# 实验4 触发器

[实验4 触发器 1](#_Toc88748789)

[4.1创建触发器 1](#_Toc88748790)

[语法格式 1](#_Toc88748791)

[参数说明 2](#_Toc88748792)

[示例 4](#_Toc88748793)

[4.2修改触发器 5](#_Toc88748794)

[语法格式 5](#_Toc88748795)

[参数说明 5](#_Toc88748796)

[示例 5](#_Toc88748797)

[4.3删除触发器 5](#_Toc88748798)

[语法格式 6](#_Toc88748799)

[参数说明 6](#_Toc88748800)

[示例 6](#_Toc88748801)

# 实验4 触发器

（1）实验目的

掌握数据库触发器的设计和使用方法

（2）实验内容和要求

在openGauss数据库上定义BEFORE触发器和AFTER触发器。能够理解不同类型触发器的作用和执行原理，验证触发器的有效性。

## 4.1创建触发器

**功能描述**

创建一个触发器。 触发器将与指定的表或视图关联，并在特定条件下执行指定的函数。

**注意事项**

* 当前仅支持在普通行存表上创建触发器，不支持在列存表、临时表、unlogged表等类型表上创建触发器。
* 如果为同一事件定义了多个相同类型的触发器，则按触发器的名称字母顺序触发它们。
* 触发器常用于多表间数据关联同步场景，对SQL执行性能影响较大，不建议在大数据量同步及对性能要求高的场景中使用。

### 语法格式

CREATE [ CONSTRAINT ] TRIGGER trigger\_name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }

ON table\_name

[ FROM referenced\_table\_name ]

{ NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } }

[ FOR [ EACH ] { ROW | STATEMENT } ]

[ WHEN ( condition ) ]

EXECUTE PROCEDURE function\_name ( arguments );

**其中event包含以下几种：**

INSERT

UPDATE [ OF column\_name [, ... ] ]

DELETE

TRUNCATE

### 参数说明

* **CONSTRAINT**

可选项，指定此参数将创建约束触发器，即触发器作为约束来使用。除了可以使用SET CONSTRAINTS调整触发器触发的时间之外，这与常规触发器相同。 约束触发器必须是AFTER ROW触发器。

* **trigger\_name**

触发器名称，该名称不能限定模式，因为触发器自动继承其所在表的模式，且同一个表的触发器不能重名。 对于约束触发器，使用SET CONSTRAINTS修改触发器行为时也使用此名称。

取值范围：符合标识符命名规范的字符串，且最大长度不超过63个字符。

* **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，约束触发器只能指定为AFTER。

* **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。

对于UPDATE事件类型，可以使用下面语法指定列：

UPDATE OF column\_name1 [, column\_name2 ... ]

表示当这些列作为UPDATE语句的目标列时，才会启动触发器，但是INSTEAD OF UPDATE类型不支持指定列信息。

* **table\_name**

需要创建触发器的表名称。

取值范围：数据库中已经存在的表名称。

* **referenced\_table\_name**

约束引用的另一个表的名称。 只能为约束触发器指定，常见于外键约束。

取值范围：数据库中已经存在的表名称。

* **DEFERRABLE | NOT DEFERRABLE**

约束触发器的启动时机，仅作用于约束触发器。这两个关键字设置该约束是否可推迟。

* **INITIALLY IMMEDIATE** **| INITIALLY DEFERRED**

如果约束是可推迟的，则这个子句声明检查约束的缺省时间，仅作用于约束触发器。

* **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。

* **condition**

决定是否实际执行触发器函数的条件表达式。当指定WHEN时，只有在条件返回true时才会调用该函数。

在FOR EACH ROW触发器中，WHEN条件可以通过分别写入OLD.column\_name或NEW.column\_name来引用旧行或新行值的列。 当然，INSERT触发器不能引用OLD和DELETE触发器不能引用NEW。

INSTEAD OF触发器不支持WHEN条件。

WHEN表达式不能包含子查询。

对于约束触发器，WHEN条件的评估不会延迟，而是在执行更新操作后立即发生。 如果条件返回值不为true，则触发器不会排队等待延迟执行。

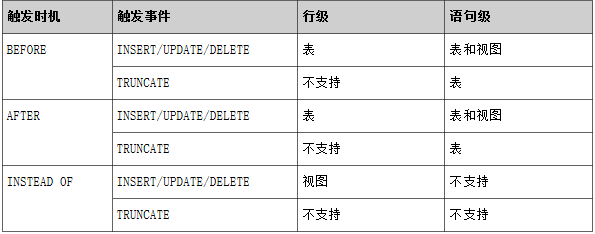
* **function\_name**

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。参数是文字字符串常量，简单的名称和数字常量也可以写在这里，但它们都将被转换为字符串。 请检查触发器函数的实现语言的描述，以了解如何在函数内访问这些参数。

**表1** 表和视图上支持的触发器种类：



**表2** PLPGSQL类型触发器函数特殊变量：



### 示例

--创建源表及触发表

openGauss=# CREATE TABLE test\_trigger\_src\_tbl(id1 INT, id2 INT, id3 INT);

openGauss=# CREATE TABLE test\_trigger\_des\_tbl(id1 INT, id2 INT, id3 INT);

--创建触发器函数

postgres=# 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;

postgres=# 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;

postgres=# 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

openGauss=# INSERT INTO test\_trigger\_src\_tbl VALUES(100,200,300);

openGauss=# SELECT \* FROM test\_trigger\_src\_tbl;

openGauss=# SELECT \* FROM test\_trigger\_des\_tbl; //查看触发操作是否生效。

--执行UPDATE触发事件并检查触发结果

openGauss=# UPDATE test\_trigger\_src\_tbl SET id3=400 WHERE id1=100;

openGauss=# SELECT \* FROM test\_trigger\_src\_tbl;

openGauss=# 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

openGauss=# DELETE FROM test\_trigger\_src\_tbl WHERE id1=100;

openGauss=# SELECT \* FROM test\_trigger\_src\_tbl;

openGauss=# SELECT \* FROM test\_trigger\_des\_tbl; //查看触发操作是否生效

## 4.2修改触发器

**功能描述**

修改触发器定义。

**注意事项**

只有触发器所在表的所有者可以执行ALTER TRIGGER操作，系统管理员默认拥有此权限。

### 语法格式

ALTER TRIGGER trigger\_name ON table\_name RENAME TO new\_name;

### 参数说明

**trigger\_name**

要修改的触发器名称。

取值范围：已存在的触发器。

**table\_name**

要修改的触发器所在的表名称。

取值范围：已存在的含触发器的表。

**new\_name**

修改后的新名称。

取值范围：符合标识符命名规范的字符串，最大长度不超过63个字符，且不能与所在表上其他触发器同名。

### 示例

--修改触发器

openGauss=# ALTER TRIGGER delete\_trigger ON test\_trigger\_src\_tbl RENAME TO delete\_trigger\_renamed;

--禁用insert\_trigger触发器

openGauss=# ALTER TABLE test\_trigger\_src\_tbl DISABLE TRIGGER insert\_trigger;

--禁用当前表上所有触发器

openGauss=# ALTER TABLE test\_trigger\_src\_tbl DISABLE TRIGGER ALL;

## 4.3删除触发器

**功能描述**

删除触发器。

**注意事项**

只有触发器的所有者可以执行DROP TRIGGER操作，系统管理员默认拥有此权限。

### 语法格式

DROP TRIGGER [ IF EXISTS ] trigger\_name ON table\_name [ CASCADE | RESTRICT ];

### 参数说明

* IF EXISTS

使用IF EXIST时，如果要删除的触发器不存在，删除操作能够执行成功，不会报错。

* schema\_name

要删除的触发器所有者。

* trigger\_name

要删除的触发器的名称。

### 示例

--删除触发器

openGauss=# DROP TRIGGER insert\_trigger ON test\_trigger\_src\_tbl;

openGauss=# DROP TRIGGER update\_trigger ON test\_trigger\_src\_tbl;

openGauss=# DROP TRIGGER delete\_trigger\_renamed ON test\_trigger\_src\_tbl;