1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название переменной | Смысл | В каких триггерах используются | |
| NEW | Тип данных RECORD. Переменная содержит новую строку базы данных для команд INSERT/UPDATE в триггерах уровня строки. В триггерах уровня оператора и для команды DELETE этой переменной значение не присваивается. | FOR EACH ROW |  |
| OLD | Тип данных RECORD. Переменная содержит старую строку базы данных для команд UPDATE/DELETE в триггерах уровня строки. В триггерах уровня оператора и для команды INSERT этой переменной значение не присваивается. | FOR EACH ROW |  |
| TG\_NAME | Тип данных name. Переменная содержит имя сработавшего триггера. | FOR EACH ROW / STATEMENT |  |
| TG\_WHEN | Тип данных text. Строка, содержащая BEFORE, AFTER или INSTEAD OF, в зависимости от определения триггера. | FOR EACH ROW / STATEMENT |  |
| TG\_LEVEL | Тип данных text. Строка, содержащая ROW или STATEMENT, в зависимости от определения триггера. | FOR EACH ROW / STATEMENT |  |
| TG\_OP | Тип данных text. Строка, содержащая INSERT, UPDATE, DELETE или TRUNCATE, в зависимости от того, для какой операции сработал триггер. | FOR EACH ROW / STATEMENT |  |
| TG\_RELID | Тип данных oid. OID таблицы, для которой сработал триггер. | FOR EACH ROW / STATEMENT |  |
| TG\_RELNAME | Тип данных name. Имя таблицы, для которой сработал триггер. Эта переменная устарела и может стать недоступной в будущих релизах. Вместо неё нужно использовать TG\_TABLE\_NAME. | FOR EACH ROW / STATEMENT |  |
| TG\_TABLE\_NAME | Тип данных name. Имя таблицы, для которой сработал триггер. | FOR EACH ROW / STATEMENT |  |
| TG\_TABLE\_SCHEMA | Тип данных name. Имя схемы, содержащей таблицу, для которой сработал триггер. | FOR EACH ROW / STATEMENT |  |
| TG\_NARGS | Тип данных integer. Число аргументов в команде CREATE TRIGGER, которые передаются в триггерную процедуру. | FOR EACH ROW / STATEMENT |  |
| TG\_ARGV[] | Тип данных массив text. Аргументы от оператора CREATE TRIGGER. Индекс массива начинается с 0. Для недопустимых значений индекса ( < 0 или >= tg\_nargs) возвращается NULL. | FOR EACH ROW / STATEMENT |  |

1.2 create table t (

id integer primary key,

s text);

insert into t values

(1,'Один'),

(2,'Два');

select \* from t;

CREATE OR REPLACE FUNCTION describe()

RETURNS trigger AS $$

DECLARE

rec record;

str text := '';

BEGIN

IF TG\_LEVEL = 'ROW' THEN

CASE TG\_OP

WHEN 'DELETE' THEN rec := OLD; str := OLD::text;

WHEN 'UPDATE' THEN rec := NEW; str := OLD || ' -> ' || NEW;

WHEN 'INSERT' THEN rec := NEW; str := NEW::text;

END CASE;

END IF;

RAISE NOTICE '% % % %: %', TG\_TABLE\_NAME, TG\_WHEN, TG\_OP, TG\_LEVEL, str;

RETURN rec;

END $$ LANGUAGE plpgsql;

CREATE TRIGGER t\_before\_stmt

BEFORE INSERT OR UPDATE OR DELETE ON t

FOR EACH STATEMENT EXECUTE PROCEDURE describe();

CREATE TRIGGER t\_after\_stmt

AFTER INSERT OR UPDATE OR DELETE ON t

FOR EACH STATEMENT EXECUTE PROCEDURE describe();

CREATE TRIGGER t\_before\_row

BEFORE INSERT OR UPDATE OR DELETE ON t

FOR EACH ROW EXECUTE PROCEDURE describe();

CREATE TRIGGER t\_after\_row

AFTER INSERT OR UPDATE OR DELETE ON t

FOR EACH ROW EXECUTE PROCEDURE describe();

INSERT INTO t VALUES (3,'aaa');

UPDATE t SET s = 'bbb';

UPDATE t SET s = 'aaa' where id = 1;

INSERT INTO t VALUES (2,'ccc'),(3,'ddd') ON CONFLICT(id) DO UPDATE SET s = EXCLUDED.s;

DELETE FROM t;

3.1 CREATE TRIGGER t\_before\_stmt

before insert or update or delete ON t

for each statement

EXECUTE FUNCTION describe();

CREATE TRIGGER t\_after\_stmt

after insert or update or delete ON t

for each statement

EXECUTE FUNCTION describe();

3.2

3.1 create or replace trigger before\_upd

before update on categories

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_ins

before insert on categories

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_upd

before update on products

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_ins

before insert on products

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_upd

before update on employees

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_ins

before insert on employees

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_upd

before update on customers

for each row

execute function log\_t\_upd\_ins();

create or replace trigger before\_ins

before insert on customers

for each row

execute function log\_t\_upd\_ins();

select \* from categories

insert into categories values (3,'Платья')

update categories set name = 'Кепки' where id = 2

select \* from products

insert into products values (3,'Наушники')

update products set name = 'Майки' where id = 2

select \* from products

insert into products values (3,'Наушники')

update products set name = 'Майки' where id = 2

select \* from t

delete from t where id > 2

insert into t values (3,'Три')

update t set s = 'Один' where id = 3

create or replace function log\_t\_upd\_ins()

returns trigger as $$

begin

if TG\_OP = 'INSERT' then

new.ins\_user:=current\_user;

new.ins\_time:=current\_timestamp;

return new;

end if;

if TG\_OP = 'UPDATE' then

new.upd\_user:=current\_user;

new.upd\_time:=current\_timestamp;

return new;

end if;

end;

$$ language plpgsql;

create or replace trigger tr\_before\_upd\_ins\_t

before update or insert on t

for each row

execute function log\_t\_upd\_ins();

CREATE TABLE categories (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

CREATE TABLE products (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

category\_id INT,

price DECIMAL(10, 2)

);

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

position VARCHAR(100),

salary DECIMAL(10, 2)

);

CREATE TABLE customers (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100),

phone VARCHAR(20)

);

INSERT INTO categories VALUES

(1,'Электроника'),

(2,'Одежда');

INSERT INTO products VALUES

(1, 'Смартфон', 5, 25000.00),

(2, 'Футболка', 4, 1500.00);

INSERT INTO employees VALUES

(1, 'Иванов Иван', 'Менеджер', 30000),

(2, 'Петров Петр', 'Продавец', 25000);

INSERT INTO customers VALUES

(1, 'Анна Смирнова', 'anna@example.com', 8546128125),

(2, 'Дмитрий Иванов', 'dima@example.com', 851293151);

ALTER TABLE categories

ADD ins\_time timestamp,

ADD ins\_user VARCHAR(50),

ADD upd\_time timestamp,

ADD upd\_user VARCHAR(50);

ALTER TABLE products

ADD ins\_time timestamp,

ADD ins\_user VARCHAR(50),

ADD upd\_time timestamp,

ADD upd\_user VARCHAR(50);

ALTER TABLE employees

ADD ins\_time timestamp,

ADD ins\_user VARCHAR(50),

ADD upd\_time timestamp,

ADD upd\_user VARCHAR(50);

ALTER TABLE customers

ADD ins\_time timestamp,

ADD ins\_user VARCHAR(50),

ADD upd\_time timestamp,

ADD upd\_user VARCHAR(50);