

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Отчет по лабораторной работе №3
по дисциплине
«Управление данными»

Выполнил студент гр. ИВТб-2301-05-00	_____ /Макаров С.А./
Преподаватель	_____ /Клюкин В.Л./

Киров 2025

Цель

Цель лабораторной работы: познакомиться с созданием пользовательских функций и триггеров в PostgreSQL, освоить работу с составными типами данных и массивами, изучить основы работы с процедурным языком PL/pgSQL.

Задание

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. Если на удаляемую строку существуют ссылки, то функция должна поднимать ошибку «Невозможно выполнить удаление, так как есть внешние ссылки».
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.

Решение

Задание 1

Если в параметре ID дисциплины указан NULL, то функция должна создать новую запись в таблице, если указан – обновить существующую запись. В случае если мы обновим запись с ID, не существующем на данный момент в таблице, ничего не произойдет.

Функция будет возвращать значение id строки, с которой была произведена работа.

Далее представлен код разработанной функции:

```
CREATE OR REPLACE FUNCTION save_category (  
    _id BIGINT,  
    _title VARCHAR(256)  
)  
RETURNS BIGINT  
AS $$  
DECLARE  
    category_id BIGINT;  
  
BEGIN  
    IF _id IS NULL THEN  
        INSERT INTO categories (title)  
        VALUES (_title)  
        RETURNING id INTO category_id;  
    ELSE  
        UPDATE categories SET  
            title = _title  
        WHERE id = _id;  
  
        category_id := _id;  
    END IF;  
  
    RETURN category_id;  
END;  
$$ LANGUAGE plpgsql;
```

Проверим работу функции. Для начала, посмотрим, как выглядит таблица до изменения (Рисунок 1):

	id [PK] bigint	title character varying (256)	created_at timestamp with time zone	updated_at timestamp with time zone
1	1	Пиццы	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
2	2	Закуски	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
3	3	Завтрак	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
4	4	Десерты	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
5	5	Коктейли	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
6	6	Кофе	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...

Рисунок 1 – Таблица categories до изменений

Теперь выполним запрос:

```
SELECT save_category(null, 'Обед');
```

На рисунке 2 представлен вывод Id новой строки:

	save_category bigint
1	7

Рисунок 2 – Id новой строки

Затем выполним запрос:

```
SELECT save_category(4, 'Сладости');
```

На рисунке 3 представлен вывод Id измененной строки:

	save_category bigint
1	4

Рисунок 3 – Id измененной строки

Проверим таблицу (Рисунок 4). Создана строка с индексом 7, а содержимое строки с индексом 4 изменено.

	id [PK] bigint	title character varying (256)	created_at timestamp with time zone	updated_at timestamp with time zone
1	1	Пиццы	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
2	2	Закуски	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
3	3	Завтрак	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
4	4	Сладости	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
5	5	Коктейли	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
6	6	Кофе	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
7	7	Обед	2025-09-20 11:51:55.941817+...	2025-09-20 11:51:55.941817+...

Рисунок 4 – Таблица categories после изменений

Задание 2

Функция delete category будет на вход принимать ID строки, которую нужно будет удалить из таблицы.

Если такого ID нет в таблице, то ничего не произойдет, скрипт отработает без ошибок.

Если в других таблицах строки ссылаются на удаляемую нами строку, мы должны выдать ошибку с текстом: «Невозможно выполнить удаление, так как есть внешние ссылки.».

Далее представлен код разработанной функции:

```
CREATE OR REPLACE FUNCTION delete_category (
    _id BIGINT
)
RETURNS VOID
AS $$
BEGIN
    DELETE FROM categories
    WHERE id = _id;

EXCEPTION
    WHEN foreign_key_violation THEN
        RAISE EXCEPTION 'Невозможно выполнить удаление,
            так как есть внешние ссылки.';
END;
$$ LANGUAGE plpgsql;
```

Далее нужно подготовить таблицы к тестам. Таблица до изменений представлена на рисунке 5:

	id [PK] bigint	title character varying (256)	created_at timestamp with time zone	updated_at timestamp with time zone
1	1	Пиццы	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
2	2	Закуски	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
3	3	Завтрак	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
4	4	Сладости	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
5	5	Коктейли	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
6	6	Кофе	2025-09-16 18:25:37.644823+...	2025-09-16 18:25:37.644823+...
7	7	Обед	2025-09-20 11:51:55.941817+...	2025-09-20 11:51:55.941817+...

Рисунок 5 – Таблица categories до изменений

Затем вызовем функцию с Id строки, на которую нет внешних ссылок. В данном примере это строка с Id 7:

```
SELECT delete_category(7);
```

Функция отработала успешно. Теперь вызовем функцию с Id 1:

```
SELECT delete_category(1);
```

Функция была завершена с ошибкой, текст которой мы указали ранее. Результат работы функции представлен на рисунке 6:

```
ERROR:  Невозможно выполнить удаление, так как есть внешние ссылки.  
CONTEXT:  PL/pgSQL function delete_category(bigint) line 8 at RAISE  
  
SQL state: P0001
```

Рисунок 6 – Результат работы функции delete category(1)

На рисунке 7 представлена функция таблицы после изменений. Строка с индексом 7 была удалена, а с индексом 1 – нет.

	id [PK] bigint	title character varying (256)	created_at timestamp with time zone	updated_at timestamp with time zone
1	1	Пиццы	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
2	2	Закуски	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
3	3	Завтрак	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
4	4	Сладости	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
5	5	Коктейли	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...
6	6	Кофе	2025-09-20 12:24:04.293228+...	2025-09-20 12:24:04.293228+...

Рисунок 7 – Таблица categories после изменений

Задание 3

Функция будет принимать на вход числовое значение, а затем возвращать таблицу, в которой выбранное нами значение будет больше либо равно заданного нами числового значения.

Реализация функции будет для таблицы product variants. Фильтрация будет производиться по полю price.

Ниже представлена разработанная функция:

```
CREATE OR REPLACE FUNCTION filter_product_variant_by_price (
    min_val INT
)
RETURNS SETOF product_variants
AS $$
BEGIN
    RETURN QUERY (SELECT * FROM product_variants
                  WHERE price >= min_val);
END;
$$ LANGUAGE plpgsql;
```

Для проверки работы функции выполним запрос:

```
SELECT * FROM filter_product_variant_by_price(500);
```

На рисунке 8 представлена таблица, возвращаемая функцией:

	id bigint	product_id bigint	image_url character vary	size integer	volume integer	weight integer	price integer	created_at timestamp with	updated_at timestamp with
1	2	1	https://me...	25	[null]	410	659	2025-09-2...	2025-09-20 ...
2	3	1	https://me...	30	[null]	590	1009	2025-09-2...	2025-09-20 ...
3	4	1	https://me...	35	[null]	800	1119	2025-09-2...	2025-09-20 ...

Рисунок 8 – Таблица, возвращаемая функцией filter product variant by price.

Задание 4

Данная функция будет принимать на вход массив объектов составного типа данных и числовое значение, по которому будет фильтроваться массив данных и возвращаться из функции.

Для начала необходимо создать тип данных. Составной тип данных будет называться t ingredient и содержать поля, как в таблице ingredients:

```
CREATE TYPE t_ingredient AS (
    id BIGINT,
    title VARCHAR(256),
    price INT
);
```

Ниже представлена разработанная функция:

```
CREATE OR REPLACE FUNCTION filter_array_of_ingredients (
    arr t_ingredient[],
    filter_var INT
)
RETURNS t_ingredient[]
AS $$
BEGIN
    RETURN ARRAY(
        SELECT (id, title, price)::t_ingredient
        FROM ingredients
        WHERE price >= filter_var
    );
END;
$$ LANGUAGE plpgsql;
```

Для проверки воспользуемся данным запросом. Он берет данные из таблицы subject, преобразует их в массив и отправляет его в качестве параметра нашей функции:

```
SELECT filter_array_of_ingredients(
    ARRAY(SELECT (id, title, price)::t_ingredient FROM ingredients),
    60
);
```

На рисунках 9 и 10 представлено сравнение таблицы ingredients и возврата функции filter array of ingredients:

	id [PK] bigint	title character varying (256)	image_url character varying (512)	price integer	created_at timestamp with time zone	updated_at timestamp with time zone
1	1	Сырный бортик	https://cdn.dodostatic...	179	2025-09-20 1...	2025-09-20 1...
2	2	Пряная говядина	https://cdn.dodostatic...	119	2025-09-20 1...	2025-09-20 1...
3	3	Моцарелла	https://cdn.dodostatic...	79	2025-09-20 1...	2025-09-20 1...
4	4	Свежие томаты	https://cdn.dodostatic...	59	2025-09-20 1...	2025-09-20 1...
5	5	Сладкий перец	https://cdn.dodostatic...	59	2025-09-20 1...	2025-09-20 1...

Рисунок 9 – Таблица ingredients

	filter_array_of_ingredients t_ingredient[]
1	{(1,\"Сырный бортик\",179),(2,\"Пряная говядина\",119),(3,Моцарелла,79)}

Рисунок 10 – Вывод результата функции filter array of ingredients

Функция вернула массив объектов, отсортированных по значению 60.

Задание 5

Данная таблица будет содержать в себе информацию о вставке/изменении таблицы ingredients:

- Первичный ключ;
- Внешнюю ссылку на строку;
- Дату и время внесенных изменений;

- Старое значение (если был произведен UPDATE);
- Новое значение.

Ниже представлен скрипт создания таблицы:

```
CREATE TABLE log_ingredeints (  
    id BIGSERIAL PRIMARY KEY,  
    ingredient_id BIGINT,  
    change_datetime TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,  
    old_value INT,  
    new_value INT,  
  
    FOREIGN KEY (ingredient_id) REFERENCES ingredients(id) ON DELETE RESTRICT  
)
```

Создадим триггерную функцию. Она будет определять, какой вид запроса был произведен, и, в соответствии с ним, корректировать запрос. Это необходимо потому, что при INSERT нам нужно заполнить поле old value значением NULL. Функция всегда будет возвращать переменную NEW.

Ниже представлен скрипт создания функции:

```
CREATE OR REPLACE FUNCTION ingredients_trigger_func ()  
RETURNS TRIGGER  
AS $$  
DECLARE  
    old_val INT;  
  
BEGIN  
    IF (TG_OP = 'UPDATE') THEN  
        old_val := OLD.price;  
    ELSEIF (TG_OP = 'INSERT') THEN  
        old_val := NULL;  
    END IF;  
  
    INSERT INTO log_ingredeints (ingredient_id, old_value, new_value)  
    VALUES (NEW.id, old_val, NEW.price);  
  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

Теперь создадим триггер. Он будет срабатывать после изменений в таблице ingredients, так как нам нужно иметь записи об уже совершенных действиях. Триггер будет реагировать на запросы UPDATE и INSERT:

```
CREATE TRIGGER commit_ingredients_change
AFTER UPDATE OR INSERT
ON ingredients
FOR EACH ROW
EXECUTE PROCEDURE ingredients_trigger_func();
```

Чтобы проверить работу триггера, создадим новую строку в таблице ingredients, а затем изменим ее значение, подставив id новой записи:

```
INSERT INTO ingredients (title, image_url, price)
VALUES ('Острый перец халапенью',
        'https://cdn.dodostatic.net/static
        /Img/Ingredients/11ee95b6bfdf98fb88a113db92d7b3df.png', 59);
UPDATE ingredients SET price = 89 WHERE id = 6;
```

На рисунке 11 можно заметить, что в таблице log ingredients добавилось 2 записи. В первой отсутствует поле old value – она соответствует добавлению нового значения. Вторая говорит нам о том, что было изменено существующее значение.

	id [PK] bigint	ingredient_id bigint	change_datetime timestamp with time zone	old_value integer	new_value integer
1	1	6	2025-09-21 13:43:42.402748+00	[null]	59
2	2	6	2025-09-21 13:47:56.243888+00	59	89

Рисунок 11 – Вывод таблицы log ingredients

Задание 6

Создадим функцию, которая будет принимать на вход название таблицы, название столбца и id поля, которое будет выведено. На выходе будет возвращаться текстовая строка с содержимым поля.

Ниже представлена разработанная функция:

```

CREATE OR REPLACE FUNCTION get_value_by_id (
    table_name VARCHAR,
    column_name VARCHAR,
    id BIGINT
)
RETURNS TEXT
AS $$
DECLARE
    result TEXT;
BEGIN
    EXECUTE 'SELECT ' || column_name || ' FROM ' || table_name || ' WHERE id = $1' US

    RETURN result;
END;
$$ LANGUAGE plpgsql;

```

Для теста выполним скрипт:

```
SELECT get_value_by_id('categories', 'title', 4);
```

На рисунке 12 представлен результат работы функции. Функция вернула содержимое столбца «title», строки с id 4, таблицы «categories».

	get_value_by_id text
1	Сладости

Рисунок 12 – Результат работы функции get value by id.

Вывод

В ходе выполнения лабораторной работы изучены основы пользовательские функции в PostgreSQL. Созданы функции для создания или обновления данных, удаления таблицы, функция с фильтрацией данных, с использованием составного типа, триггер для логирования изменений в таблице, функцию, использующую для получения результата динамически формируемый запрос.