МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет» (ФГБОУ ВО «ВятГУ») Институт математики и информационных систем Факультет автоматики и вычислительной техники Кафедра электронных вычислительных машин

Отчёт по лабораторной работа № 3 по дисциплине "Базы данных"

Выполнил студент группы ИВТб-3301-04-00 /Бикметов И.Р.

Проверил доцент кафедры ЭВМ /Клюкин В. Л.

Киров 2025

1. Задание

При выполнении работы нужно использовать БД, созданную в лабораторных работах №1 и №2. Нужно выполнить следующие шаги.

1) Для любой таблицы создать функцию save\_имя\_таблицы, которая принимает на вход параметры, соответствующие её столбцам, и, если переданное значение первичного ключа равно null, выполняет запрос insert, иначе – запрос update для соответствующей строки. Функция должна вернуть значение первичного ключа вставленной или изменённой строки.

2) Для любой таблицы, на которую имеются внешние ключи, создать функцию delete\_имя\_таблицы, принимающую на вход значение первичного ключа строки и ничего не возвращающую.

3) Для таблицы, содержащей столбец с числовыми значениям, создать функцию, которая принимает на вход число – минимальное значение – и возвращает setof имя\_таблицы – множество строк, в которых значение числа больше или равно переданному аргументу.

4) Создать составной тип, содержащий не менее 2-3 полей, по крайней мере одно из которых должно быть числовым. Создать функцию, которая принимает массив объектов этого типа и минимальное значение для указанного поля. Функция должна возвращать массив отфильтрованных по переданному значению объектов.

5) Для любой таблицы создать таблицу log\_имя\_таблицы, которая будет содержать лог изменений по любому выбранному столбцу этой таблицы. Для этого нужны столбцы:

* первичный ключ;
* внешний ключ на выбранную таблицу;
* дата изменения строки;
* старое значение столбца;
* новое значение столбца.

Реализовать заполнение таблицы с логом с помощью триггеров на вставку/изменение строк.

6) Реализовать любую функцию на свой выбор, использующую для получения результата динамически формируемый запрос.

Все функции должны быть реализованы на PL/pgSQL. Отчет по лабораторной работе должен содержать код создания перечисленных функций, составного типа из шага 4 и таблицы из шага 5, а также демонстрацию работы созданных функций.

1. Ход работы

Предметная область связана с обработкой заявок от клиентов в CRM системе ЖКХ, на каждую заявку может быть добавлено несколько услуг при этом в каждой услуге может быть несколько исполнителей.

Сначала было принято решение реализовать функцию save\_user, которая принимает на входе имя и пароль пользователя. В случае, если данный показатель, уже есть в таблице, то его значение меняется на новое. Если же его не было, то добавляется новое значение. В случае NULL, ничего не происходит, таблица просто обновляет старое значение на себя же.

На следующем шаге реализовалась функция del\_user, которая удаляет строку по заданному id. Если введенного id нет в таблице, то ничего не произойдет, будет выведено сообщение о невозможности удаления данной строки из таблицы. В случае же, если введенный идентификатор присутствует в таблице, то произойдет соответствующее удаление.

Следующая функция – фильтрация таблицы по заданному ключу. Реализация функции будет производиться для таблицы услуг, requests\_services\_to\_position. Фильтрация будет производиться по полю price.

Функция фильтрации массива объектов будет принимать на вход массив данных и числовое значение, по которому будет фильтроваться массив данных и возвращаться из функции. Было принято решение реализовать данную функцию для таблицы requests\_client\_applications.

Функция логирования действий для таблицы users будет не простой, а триггерной. Триггерная функция будет определять какой вид запроса был произведён и будет в соответствии с ним корректировать запрос. Триггер будет срабатывать после внесения изменений в таблицу users.

Функция для создания динамически формируемых запросов. Созданная функция принимает на вход название таблицы, название столбца и id поля, которое будет выведено. На выходе будет возвращаться текстовая строка с содержимым поля.

Все результаты и реализация приведены в приложениях А и Б.

1. Листинг программы

Листинг программы приведен в приложении А.

1. Экранные формы

Экранные формы приведены в приложении Б.

1. Вывод

В ходе выполнения лабораторной работы были изучены основы написания пользовательских функций на процедурном языке PL/pgSQL. Освоена работа с составными типами данных и массивами. Познакомились с созданием триггеров.

Приложение А

(обязательное)

Листинг программы

* Обновление данных в строке и добавление новых

CREATE OR REPLACE FUNCTION save\_user(\_id INTEGER, \_name TEXT, \_password TEXT)

RETURNS INTEGER AS $$

DECLARE

user\_id INTEGER;

BEGIN

IF \_id IS NULL THEN

INSERT INTO users(name, password)

VALUES(\_name, \_password)

RETURNING id INTO user\_id;

RETURN user\_id;

ELSE

UPDATE users SET

name = \_name,

password = \_password

WHERE id = \_id;

RETURN \_id;

END IF;

END;

$$ LANGUAGE plpgsql;

* Удаление строки по id

CREATE OR REPLACE FUNCTION del\_user(\_id INTEGER)

RETURNS VOID AS $$

DECLARE

BEGIN

DELETE FROM users WHERE id = \_id;

EXCEPTION

WHEN foreign\_key\_violation THEN

RAISE EXCEPTION 'Невозможно выполнить удаление, так как есть внешние ссылки.';

END;

$$ LANGUAGE plpgsql;

* Функция фильтрации по числовому значению

CREATE OR REPLACE FUNCTION sort\_by\_min(\_minPrice INTEGER)

RETURNS SETOF requests\_services\_to\_position AS $$

DECLARE

BEGIN

RETURN QUERY (SELECT \* FROM requests\_services\_to\_position WHERE price >= \_minPrice);

END;

$$ LANGUAGE plpgsql;

* Функция фильтрации массива объектов

CREATE TYPE t\_request AS (

id INTEGER,

resident\_name TEXT,

comments TEXT,

client\_id INTEGER,

object\_type\_id INTEGER

);

CREATE OR REPLACE FUNCTION filter\_array\_of\_requests (

arr t\_request[],

filter\_var INTEGER

)

RETURNS t\_request[]

AS $$

BEGIN

RETURN ARRAY( /\* Преобразуем выборку в массив \*/

SELECT (id, resident\_name, comments, client\_id, object\_type\_id) /\* Создаем таблицу из элементов массива \*/

FROM unnest(arr)

WHERE object\_type\_id >= filter\_var

);

END;

$$ LANGUAGE plpgsql;

* Функция для создания таблицы с логом изменений

CREATE TABLE log\_users (

id BIGSERIAL PRIMARY KEY,

user\_id BIGINT REFERENCES users(id),

change\_datetime TIMESTAMP DEFAULT NOW(),

old\_value TEXT DEFAULT NULL,

new\_value TEXT DEFAULT NULL

);

CREATE OR REPLACE FUNCTION trigger\_func()

RETURNS TRIGGER AS $$

DECLARE

old\_val TEXT;

BEGIN

/\* Определяем "старое" значение \*/

IF (TG\_OP = 'UPDATE') THEN

old\_val := OLD.name;

ELSIF (TG\_OP = 'INSERT') THEN

old\_val := NULL;

end if;

/\* Производим запрос \*/

INSERT INTO log\_users

(user\_id, old\_value, new\_value)

VALUES

(NEW.id, old\_val, NEW.name);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER commit\_users\_change

AFTER UPDATE OR INSERT

ON users

FOR EACH ROW

EXECUTE PROCEDURE trigger\_func();

* Функция для создания динамически формируемого запроса

CREATE OR REPLACE FUNCTION get\_value\_by\_id (

tableName VARCHAR,

columnName VARCHAR,

id BIGINT

)

RETURNS TEXT AS $$

DECLARE

res TEXT;

BEGIN

EXECUTE 'SELECT '|| columnName ||' FROM '|| tableName ||' WHERE id = $1' USING id INTO res;

RETURN res;

END;

$$ LANGUAGE plpgsql;

Приложение Б

(обязательное)

Экранные формы

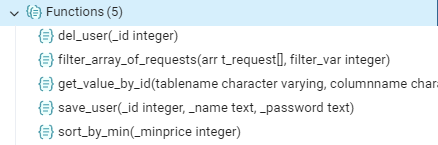


Рисунок 1 – Отображение функций в среде разработки

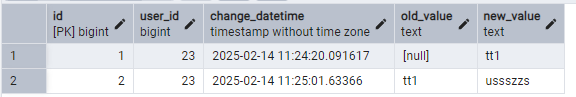


Рисунок 2- Таблица с логом изменений в таблице users

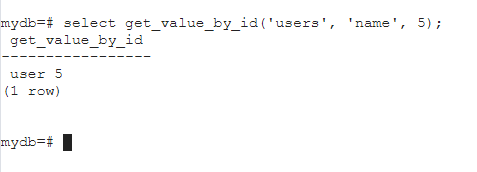


Рисунок 3 – Таблица динамически сформированного запроса