**ЛАБОРАТОРНАЯ РАБОТА № 5. ТРИГГЕРЫ**

Триггер − это хранимая процедура особого типа, активизируемая при каком-либо событии, связанном с изменением данных конкретной таблицы. Триггеры ассоциируются с такими операциями, как вставка (INSERT), обновление (UPDATE) и удаление (DELETE) данных.

В PostgreSQL триггеры могут вызывать функции, написанные на любом языке, но за одним исключением: функция не может быть полностью реа­лизована на SQL

Триггер создается на основе существующей функции. В определении триггера указывается, должна ли заданная функция вызывать­ся до или после выполнения некоторой операции с таблицей. Синтаксис опреде­ления триггера выглядит так:

CREATE TRIGGER триггер {BEFORE | AFTER} {событие [OR событие ... ]} ON таблица

FOR EACH { ROW | STATEMENT }

EXECUTE PROCEDURE функция (аргументы)

В качестве примера создадим триггер.

Приведем еще один пример использования триггеров при ведении журналов изменений. Журнал изменений в БД представляет собой таблицу БД, в которой фиксируются действия над всей БД или отдельными ее таблицами. В многопользовательских системах ведение такого журнала позволяет определить источник недостоверных или искаженных данных.

Определим в БД таблицу ZAKAZY\_JOURNAL: в которую будем автоматически записывать любые изменения, добавления, удаления в таблице zakazy. При этом будем фиксировать дату, операцию (INS, UPD, DEL) над таблицей zakazy, а также новое и старое значения столбцов (old\_values, new\_values). Для операции удаления новое значение столбцов new\_values будет пустым. Для операции добавления пустым будет старое значение столбцов old\_values.

К примеру: пусть в таблицу klient внесены некоторые изменения. Тогда, выполнив оператор

SELECT \*

FROM zakazy.klient\_JOURNAL;

получим историю изменений таблицы klient:

CREATE OR REPLACE FUNCTION zakazy.klient\_action()

RETURNS trigger AS

$BODY$

DECLARE

op TEXT;

new\_v TEXT;

old\_v TEXT;

BEGIN

IF(TG\_OP = 'INSERT') THEN

OP = 'INS';

new\_v = NEW.id\_klient ||', ' || NEW.fio\_klient ||', ' ||NEW.name\_klient ||

', ' || NEW.adress || ', ' || NEW.nomer;

old\_v = '';

INSERT INTO zakazy.klient\_journal (dat\_izm, deistvie, old\_values, new\_values)

VALUES (NOW(), op, old\_v, new\_v);

RETURN NEW;

ELSEIF (TG\_OP = 'UPDATE') THEN

OP = 'UPD';

new\_v = NEW.id\_klient ||', ' || NEW.fio\_klient ||', ' ||NEW.name\_klient ||

', ' || NEW.adress || ', ' || NEW.nomer;

old\_v = OLD.id\_klient ||', ' || OLD.fio\_klient ||', ' ||OLD.name\_klient ||

', ' || OLD.adress || ', ' || OLD.nomer;

INSERT INTO zakazy.klient\_journal (dat\_izm, deistvie, old\_values, new\_values)

VALUES (NOW(), op, old\_v, new\_v);

RETURN NEW;

ELSE

OP = 'DEL';

new\_v = '';

old\_v = OLD.id\_klient ||', ' || OLD.fio\_klient ||', ' ||OLD.name\_klient ||

', ' || OLD.adress || ', ' || OLD.nomer;

INSERT INTO zakazy.klient\_journal (dat\_izm, deistvie, old\_values, new\_values)

VALUES (NOW(),op,old\_v,new\_v);

RETURN NEW;

END IF;

RETURN NULL;

END;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION zakazy.klient\_action()

OWNER TO postgres;