURN <NEW或OLD>;
END;
$$ LANGUAGE PLPGSQL;
```

语法格式

- 创建触发器

```
CREATE TRIGGER trigger_name { BEFORE | AFTER | INSTEAD OF } { event [ O
}
ON table_name
[ FOR [ EACH ] { ROW | STATEMENT } ]    [ WHEN ( condition ) ]
EXECUTE PROCEDURE function_name ( arguments );
```

- 修改触发器

```
ALTER TRIGGER trigger_name ON table_name RENAME TO new_trigger_name;
```

- 删除触发器

```
DROP TRIGGER trigger_name ON table_name [ CASCADE | RESTRICT ];
```

参数说明

- **trigger_name**

触发器名称。

- **BEFORE**

触发器函数是在触发事件发生前执行。

- **AFTER**

触发器函数是在触发事件发生后执行。

- **INSTEAD OF**

触发器函数直接替代触发事件。

- **event**

启动触发器的事件，取值范围包括：INSERT、UPDATE、DELETE或TRUNCATE，也可以通过OR同时指定多个触发事件。

- **table_name**

触发器对应的表名称。

- **FOR EACH ROW | FOR EACH STATEMENT**

触发器的触发频率。

- FOR EACH ROW是指该触发器是受触发事件影响的每一行触发一次。
- FOR EACH STATEMENT是指该触发器是每个SQL语句只触发一次。

未指定时默认值为FOR EACH STATEMENT。约束触发器只能指定为FOR EACH ROW。

- **function_name**

用户定义的函数，必须声明为不带参数并返回类型为触发器，在触发器触发时执行。

- **arguments**

执行触发器时要提供给函数的可选的以逗号分隔的参数列表。

- **new_trigger_name**

修改后的新触发器名称。

1. --创建源表及触发表

```
CREATE TABLE test_trigger_src_tbl(id1 INT, id2 INT, id3 INT);
CREATE TABLE test_trigger_des_tbl(id1 INT, id2 INT, id3 INT);
```

2. --创建触发器函数

```
CREATE OR REPLACE FUNCTION tri_insert_func() RETURNS TRIGGER AS
$$
DECLARE
BEGIN
iNSERT INTO test_trigger_des_tbl VALUES(NEW.id1, NEW.id2, NEW.id3);
RETURN NEW;
END
$$ LANGUAGE PLPGSQL;
```

```
openGauss=# CREATE OR REPLACE FUNCTION tri_update_func() RETURNS TRIGGER AS
$$
DECLARE
BEGIN
UPDATE test_trigger_des_tbl SET id3 = NEW.id3 WHERE id1=OLD.id1;
RETURN OLD;
END
$$ LANGUAGE PLPGSQL;
```

```
openGauss=# CREATE OR REPLACE FUNCTION TRI_DELETE_FUNC() RETURNS TRIGGER AS
$$
```

```
DECLARE
BEGIN
DELETE FROM test_trigger_des_tbl WHERE id1=OLD.id1;
RETURN OLD;
end
$$ LANGUAGE PLPGSQL;
```

创建INSERT触发器

```
openGauss=# CREATE TRIGGER insert_trigger
BEFORE INSERT ON test_trigger_src_tbl
FOR EACH ROW
EXECUTE PROCEDURE tri_insert_func();
```

创建UPDATE触发器

```
openGauss=# CREATE TRIGGER update_trigger
AFTER UPDATE ON test_trigger_src_tbl
FOR EACH ROW
EXECUTE PROCEDURE tri_update_func();
```

--创建DELETE触发器

```
openGauss=# CREATE TRIGGER delete_trigger
BEFORE DELETE ON test_trigger_src_tbl
FOR EACH ROW
EXECUTE PROCEDURE tri_delete_func();
```

--执行INSERT触发事件并检查触发结果

```
INSERT INTO test_trigger_src_tbl VALUES(100,200,300);
SELECT * FROM test_trigger_src_tbl;
SELECT * FROM test_trigger_des_tbl; //查看触发操作是否生效。
```

3. --执行UPDATE触发事件并检查触发结果

```
UPDATE test_trigger_src_tbl SET id3=400 WHERE id1=100;
SELECT * FROM test_trigger_src_tbl;
SELECT * FROM test_trigger_des_tbl; //查看触发操作是否生效
```

--执行DELETE触发事件并检查触发结果

```
DELETE FROM test_trigger_src_tbl WHERE id1=100;
SELECT * FROM test_trigger_src_tbl;
SELECT * FROM test_trigger_des_tbl; //查看触发操作是否生效
```

4. --修改触发器

```
ALTER TRIGGER delete_trigger ON test_trigger_src_tbl RENAME TO delete_trigger_renamed;
```

5. --删除触发器

```
DROP TRIGGER insert_trigger ON test_trigger_src_tbl;
```

```
DROP TRIGGER update_trigger ON test_trigger_src_tbl;
```

```
DROP TRIGGER delete_trigger_renamed ON test_trigger_src_tbl;
```