

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение  
высшего образования «Санкт-Петербургский политехнический  
университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 «Математика и компьютерные науки»

Отчет о выполнении лабораторных работ по предмету  
«Методы проектирования баз данных»

Студент,  
группы 5130201/20101

\_\_\_\_\_ Астафьев И. Е.

Преподаватель

\_\_\_\_\_ Попов С. Г.

«\_\_\_\_\_» \_\_\_\_\_ 2024г.

Санкт-Петербург, 2024

# Содержание

<b>Введение</b>	<b>3</b>
<b>1 Постановка задачи</b>	<b>4</b>
<b>2 Создание представления</b>	<b>5</b>
2.1 Формулировка задания . . . . .	5
2.2 Реализация . . . . .	5
2.3 Результат . . . . .	6
2.4 Запрос с использованием представления . . . . .	6
<b>3 Создание триггера</b>	<b>7</b>
3.1 Формулировка задания . . . . .	7
3.2 Реализация . . . . .	7
3.3 Результат . . . . .	11
<b>4 Создание пользователей и разделение прав</b>	<b>15</b>
4.1 Формулировка задания . . . . .	15
4.2 Реализация . . . . .	15
4.3 Результат . . . . .	17
<b>5 Создание функции</b>	<b>21</b>
5.1 Формулировка задания . . . . .	21
5.2 Реализация . . . . .	22
5.3 Результат . . . . .	22
<b>6 Создание процедуры</b>	<b>23</b>
6.1 Формулировка задания . . . . .	23
6.2 Реализация . . . . .	24
6.3 Результат . . . . .	26
<b>7 Использование транзакций</b>	<b>29</b>
7.1 Формулировка задания . . . . .	29
7.2 Реализация и результат . . . . .	30
<b>Список используемой литературы</b>	<b>33</b>

# Введение

В современном мире информация является одним из ключевых ресурсов, и эффективное управление ею становится всё более важным для организаций любого масштаба. Базы данных позволяют эффективно хранить, обрабатывать и извлекать данные. Проектирование баз данных – это процесс создания логической и физической структуры базы данных, который включает в себя определение сущностей, атрибутов, связей между ними и других элементов. Целью лабораторных работ по методам проектирования баз данных является практическое освоение основных принципов и подходов к работе с ними.

В базе данных для «Сети ресторанов быстрого питания» реализованы улучшения для упрощения и оптимизации взаимодействия с данными. Внедрены новое представление и триггеры для автоматического обновления информации, разработаны функция и процедура для удобного управления данными, а также проанализирован уровень изоляции `READ COMMITTED` и неповторяющееся чтение.

# 1 Постановка задачи

В данных лабораторных работах необходимо:

- создать представление (view) для подсчета использования ингредиентов и заказов блюд;
- создать триггер для отображения изменения количества клиентов в каждом ресторане;
- создать пользователей: первый должен иметь права на просмотр представления, а второй на просмотр, а также на вставку, удаление и редактирование строк в используемых для создания представления таблицах;
- написать функцию, которая принимает фамилию, имя и отчество и форматирует их;
- написать процедуру для проверки и дополнения данных в таблицах;
- реализовать управление транзакциями и проверить неповторяющееся чтение.

## 2 Создание представления

### 2.1 Формулировка задания

Для каждого ингредиента посчитать число блюд, в которых он используется и посчитать число заказов этих блюд.

### 2.2 Реализация

Код для создания представления:

```
1 CREATE OR REPLACE VIEW ingredient_view AS
2     SELECT
3         i.id_ingredient,
4         i.name,
5         COUNT(DISTINCT d.id_dish) AS num_dishes,
6         COUNT(o.id_order) AS num_orders
7     FROM
8         ingredient i
9     LEFT JOIN
10        ready_ingredient ri ON i.id_ingredient = ri.id_ingredient
11    LEFT JOIN
12        dish d ON ri.id_dish = d.id_dish
13    LEFT JOIN
14        list_of_dishes lod ON d.id_dish = lod.id_dish
15    LEFT JOIN
16        "order" o ON lod.id_order = o.id_order
17    GROUP BY
18        i.id_ingredient, i.name;
```

Этот запрос создаёт представление `ingredient_view`, обобщающее информацию об ингредиентах, блюдах и заказах:

- Для каждого ингредиента он показывает его ID и название.
- Считает, в скольких уникальных блюдах используется ингредиент.
- Определяет, в скольких уникальных заказах фигурируют эти блюда.

Данные собираются через соединение таблиц (`LEFT JOIN`), где ингредиенты связываются с блюдами, а блюда — с заказами. Результат группируется по идентификатору и названию ингредиента.

## 2.3 Результат

	id_ingredient integer	name character varying (50)	num_dishes bigint	num_orders bigint
1	1	Молотая говядина	3	12251
2	2	Куриные грудки	1	4639
3	3	Бекон	2	8592
4	4	Ветчина	1	4707
5	5	Сыр чеддер	3	12151
6	6	Сыр моцарелла	1	4727
7	7	Пармезан	1	4763
8	8	Ромен листья	2	8710
9	9	Томаты	0	0
10	10	Огурцы	0	0
11	11	Лук красный	2	8745
12	12	Лук белый	2	8809
13	13	Лук зеленый	6	19764
14	14	Салат Айсберг	2	8565
15	15	Жареный картофель	5	17482

Рис. 1. Таблица с представлением

Время выполнения запроса для представления: 1.704 с.

## 2.4 Запрос с использованием представления

### Запрос

Для каждого состояния ингредиента посчитать число блюд в которых оно используется.

```
1 SELECT
2     cnd.id_condition,
3     cnd.condition,
4     SUM(iv.num_dishes) AS number_of_dishes
5 FROM
6     condition cnd
7 JOIN
8     ready_ingredient ri ON cnd.id_condition = ri.id_condition
9 JOIN
```

```

10      ingredient_view iv ON ri.id_ingredient = iv.id_ingredient
11 GROUP BY
12      cnd.id_condition;

```

## Объяснение запроса

Запрос подсчитывает общее количество блюд для каждого уникального состояния, используя данные из таблицы состояний (`condition`), связывающей их с ингредиентами (`ready_ingredient`), и представления (`ingredient_view`), содержащего количество блюд для каждого ингредиента.

## Результат запроса

Результатом является список состояний с общей суммой блюд для каждого из них (Рис. 2).

	id_condition [PK] integer	condition character	number_of_dishes numeric
1	1	Сырой	134
2	2	Жареный	149
3	3	Вареный	96

Рис. 2. Результат запроса с использованием представления

## 3 Создание триггера

### 3.1 Формулировка задания

Для каждого ресторана посчитать число клиентов, которые к ним приходили, чтобы данные динамически обновлялись при добавлении или удалении клиентов в ресторане.

### 3.2 Реализация

Сначала необходимо создать и заполнить дополнительную таблицу, которая для каждого ресторана хранит количество клиентов, делавших в нем хоть раз заказ.

```

1 CREATE TABLE IF NOT EXISTS restaurant_clients (
2     id_restaurant integer NOT NULL,
3     "name" varchar(260),
4     address varchar(200),
5     clients_count integer NOT NULL
6 );
7
8 INSERT INTO restaurant_clients (id_restaurant, "name", address, clients_count)
9     SELECT
10         restaurant.id_restaurant,
11         restaurant."name",

```

```

12         restaurant.address,
13         COALESCE(count(client_order.id_client), 0) as clients_count
14     FROM
15         restaurant
16         LEFT JOIN
17         "order"
18         ON restaurant.id_restaurant = "order".id_restaurant
19         LEFT JOIN
20         client_order
21         ON "order".id_order = client_order.id_order
22     GROUP BY restaurant.id_restaurant;

```

Этот SQL-запрос выполняет две задачи:

1. Создание таблицы:

Создает таблицу `restaurant_clients`, если она еще не существует, со столбцами для идентификатора ресторана, названия, адреса и количества клиентов.

2. Вставка данных:

- Заполняет таблицу данными, извлекая их из таблиц `restaurant`, `order` и `client_order`.
- Считает количество клиентов для каждого ресторана, основываясь на заказах, и вставляет эту информацию вместе с названием и адресом ресторана в новую таблицу.

Затем создается триггер и функция, которые обрабатывают изменения в таблице `client_order`.

```

1  -- По номеру заказа получаем информацию об id ресторана, в котором он был сделан,
2  -- затем увеличиваем/уменьшаем число клиентов
3  CREATE OR REPLACE FUNCTION update_client_count()
4  RETURNS TRIGGER
5  LANGUAGE plpgsql
6  AS
7  $$
8  DECLARE
9      restaurant_id INT;
10     client_id INT;
11     count_clients INT;
12 BEGIN
13     IF TG_OP = 'INSERT' THEN
14         SELECT id_restaurant INTO restaurant_id
15         FROM "order"
16         WHERE "order".id_order = NEW.id_order;
17
18         SELECT id_client INTO client_id
19         FROM client_order
20         WHERE client_order.id_order = NEW.id_order;
21

```



```

22      -- Пересчитываем count_clients после вставки
23      SELECT COUNT(DISTINCT client_order.id_order) INTO count_clients
24      FROM client_order
25      JOIN "order" ON client_order.id_order = "order".id_order
26      WHERE "order".id_restaurant = restaurant_id AND client_order.id_client = client_id;
27
28      IF count_clients = 1 THEN
29          -- Увеличиваем кол-во клиентов
30          UPDATE restaurant_clients
31          SET clients_count = clients_count + 1
32          WHERE id_restaurant = restaurant_id;
33      END IF;
34
35      ELSIF TG_OP = 'DELETE' THEN
36          SELECT id_restaurant INTO restaurant_id
37          FROM "order"
38          WHERE "order".id_order = OLD.id_order;
39
40          SELECT id_client INTO client_id
41          FROM client_order
42          WHERE client_order.id_order = OLD.id_order;
43
44          -- Пересчитываем count_clients после удаления
45          SELECT COUNT(DISTINCT client_order.id_order) INTO count_clients
46          FROM client_order
47          JOIN "order" ON client_order.id_order = "order".id_order
48          WHERE "order".id_restaurant = restaurant_id AND client_order.id_client = client_id;
49
50          IF count_clients = 0 THEN
51              -- Уменьшаем кол-во клиентов
52              UPDATE restaurant_clients
53              SET clients_count = clients_count - 1
54              WHERE id_restaurant = restaurant_id;
55          END IF;
56      END IF;
57
58      RETURN NULL;
59  END;
60  $$;
61
62  -- Триггер
63  CREATE OR REPLACE TRIGGER after_update_client_order
64      AFTER INSERT OR DELETE ON client_order
65      FOR EACH ROW
66      EXECUTE FUNCTION update_client_count();

```

Этот SQL-запрос создает функцию и триггер для автоматического обновления количества клиентов в ресторане:

1. Функция `update_client_count` обрабатывает вставку (`INSERT`) или удаление

(DELETE) записей в таблицу `client_order`.

- При вставке: получает ID ресторана и клиента; если это первый заказ клиента в ресторане, увеличивает количество клиентов в таблице `restaurant_clients`.
  - При удалении: если это был единственный заказ клиента в ресторане, уменьшает количество клиентов.
2. Триггер `after_update_client_order`: запускается после вставки или удаления записей в `client_order` и активирует функцию `update_client_count` для соответствующей обработки.

В результате автоматически поддерживается корректное количество клиентов для каждого ресторана.

После этого создается функция и триггер для обработки изменений в таблице `restaurant`.

```
1 CREATE OR REPLACE FUNCTION update_restaurant_clients()
2 RETURNS TRIGGER
3 LANGUAGE plpgsql
4 AS $$
5 BEGIN
6     IF TG_OP = 'INSERT' THEN
7         -- Вставляет новую строку в restaurant_clients с числом клиентов по умолчанию
7         -- ↪ равным 0
8         INSERT INTO restaurant_clients (id_restaurant, "name", address, clients_count)
9         VALUES (NEW.id_restaurant, NEW."name", NEW.address, 0);
10
11     ELSIF TG_OP = 'UPDATE' THEN
12         -- Обновляет соответствующую строку в restaurant_clients
13         UPDATE restaurant_clients
14         SET "name" = NEW."name", address = NEW.address
15         WHERE id_restaurant = OLD.id_restaurant;
16
17     ELSIF TG_OP = 'DELETE' THEN
18         -- Удаляет соответствующую строку из restaurant_clients
19         DELETE FROM restaurant_clients
20         WHERE id_restaurant = OLD.id_restaurant;
21     END IF;
22     RETURN NULL;
23 END;
24 $$;
25
26
27 -- Триггер для вставки и удаления
28 CREATE TRIGGER restaurant_clients_insert
29 AFTER INSERT OR DELETE ON restaurant
30 FOR EACH ROW EXECUTE FUNCTION update_restaurant_clients();
31
32 -- Триггер для обновления названия или адреса ресторана
33 CREATE TRIGGER restaurant_clients_update
```

```
34 AFTER UPDATE OF name, address ON restaurant
35 FOR EACH ROW EXECUTE FUNCTION update_restaurant_clients();
```

Этот SQL-запрос создает функцию и триггеры для автоматического управления информацией о ресторанах и их клиентах в базе данных:

1. Функция `update_restaurant_clients` обрабатывает операции вставки (`INSERT`), обновления (`UPDATE`) и удаления (`DELETE`) записей в таблице `restaurant`.
  - При вставке: добавляет новую запись в таблицу `restaurant_clients` с указанием идентификатора ресторана, имени, адреса и с числом клиентов, равным 0. Это создаёт запись для нового ресторана с изначальным нулевым количеством клиентов.
  - При обновлении: обновляет соответствующую запись в таблице `restaurant_clients` с новыми значениями имени и адреса ресторана. Это обеспечивает актуальность данных о ресторане при изменении этих полей.
  - При удалении: удаляет запись из `restaurant_clients`, соответствующую ресторану, который удаляется из `restaurant`. Таким образом, удалённые рестораны не будут отображаться в списке клиентов.
2. Триггер `restaurant_clients_insert`: срабатывает после вставки или удаления записей в таблице `restaurant`. Он активирует функцию `update_restaurant_clients` для обработки соответствующих изменений.
3. Триггер `restaurant_clients_update`: запускается после обновления полей `name` или `address` в таблице `restaurant`. Он также активирует функцию `update_restaurant_clients` для поддержания актуальной информации в таблице `restaurant_clients`.

В результате автоматически поддерживается актуальная информация о ресторанах и их клиентах, включая добавление, обновление и удаление данных.

### 3.3 Результат

Как можно видеть на Рис. 3, у ресторана с ID=47 `clients_count`=597 (число клиентов).

	<b>id_restaurant</b> integer	<b>name</b> character varying (260)	<b>address</b> character varying (200)	<b>clients_count</b> integer
37	22	Бургерное Вре...	Пермь, ул. Механическая, д. 6	500
38	13	Блинная Сказка	Челябинск, ул. Сибирская, д. 56	540
39	33	Индийский Царь	Махачкала, ул. Специй, д. 61	627
40	1	Бургерный Рай	Москва, ул. Ленина, д. 10	657
41	5	Тостерия	Казань, ул. Спортивная, д. 5	648
42	18	Лапша Лаб	Ярославль, ул. Чайковского, д. 7	593
43	2	Суши Весна	Санкт-Петербург, Невский проспект, д. 22	627
44	16	Сицилийские Каникулы	Саратов, ул. Итальянская, д. 12	610
45	27	Сладкий Район	Пенза, ул. Сахарова, д. 17	618
46	23	Зелёный Дом	Краснодар, ул. Зеленая, д. 10	594
47	44	Жареная Веселость	Мурманск, ул. Полярная, д. 22	651
48	11	Веселые Пельмени	Самара, ул. Волжская, д. 48	576
49	8	Закусочная у Джо	Владивосток, ул. Океанская, д. 21	595
50	47	Окрошечная Столовая	Сыктывкар, ул. Холодильная, д. 10	597

Рис. 3. Промежуточная таблица `restaurant_clients` до добавления клиента

Затем добавим одного клиента следующим запросом:

```

1 INSERT INTO client(id_client, surname, "name", patronymic, phone_number)
2   VALUES(500, 'АСТАФЬЕВ', 'ИГОРЬ', 'ЕВГЕНЬЕВИЧ', '+799999999999999');
3 INSERT INTO "order"(id_order, order_type, id_restaurant) VALUES(150000,
4   ↳ 'takeaway'::order_type, 47);
5 INSERT INTO client_order(id_client_order, id_client, id_order) VALUES(30001, 500,
6   ↳ 150000);

```

Как можно видеть теперь на Рис. 4, в таблице для ресторана с ID=47 число клиентов увеличилось на 1.

	<b>id_restaurant</b> integer	<b>name</b> character varying (260)	<b>address</b> character varying (200)	<b>clients_count</b> integer
37	22	Бургерное Вре...	Пермь, ул. Механическая, д. 6	500
38	13	Блинная Сказка	Челябинск, ул. Сибирская, д. 56	540
39	33	Индийский Царь	Махачкала, ул. Специй, д. 61	627
40	1	Бургерный Рай	Москва, ул. Ленина, д. 10	657
41	5	Тостерия	Казань, ул. Спортивная, д. 5	648
42	18	Лапша Лаб	Ярославль, ул. Чайковского, д. 7	593
43	2	Суши Весна	Санкт-Петербург, Невский проспект, д. 22	627
44	16	Сицилийские Каникулы	Саратов, ул. Итальянская, д. 12	610
45	27	Сладкий Район	Пенза, ул. Сахарова, д. 17	618
46	23	Зелёный Дом	Краснодар, ул. Зеленая, д. 10	594
47	44	Жареная Веселость	Мурманск, ул. Полярная, д. 22	651
48	11	Веселые Пельмени	Самара, ул. Волжская, д. 48	576
49	8	Закусочная у Джо	Владивосток, ул. Океанская, д. 21	595
50	47	Окрошечная Столовая	Сыктывкар, ул. Холодильная, д. 10	598

Рис. 4. Промежуточная таблица `restaurant_clients` после добавления клиента

Теперь удалим недавно добавленного клиента.

```
1 DELETE FROM client WHERE id_client = 500;
```

Так как включено каскадное удаление, то все связанные с данным клиентом строки в других таблицах так же удалятся, в том числе и в таблице `client_order`. Как можно видеть на Рис. 5, количество клиентов для ресторана с ID=47 уменьшилось на 1 и снова стало 597.

	id_restaurant integer	name character varying (260)	address character varying (200)	clients_count integer
38	13	Блинная Сказка	Челябинск, ул. Сибирская, д. 56	540
39	33	Индийский Царь	Махачкала, ул. Специй, д. 61	627
40	1	Бургерный Рай	Москва, ул. Ленина, д. 10	657
41	5	Тостерия	Казань, ул. Спортивная, д. 5	648
42	18	Лапша Лаб	Ярославль, ул. Чайковского, д. 7	593
43	2	Суши Весна	Санкт-Петербург, Невский проспект, д. 22	627
44	16	Сицилийские Каникулы	Саратов, ул. Итальянская, д. 12	610
45	27	Сладкий Район	Пенза, ул. Сахарова, д. 17	618
46	23	Зелёный Дом	Краснодар, ул. Зеленая, д. 10	594
47	44	Жареная Веселость	Мурманск, ул. Полярная, д. 22	651
48	11	Веселые Пельмени	Самара, ул. Волжская, д. 48	576
49	8	Закусочная у Джо	Владивосток, ул. Океанская, д. 21	595
50	47	Окрошечная Столовая	Сыктывкар, ул. Холодильная, д. 10	597

Рис. 5. Промежуточная таблица `restaurant_clients` после удаления клиента

Теперь необходимо проверить триггеры, срабатывающие при изменении в таблице `restaurant`. Добавление нового ресторана.

```
1 INSERT INTO restaurant (id_restaurant, name, registration_info, address, space,
  ↳ capacity)
2 VALUES (51, 'ГастроLAB', 'ИНН 8901234567', 'Новосибирск, ул. Ленина, д. 20', 120, 150);
```

	id_restaurant integer	name character varying (260)	address character varying (200)	clients_count integer
1	51	ГастроLAB	Новосибирск, ул. Ленина, д. 20	0
2	42	Сладкая Жизнь	Ставрополь, ул. Кондитерская, д. 12	612
3	29	Паста Плюс	Владимир, ул. Равиоли, д. 21	636
4	4	Кафе на Речном	Нижний Новгород, Речная наб., д. 8	617
5	34	Тайный Уголок	Великий Новгород, ул. Тайландская, д. 9	606
6	41	Дары Моря	Керчь, ул. Морская, д. 6	572
7	46	Минутка Паузы	Тула, ул. Перекусовая, д. 24	589

Рис. 6. Промежуточная таблица `restaurant_clients` после добавления ресторана

Как можно увидеть на Рис. 6, в таблице `restaurant_clients` появилась новая строка с добавленным рестораном. Теперь изменим название ресторана с ID=4, который как видно на Рис. 6, называется «Кафе на речном».

```
1 UPDATE restaurant
2 SET name = 'Вкусный берег'
3 WHERE id_restaurant = 4;
```

После изменения названия ресторана на «Вкусный берег» в таблице `restaurant_clients` название так же поменялось, что можно увидеть на Рис. 7.

	<code>id_restaurant</code> integer	<code>name</code> character varying (260)	<code>address</code> character varying (200)	<code>clients_count</code> integer
1	51	ГастроЛАВ	Новосибирск, ул. Ленина, д. 20	0
2	4	Вкусный берег	Нижний Новгород, Речная наб., д. 8	617
3	42	Сладкая Жизнь	Ставрополь, ул. Кондитерская, д. 12	612
4	29	Паста Плюс	Владимир, ул. Равиоли, д. 21	636
5	34	Тайный Уголок	Великий Новгород, ул. Тайландская, д. 9	606
6	41	Дары Моря	Керчь, ул. Морская, д. 6	572
7	46	Минутка Паузы	Тула, ул. Перекусовая, д. 24	589

Рис. 7. Промежуточная таблица `restaurant_clients` после изменения названия ресторана

Теперь удалим ресторан, который до этого был добавлен (ID=51).

```
1 DELETE FROM restaurant WHERE id_restaurant = 51;
```

Как можно видеть на Рис. 8 в таблице `restaurant_clients` больше нет ресторана с ID=51.

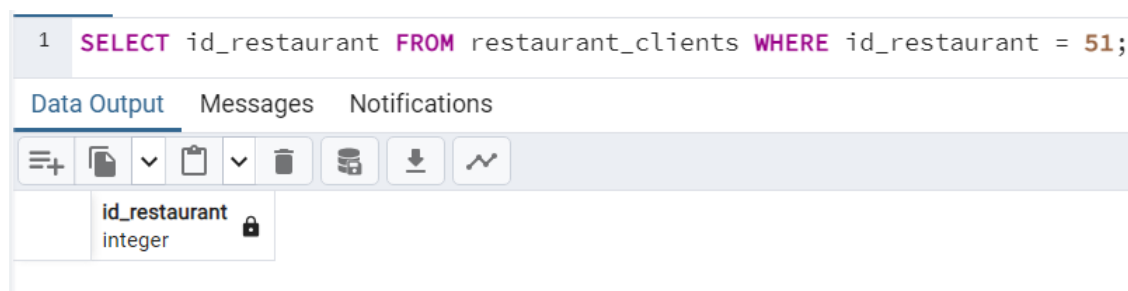


Рис. 8. Промежуточная таблица `restaurant_clients` после удаления ресторана

Для демонстрации наличия триггеров на Рис. 9 представлены триггеры из таблицы `information_schema`, которая содержит метаданные о базе данных. Чтобы увидеть эти данные, необходимо выполнить следующий запрос:

```
1 SELECT
2     trigger_name,
3     event_object_table,
```

```

4      event_manipulation,
5      action_statement
6 FROM information_schema.triggers;

```

	trigger_name name	event_object_table name	event_manipulation character varying	action_statement character varying
1	after_update_client_order	client_order	INSERT	EXECUTE FUNCTION update_client_count()
2	after_update_client_order	client_order	DELETE	EXECUTE FUNCTION update_client_count()
3	restaurant_clients_insert	restaurant	INSERT	EXECUTE FUNCTION update_restaurant_clients()
4	restaurant_clients_insert	restaurant	DELETE	EXECUTE FUNCTION update_restaurant_clients()
5	restaurant_clients_update	restaurant	UPDATE	EXECUTE FUNCTION update_restaurant_clients()

Рис. 9. Триггер в таблице `information_schema`

Как можно видеть, для таблицы `client_order` существует триггер на события `INSERT` и `DELETE`, при срабатывании которого выполняется функция `update_client_count()`, для таблицы `restaurant` - триггер на события `INSERT`, `DELETE` и `UPDATE`, при срабатывании которых выполняется функция `update_restaurant_clients()`.

## 4 Создание пользователей и разделение прав

### 4.1 Формулировка задания

Создать двух пользователей, один из которых имеет право на просмотр созданного представления `ingredients_view`, а второй на просмотр `textttingredients_view` а также просмотр, вставку, удаление и редактирование строк в таблицах, используемых для создания этого представления.

### 4.2 Реализация

```

1  -- Создание пользователя butters и назначение прав
2  CREATE USER butters WITH PASSWORD '1234';
3  GRANT SELECT ON ingredient_view TO butters;
4
5  -- Создание пользователя cartman и назначение прав
6  CREATE USER cartman WITH PASSWORD '1234';
7  GRANT SELECT, INSERT, DELETE, UPDATE ON ingredient TO cartman;
8  GRANT SELECT, INSERT, DELETE, UPDATE ON "order" TO cartman;
9  GRANT SELECT, INSERT, DELETE, UPDATE ON ready_ingredient TO cartman;
10 GRANT SELECT, INSERT, DELETE, UPDATE ON dish TO cartman;
11 GRANT SELECT, INSERT, DELETE, UPDATE ON list_of_dishes TO cartman;
12 GRANT SELECT ON ingredient_view TO cartman;

```

В данном запросе:

- Создаётся пользователь `butters` с паролем, предоставляется право на чтение из `ingredient_view`.

- Создаётся пользователь `cartman` с полными правами (чтение, вставка, удаление, обновление) на таблицы `ingredient`, `order`, `ready_ingredient`, `dish`, `list_of_dishes` и правом на чтение из `ingredient_view`.

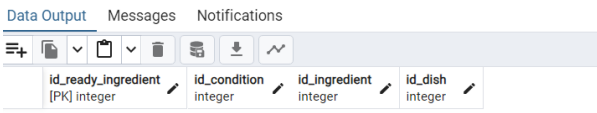
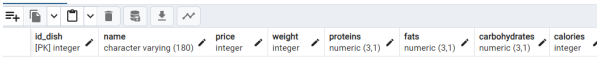

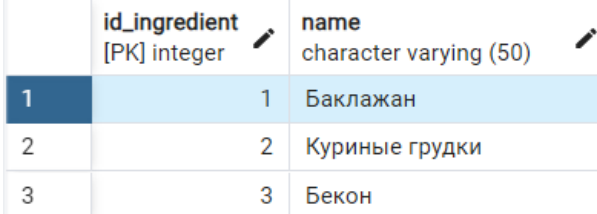
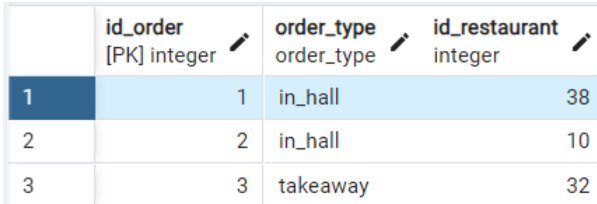


## 4.3 Результат

№	butters	cartman																																							
1	Выборка всех записей из ingredient_view																																								
	SELECT * FROM ingredient_view																																								
	<table><thead><tr><th></th><th>id_ingredient integer</th><th>name character varying (50)</th><th>num_dishes bigint</th><th>num_orders bigint</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Молотая говядина</td><td>3</td><td>15333</td></tr><tr><td>2</td><td>2</td><td>Куриные грудки</td><td>1</td><td>5003</td></tr><tr><td>3</td><td>3</td><td>Бекон</td><td>2</td><td>10040</td></tr></tbody></table>		id_ingredient integer	name character varying (50)	num_dishes bigint	num_orders bigint	1	1	Молотая говядина	3	15333	2	2	Куриные грудки	1	5003	3	3	Бекон	2	10040	<table><thead><tr><th></th><th>id_ingredient integer</th><th>name character varying (50)</th><th>num_dishes bigint</th><th>num_orders bigint</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Молотая говядина</td><td>3</td><td>15333</td></tr><tr><td>2</td><td>2</td><td>Куриные грудки</td><td>1</td><td>5003</td></tr><tr><td>3</td><td>3</td><td>Бекон</td><td>2</td><td>10040</td></tr></tbody></table>		id_ingredient integer	name character varying (50)	num_dishes bigint	num_orders bigint	1	1	Молотая говядина	3	15333	2	2	Куриные грудки	1	5003	3	3	Бекон	2
	id_ingredient integer	name character varying (50)	num_dishes bigint	num_orders bigint																																					
1	1	Молотая говядина	3	15333																																					
2	2	Куриные грудки	1	5003																																					
3	3	Бекон	2	10040																																					
	id_ingredient integer	name character varying (50)	num_dishes bigint	num_orders bigint																																					
1	1	Молотая говядина	3	15333																																					
2	2	Куриные грудки	1	5003																																					
3	3	Бекон	2	10040																																					
2	Выборка всех записей из ingredient																																								
	SELECT * FROM ingredient																																								
	<div>ERROR: нет доступа к таблице ingredient</div> <div>ОШИБКА: нет доступа к таблице ingredient</div> <div>SQL state: 42501</div>	<table><thead><tr><th></th><th>id_ingredient [PK] integer</th><th>name character varying (50)</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Молотая говядина</td></tr><tr><td>2</td><td>2</td><td>Куриные грудки</td></tr><tr><td>3</td><td>3</td><td>Бекон</td></tr></tbody></table>		id_ingredient [PK] integer	name character varying (50)	1	1	Молотая говядина	2	2	Куриные грудки	3	3	Бекон																											
	id_ingredient [PK] integer	name character varying (50)																																							
1	1	Молотая говядина																																							
2	2	Куриные грудки																																							
3	3	Бекон																																							
3	Выборка всех записей из order																																								
	SELECT * FROM "order"																																								
	<div>ERROR: нет доступа к таблице order</div> <div>ОШИБКА: нет доступа к таблице order</div> <div>SQL state: 42501</div>	<table><thead><tr><th></th><th>id_order [PK] integer</th><th>order_type order_type</th><th>id_restaurant integer</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>takeaway</td><td>38</td></tr><tr><td>2</td><td>150000</td><td>takeaway</td><td>47</td></tr><tr><td>3</td><td>2</td><td>in_hall</td><td>10</td></tr></tbody></table>		id_order [PK] integer	order_type order_type	id_restaurant integer	1	1	takeaway	38	2	150000	takeaway	47	3	2	in_hall	10																							
	id_order [PK] integer	order_type order_type	id_restaurant integer																																						
1	1	takeaway	38																																						
2	150000	takeaway	47																																						
3	2	in_hall	10																																						
4	Выборка всех записей из ready_ingredient																																								
	SELECT * FROM ready_ingredient																																								
	<div>ERROR: нет доступа к таблице ready_ingredient</div> <div>ОШИБКА: нет доступа к таблице ready_ingredient</div> <div>SQL state: 42501</div>	<table><thead><tr><th></th><th>id_ready_ingredient [PK] integer</th><th>id_condition integer</th><th>id_ingredient integer</th><th>id_dish integer</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>1</td><td>5</td><td>1</td></tr><tr><td>2</td><td>2</td><td>1</td><td>59</td><td>1</td></tr><tr><td>3</td><td>3</td><td>1</td><td>33</td><td>2</td></tr></tbody></table>		id_ready_ingredient [PK] integer	id_condition integer	id_ingredient integer	id_dish integer	1	1	1	5	1	2	2	1	59	1	3	3	1	33	2																			
	id_ready_ingredient [PK] integer	id_condition integer	id_ingredient integer	id_dish integer																																					
1	1	1	5	1																																					
2	2	1	59	1																																					
3	3	1	33	2																																					
5	Выборка всех записей из dish																																								
	SELECT * FROM dish																																								
	<div>ERROR: нет доступа к таблице dish</div> <div>ОШИБКА: нет доступа к таблице dish</div> <div>SQL state: 42501</div>	<table><thead><tr><th></th><th>id_dish [PK] integer</th><th>name character varying (180)</th><th>price integer</th><th>weight integer</th><th>proteins numeric (3,1)</th><th>fats numeric (3,1)</th><th>carbohydrates numeric (3,1)</th><th>calories integer</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Гамбургер классический</td><td>112</td><td>241</td><td>5.0</td><td>26.0</td><td>12.0</td><td>180</td></tr><tr><td>2</td><td>2</td><td>Чизбургер</td><td>268</td><td>283</td><td>12.0</td><td>10.0</td><td>16.0</td><td>750</td></tr><tr><td>3</td><td>3</td><td>Биг Бургер</td><td>162</td><td>267</td><td>22.0</td><td>18.0</td><td>26.0</td><td>398</td></tr></tbody></table>		id_dish [PK] integer	name character varying (180)	price integer	weight integer	proteins numeric (3,1)	fats numeric (3,1)	carbohydrates numeric (3,1)	calories integer	1	1	Гамбургер классический	112	241	5.0	26.0	12.0	180	2	2	Чизбургер	268	283	12.0	10.0	16.0	750	3	3	Биг Бургер	162	267	22.0	18.0	26.0	398			
	id_dish [PK] integer	name character varying (180)	price integer	weight integer	proteins numeric (3,1)	fats numeric (3,1)	carbohydrates numeric (3,1)	calories integer																																	
1	1	Гамбургер классический	112	241	5.0	26.0	12.0	180																																	
2	2	Чизбургер	268	283	12.0	10.0	16.0	750																																	
3	3	Биг Бургер	162	267	22.0	18.0	26.0	398																																	

№	butters	cartman																			
6	Выборка всех записей из list_of_dishes																				
	SELECT * FROM list_of_dishes																				
	<div>ERROR: нет доступа к таблице list_of_dishes</div> <div>ОШИБКА: нет доступа к таблице list_of_dishes</div> <div>SQL state: 42501</div>	<table><tr><th></th><th>id_list_of_dishes [PK] integer</th><th>id_dish integer</th><th>id_order integer</th></tr><tr><td>1</td><td>1</td><td>31</td><td>1</td></tr><tr><td>2</td><td>2</td><td>12</td><td>1</td></tr><tr><td>3</td><td>3</td><td>48</td><td>1</td></tr></table>		id_list_of_dishes [PK] integer	id_dish integer	id_order integer	1	1	31	1	2	2	12	1	3	3	48	1			
	id_list_of_dishes [PK] integer	id_dish integer	id_order integer																		
1	1	31	1																		
2	2	12	1																		
3	3	48	1																		
7	Вставка строки в ingredient																				
	INSERT INTO ingredient (id_ingredient, name) VALUES (61, Мясо лягушки);																				
	<div>ERROR: нет доступа к таблице ingredient</div> <div>ОШИБКА: нет доступа к таблице ingredient</div> <div>SQL state: 42501</div>	<table><tr><th></th><th>id_ingredient [PK] integer</th><th>name character varying (50)</th></tr><tr><td>59</td><td>59</td><td>Спаржа</td></tr><tr><td>60</td><td>60</td><td>Сливки</td></tr><tr><td>61</td><td>61</td><td>Мясо лягушки</td></tr></table>		id_ingredient [PK] integer	name character varying (50)	59	59	Спаржа	60	60	Сливки	61	61	Мясо лягушки							
	id_ingredient [PK] integer	name character varying (50)																			
59	59	Спаржа																			
60	60	Сливки																			
61	61	Мясо лягушки																			
8	Вставка строки в order																				
	INSERT INTO "order"(id_order, order_type, id_restaurant) VALUES (30002, 'in_hall'::order_type, 3);																				
	<div>ERROR: нет доступа к таблице order</div> <div>ОШИБКА: нет доступа к таблице order</div> <div>SQL state: 42501</div>	<table><tr><th></th><th>id_order [PK] integer</th><th>order_type order_type</th><th>id_restaurant integer</th></tr><tr><td>1</td><td>150000</td><td>takeaway</td><td>47</td></tr><tr><td>2</td><td>30002</td><td>in_hall</td><td>3</td></tr><tr><td>3</td><td>30000</td><td>takeaway</td><td>32</td></tr><tr><td>4</td><td>29999</td><td>takeaway</td><td>45</td></tr></table>		id_order [PK] integer	order_type order_type	id_restaurant integer	1	150000	takeaway	47	2	30002	in_hall	3	3	30000	takeaway	32	4	29999	takeaway
	id_order [PK] integer	order_type order_type	id_restaurant integer																		
1	150000	takeaway	47																		
2	30002	in_hall	3																		
3	30000	takeaway	32																		
4	29999	takeaway	45																		
9	Вставка строки в ready_ingredient																				
	INSERT INTO ready_ingredient (id_ready_ingredient, id_condition, id_ingredient, id_dish) VALUES (126, 1, 1, 1);																				
	<div>ERROR: нет доступа к таблице ready_ingredient</div> <div>ОШИБКА: нет доступа к таблице ready_ingredient</div> <div>SQL state: 42501</div>	<table><tr><th></th><th>id_ready_ingredient [PK] integer</th><th>id_condition integer</th><th>id_ingredient integer</th><th>id_dish integer</th></tr><tr><td>124</td><td>124</td><td>1</td><td>57</td><td>50</td></tr><tr><td>125</td><td>125</td><td>3</td><td>26</td><td>50</td></tr><tr><td>126</td><td>126</td><td>1</td><td>1</td><td>1</td></tr></table>		id_ready_ingredient [PK] integer	id_condition integer	id_ingredient integer	id_dish integer	124	124	1	57	50	125	125	3	26	50	126	126	1	1
	id_ready_ingredient [PK] integer	id_condition integer	id_ingredient integer	id_dish integer																	
124	124	1	57	50																	
125	125	3	26	50																	
126	126	1	1	1																	

№	butters	cartman																																							
10	Вставка строки в dish																																								
	INSERT INTO dish (id_dish, name, price, weight, proteins, fats, carbohydrates, calories) VALUES (51, 'Мясо по-французски', 1234, 100, 1, 2, 3, 100);																																								
	<div>ERROR: нет доступа к таблице dish</div> <div>ОШИБКА: нет доступа к таблице dish</div> <div>SQL state: 42501</div>	<table><tr><th>id_dish</th><th>name</th><th>price</th><th>weight</th><th>proteins</th><th>fats</th><th>carbohydrates</th><th>calories</th></tr><tr><td>[PK] integer</td><td>character varying (180)</td><td>integer</td><td>integer</td><td>numeric (3,1)</td><td>numeric (3,1)</td><td>numeric (3,1)</td><td>integer</td></tr><tr><td>49</td><td>Картофельные дольки в остром соусе</td><td>177</td><td>295</td><td>7.0</td><td>25.0</td><td>6.0</td><td>655</td></tr><tr><td>50</td><td>Корн-дог</td><td>265</td><td>275</td><td>32.0</td><td>9.0</td><td>22.0</td><td>255</td></tr><tr><td>51</td><td>Мясо по-французски</td><td>1234</td><td>100</td><td>1.0</td><td>2.0</td><td>3.0</td><td>100</td></tr></table>	id_dish	name	price	weight	proteins	fats	carbohydrates	calories	[PK] integer	character varying (180)	integer	integer	numeric (3,1)	numeric (3,1)	numeric (3,1)	integer	49	Картофельные дольки в остром соусе	177	295	7.0	25.0	6.0	655	50	Корн-дог	265	275	32.0	9.0	22.0	255	51	Мясо по-французски	1234	100	1.0	2.0	3.0
id_dish	name	price	weight	proteins	fats	carbohydrates	calories																																		
[PK] integer	character varying (180)	integer	integer	numeric (3,1)	numeric (3,1)	numeric (3,1)	integer																																		
49	Картофельные дольки в остром соусе	177	295	7.0	25.0	6.0	655																																		
50	Корн-дог	265	275	32.0	9.0	22.0	255																																		
51	Мясо по-французски	1234	100	1.0	2.0	3.0	100																																		
11	Вставка строки в list_of_dishes																																								
	INSERT INTO list_of_dishes (id_list_of_dishes, id_dish, id_order) VALUES (254873, 1, 2);																																								
	<div>ERROR: нет доступа к таблице list_of_dishes</div> <div>ОШИБКА: нет доступа к таблице list_of_dishes</div> <div>SQL state: 42501</div>	<table><tr><th></th><th>id_list_of_dishes</th><th>id_dish</th><th>id_order</th></tr><tr><td></td><td>[PK] integer</td><td>integer</td><td>integer</td></tr><tr><td>1</td><td>254873</td><td>1</td><td>2</td></tr><tr><td>2</td><td>254872</td><td>12</td><td>30000</td></tr><tr><td>3</td><td>254871</td><td>41</td><td>30000</td></tr></table>		id_list_of_dishes	id_dish	id_order		[PK] integer	integer	integer	1	254873	1	2	2	254872	12	30000	3	254871	41	30000																			
	id_list_of_dishes	id_dish	id_order																																						
	[PK] integer	integer	integer																																						
1	254873	1	2																																						
2	254872	12	30000																																						
3	254871	41	30000																																						
12	Удаление из ingredient																																								
	DELETE FROM ingredient WHERE id_ingredient = 61;																																								
	<div>ERROR: нет доступа к таблице ingredient</div> <div>ОШИБКА: нет доступа к таблице ingredient</div> <div>SQL state: 42501</div>	<div><div>1</div><div>SELECT * FROM ingredient WHERE id_ingredient = 61;</div></div> <div><div>Data Output</div><div>Messages</div><div>Notifications</div></div> <div><div><div>≡</div><div>📄</div><div>▼</div><div>📋</div><div>▼</div><div>🗑</div><div>📦</div><div>⬇</div><div>📶</div></div><div><table><tr><th>id_ingredient</th><th>name</th></tr><tr><td>[PK] integer</td><td>character varying (50)</td></tr></table></div></div>	id_ingredient	name	[PK] integer	character varying (50)																																			
id_ingredient	name																																								
[PK] integer	character varying (50)																																								
13	Удаление из order																																								
	DELETE FROM "order"WHERE id_order = 30002;																																								
	<div>ERROR: нет доступа к таблице order</div> <div>ОШИБКА: нет доступа к таблице order</div> <div>SQL state: 42501</div>	<div><div>1</div><div>SELECT * FROM "order" WHERE id_order = 30002;</div></div> <div><div>Data Output</div><div>Messages</div><div>Notifications</div></div> <div><div><div>≡</div><div>📄</div><div>▼</div><div>📋</div><div>▼</div><div>🗑</div><div>📦</div><div>⬇</div><div>📶</div></div><div><table><tr><th>id_order</th><th>order_type</th><th>id_restaurant</th></tr><tr><td>[PK] integer</td><td>order_type</td><td>integer</td></tr></table></div></div>	id_order	order_type	id_restaurant	[PK] integer	order_type	integer																																	
id_order	order_type	id_restaurant																																							
[PK] integer	order_type	integer																																							

№	butters	cartman
14	Удаление из ready_ingredient	
	DELETE FROM ready_ingredient WHERE id_ready_ingredient = 126;	
	<p>ERROR: нет доступа к таблице ready_ingredient</p> <p>ОШИБКА: нет доступа к таблице ready_ingredient SQL state: 42501</p>	<pre>1 SELECT * FROM ready_ingredient WHERE id_ready_ingredient = 126;</pre> 
15	Удаление из dish	
	DELETE FROM dish WHERE id_dish = 51;	
	<p>ERROR: нет доступа к таблице dish</p> <p>ОШИБКА: нет доступа к таблице dish SQL state: 42501</p>	<pre>1 SELECT * FROM dish WHERE id_dish = 51;</pre> 
16	Удаление из list_of_dishes	
	DELETE FROM list_of_dishes WHERE id_list_of_dishes = 254873;	
	<p>ERROR: нет доступа к таблице list_of_dishes</p> <p>ОШИБКА: нет доступа к таблице list_of_dishes SQL state: 42501</p>	<pre>1 SELECT * FROM list_of_dishes WHERE id_list_of_dishes = 254873;</pre> 
17	Обновление данных ingredient	
	UPDATE ingredient SET name = 'Баклажан' WHERE id_ingredient = 1;	
	<p>ERROR: нет доступа к таблице ingredient</p> <p>ОШИБКА: нет доступа к таблице ingredient SQL state: 42501</p>	
18	Обновление данных order	
	UPDATE "order" SET order_type = 'in_hall'::order_type WHERE id_order = 1;	
	<p>ERROR: нет доступа к таблице order</p> <p>ОШИБКА: нет доступа к таблице order SQL state: 42501</p>	

№	butters	cartman																															
19	Обновление данных ready_ingredient																																
	UPDATE ready_ingredient SET id_ingredient = 1 WHERE id_ready_ingredient = 1;																																
	<div>ERROR: нет доступа к таблице ready_ingredient</div> <div>ОШИБКА: нет доступа к таблице ready_ingredient SQL state: 42501</div>	<table><tr><th></th><th>id_ready_ingredient [PK] integer</th><th>id_condition integer</th><th>id_ingredient integer</th><th>id_dish integer</th></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td><td>1</td><td>59</td><td>1</td></tr><tr><td>3</td><td>3</td><td>1</td><td>33</td><td>2</td></tr></table>		id_ready_ingredient [PK] integer	id_condition integer	id_ingredient integer	id_dish integer	1	1	1	1	1	2	2	1	59	1	3	3	1	33	2											
	id_ready_ingredient [PK] integer	id_condition integer	id_ingredient integer	id_dish integer																													
1	1	1	1	1																													
2	2	1	59	1																													
3	3	1	33	2																													
20	Обновление данных dish																																
	UPDATE dish SET name = 'Сибирский бургер' WHERE id_dish = 1;																																
	<div>ERROR: нет доступа к таблице dish</div> <div>ОШИБКА: нет доступа к таблице dish SQL state: 42501</div>	<table><tr><th>id_dish [PK] integer</th><th>name character varying (180)</th><th>price integer</th><th>weight integer</th><th>proteins numeric (3,1)</th><th>fats numeric (3,1)</th><th>carbohydrates numeric (3,1)</th><th>calories integer</th></tr><tr><td>1</td><td>Сибирский бургер</td><td>112</td><td>241</td><td>5.0</td><td>26.0</td><td>12.0</td><td>180</td></tr><tr><td>2</td><td>Чиабургер</td><td>268</td><td>283</td><td>12.0</td><td>10.0</td><td>16.0</td><td>750</td></tr><tr><td>3</td><td>Бер Бургер</td><td>162</td><td>267</td><td>22.0</td><td>18.0</td><td>26.0</td><td>398</td></tr></table>	id_dish [PK] integer	name character varying (180)	price integer	weight integer	proteins numeric (3,1)	fats numeric (3,1)	carbohydrates numeric (3,1)	calories integer	1	Сибирский бургер	112	241	5.0	26.0	12.0	180	2	Чиабургер	268	283	12.0	10.0	16.0	750	3	Бер Бургер	162	267	22.0	18.0	26.0
id_dish [PK] integer	name character varying (180)	price integer	weight integer	proteins numeric (3,1)	fats numeric (3,1)	carbohydrates numeric (3,1)	calories integer																										
1	Сибирский бургер	112	241	5.0	26.0	12.0	180																										
2	Чиабургер	268	283	12.0	10.0	16.0	750																										
3	Бер Бургер	162	267	22.0	18.0	26.0	398																										
21	Обновление данных list_of_dishes																																
	UPDATE list_of_dishes SET id_dish = 1 WHERE id_list_of_dishes = 1;																																
	<div>ERROR: нет доступа к таблице list_of_dishes</div> <div>ОШИБКА: нет доступа к таблице list_of_dishes SQL state: 42501</div>	<table><tr><th></th><th>id_list_of_dishes [PK] integer</th><th>id_dish integer</th><th>id_order integer</th></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td><td>12</td><td>1</td></tr><tr><td>3</td><td>3</td><td>48</td><td>1</td></tr></table>		id_list_of_dishes [PK] integer	id_dish integer	id_order integer	1	1	1	1	2	2	12	1	3	3	48	1															
	id_list_of_dishes [PK] integer	id_dish integer	id_order integer																														
1	1	1	1																														
2	2	12	1																														
3	3	48	1																														
22	Выборка всех записей из client																																
	SELECT * FROM client;																																
	<div>ERROR: нет доступа к таблице client</div> <div>ОШИБКА: нет доступа к таблице client SQL state: 42501</div>	<div>ERROR: нет доступа к таблице client</div> <div>ОШИБКА: нет доступа к таблице client SQL state: 42501</div>																															

Таблица 1. Таблица прав пользователей

## 5 Создание функции

### 5.1 Формулировка задания

Для фамилии, имени и отчества, передаваемых в формате строк, вернуть строку в формате «И. О. Фамилия». Если отчества нет (пустая строка или значение NULL), то вернуть строку формата «И. Фамилия».

## 5.2 Реализация

```
1 CREATE FUNCTION get_initials_with_surname(surname varchar(30),
2                                           name varchar(30),
3                                           patronymic varchar(30))
4 RETURNS varchar(36) AS $$
5 BEGIN
6     IF patronymic IS NULL OR patronymic = '' THEN
7         RETURN substring(name for 1) || '. ' || surname;
8     ELSE
9         RETURN substring(name for 1) || '. ' || substring(patronymic for 1) ||
            ↳ '. ' || surname;
10    END IF;
11 END$$ LANGUAGE plpgsql;
```

Функция принимает три аргумента: surname (фамилия), name (имя) и patronymic (отчество), каждый из которых имеет тип `varchar(30)`.

Функция возвращает строку (`varchar(36)`) и работает следующим образом:

1. Проверяет, является ли отчество (patronymic) пустым или равным `NULL`.
2. Если отчество отсутствует или пустое, функция возвращает строку, состоящую из первой буквы имени (name), точки, пробела и фамилии (surname).
3. Если отчество присутствует, функция возвращает строку, состоящую из первой буквы имени, точки, пробела, первой буквы отчества, точки и фамилии.
4. Инициалы имени и отчества, а также фамилия объединяются в одну строку с помощью операции конкатенации.

Таким образом, функция формирует строку из инициалов и фамилии, в зависимости от наличия отчества.

## 5.3 Результат

Работу функции можно проверить выполнив следующий запрос, который для каждого клиента из таблицы `client` выводит его фамилию с инициалами и номер телефона.

```
1 SELECT
2     get_initials_with_surname(surname, name, patronymic) AS "ФИО",
3     phone_number AS "Тел. номер"
4 FROM client;
```

На Рис. 10 представлен результат выполнения этого запроса, использующего функцию.

	ФИО character varying	Тел. номер character
1	Д. Н. АБАБИЛОВ	+000000000000001
2	В. И. АБАБИЛОВ	+000000000000002
3	С. И. АБАБИЛОВ	+000000000000003
4	Ю. В. АБАБИЛОВ	+000000000000004
5	А. Н. АБАБИЛОВ	+000000000000005

Рис. 10. Таблица client с преобразованными фамилией, именем и отчеством

Для строк, в которых у клиента отсутствует отчество, следующий запрос так же будет корректно работать. Результат продемонстрирован на Рис. 11.

```

1 SELECT
2     get_initials_with_surname(surname, name, patronymic) AS "ФИО",
3     phone_number AS "Тел. номер"
4 FROM client
5 WHERE client.patronymic = '';

```

ФИО character varying	Тел. номер character
И. ИВАНОВ	+111111111111111

Рис. 11. Строки таблицы client с преобразованными фамилией, именем и отсутствующим отчеством

Неправильное применение функции, например, при попытке вызвать функцию используя другую таблицу, то есть с некорректными для функции данными, приведет к ошибке. Можно попробовать выполнить следующий запрос.

```

1 SELECT
2     get_initials_with_surname(surname, name, patronymic) AS "ФИО"
3 FROM dish;

```

Однако запрос вызовет ошибку, которая продемонстрирована на Рис. 12.

```

ERROR:  столбец "surname" не существует
LINE 2:  get_initials_with_surname(surname, name, patronymic) AS "ФИО...
      ^

```

Рис. 12. Ошибка при некорректном вызове функции get\_initials\_with\_surname()

## 6 Создание процедуры

### 6.1 Формулировка задания

Написать запрос, который выполняет проверку наличия данных в нескольких таблицах на основе переданных параметров:

- Наименование ресторана, адрес и номер свидетельства о регистрации должны соответствовать записи в таблице `restaurant`.
- Название ингредиента должно присутствовать в таблице `ingredient`.
- Количество указанного ингредиента на складе в указанном ресторане в таблице `ingredient_in_storage` должно принять новое передаваемое значение.

При отсутствии ресторана с данными параметрами, добавить запись в таблицу `restaurant`, при отсутствии ингредиента с данным названием, добавить запись в соответствующую таблицу. При отсутствии на складе строки с указанными рестораном и ингредиентом, добавить запись в таблицу `ingredient_in_storage`.

## 6.2 Реализация

```

1 CREATE OR REPLACE PROCEDURE upsert_restaurant_ingredient(
2     restaurant_name TEXT,
3     restaurant_address TEXT,
4     restaurant_registration_info TEXT,
5     ingredient_name TEXT,
6     ingredient_amount INT
7 )
8 LANGUAGE plpgsql
9 AS $$
10 DECLARE
11     restaurant_id INT;
12     ingredient_id INT;
13     storage_id INT;
14 BEGIN
15     -- Get or insert restaurant
16     SELECT id_restaurant INTO restaurant_id
17     FROM restaurant
18     WHERE name = restaurant_name AND address = restaurant_address AND registration_info
19     ↪ = restaurant_registration_info;
20
21     IF restaurant_id IS NULL THEN
22         LOCK TABLE restaurant IN EXCLUSIVE MODE;
23         SELECT max(id_restaurant) + 1 INTO restaurant_id FROM restaurant;
24         INSERT INTO restaurant (id_restaurant, name, address, registration_info, space,
25     ↪ capacity)
26         VALUES (restaurant_id, restaurant_name, restaurant_address,
27     ↪ restaurant_registration_info, DEFAULT, DEFAULT);
28     END IF;
29
30     -- Get or insert ingredient
31     SELECT id_ingredient INTO ingredient_id
32     FROM ingredient
33     WHERE name = ingredient_name;
34
35     IF ingredient_id IS NULL THEN

```



```

33         LOCK TABLE ingredient IN EXCLUSIVE MODE;
34         SELECT max(id_ingredient) + 1 INTO ingredient_id FROM ingredient;
35         INSERT INTO ingredient (id_ingredient, name)
36         VALUES (ingredient_id, ingredient_name);
37     END IF;
38
39     -- Get or insert/update ingredient_in_storage
40     SELECT id_ingredient_in_storage INTO storage_id
41     FROM ingredient_in_storage
42     WHERE id_ingredient = ingredient_id AND id_restaurant = restaurant_id;
43
44     IF storage_id IS NULL THEN
45         LOCK TABLE ingredient_in_storage IN EXCLUSIVE MODE;
46         SELECT max(id_ingredient_in_storage) + 1 INTO storage_id FROM
47             ↪ ingredient_in_storage;
48         INSERT INTO ingredient_in_storage (id_ingredient_in_storage, amount,
49             ↪ id_restaurant, id_ingredient)
50         VALUES (storage_id, ingredient_amount, restaurant_id, ingredient_id);
51     ELSE
52         UPDATE ingredient_in_storage
53         SET amount = ingredient_amount
54         WHERE id_ingredient_in_storage = storage_id;
55     END IF;
56 END;
57 $$;

```

Процедура `upsert_restaurant_ingredient()` выполняет операцию «UPSERT» (вставка или обновление) для данных о ресторанах, ингредиентах и их наличии на складе. Она принимает параметры: `restaurant_name` - имя ресторана, `restaurant_address` - адрес ресторана, `restaurant_registration_info` - регистрационную информацию, `ingredient_name` - название ингредиента, `ingredient_amount` - количество ингредиента на складе. `ingredient_amount` имеет тип `INT`, остальные тип `TEXT`.

Сначала проверяет наличие ресторана в таблице `restaurant`: если нет, добавляет новую запись с автоинкрементом ID. Потом проверяет наличие ингредиента в таблице `ingredient`: если его нет, создает новый. Наконец, проверяет таблицу `ingredient_in_storage`: если записи о данном ингредиенте в этом ресторане нет, вставляется новая с количеством, иначе количество обновляется.

Так как столбец `id_client` использует тип данных `integer`, а в PostgreSQL не поддерживается автоматическая инкрементация данного типа, то при вставке строк используется встроенная функция `max()`, которая находит максимальный id в таблице клиент и записывает результат функции плюс один записывается в переменную. Переменная является новым id и используется для последующей вставки строки в таблицу. Однако может возникнуть ситуация, когда между поиском id и вставкой новой строки может вклиниться другая транзакция, выполняющая ту же процедуру, и добавить новую строку, так что найденный id не будет соответствовать ожидаемому, и процедура сработает неправильно или вызовет ошибку. Для обеспечения целостности данных и предотвращения конкурентного изменения таблиц, процедура использует блокировки таблиц: таблицы `restaurant`, `ingredient`, `ingredient_in_storage` блокируется в режиме `EXCLUSIVE MODE` перед проверкой и вставкой строки.

## 6.3 Результат

Данную процедуру можно использовать для быстрой вставки ресторана, ингредиента, а также обновления количества заданного ингредиента в указанном ресторане.

### 1 вариант использования

Так, следующий запрос добавит новый ресторан «Пельменная» с указанными характеристиками в таблицу `restaurant` (Рис. 13), новый ингредиент «Фарш свинина-говядина» в таблицу `ingredient` (Рис. 14), а так же создаст запись для данного ресторана и ингредиента с количеством 100 в таблице `ingredient_in_storage` (Рис. 15).

```
1 CALL upsert_restaurant_ingredient(  
2     'Пельменная',  
3     'Самара, ул. Ленина, д. 10',  
4     'ИНН 0123456789',  
5     'Фарш свинина-говядина',  
6     100  
7 );
```

id_restaurant [PK] integer	name character varying (260)	registration_info text	address character varying (200)	space integer	capacity integer
52	Пельменная	ИНН 0123456789	Самара, ул. Ленина, д. 10	100	100
51	Вкусная-закусная	ИНН 0123456789	Санкт-Петербург, улица Марата, д. 10	100	100
50	Бургерная у Пролетариата	ИНН 0123456789	Владикавказ, пр. Октябрьский, д. 17	139	115
49	Французский Бистро	ИНН 9012345678	Чита, ул. Гурманов, д. 35	129	83
48	Тортилья Таун	ИНН 8901234567	Белгород, ул. Мексикаская, д. 21	171	81

Рис. 13. Добавленная запись в таблицу `restaurant`

id_ingredient [PK] integer	name character varying (50)
61	Фарш свинина-говядина
60	Сливки
59	Спаржа
58	Хумус
57	Авокадо

Рис. 14. Добавленная запись в таблицу `ingredient`

id_ingredient_in_storage [PK] integer	amount integer	id_restaurant integer	id_ingredient integer
3002	100	52	61
3001	30	51	1
3000	200	50	60
2999	200	50	59
2998	200	50	58
2997	200	50	57

Рис. 15. Добавленная запись в таблицу `ingredient_in_storage`

## 2 вариант использования

Следующий запрос добавит запись в таблицу `restaurant` (Рис. 16), а так же строку в таблицу `ingredient_in_storage` (Рис. 17), таблица `ingredient` останется прежней, так как ингредиент «Фарш свинина-говядина» уже присутствует в ней.

```

1 CALL upsert_restaurant_ingredient(
2     'Закусочная Весла',
3     'Краснодар, ул. Восточная, д. 21',
4     'ИНН 1234567890',
5     'Фарш свинина-говядина',
6     100
7 );

```

id_restaurant [PK] integer	name character varying (260)	registration_info text	address character varying (200)	space integer	capacity integer
53	Закусочная Весла	ИНН 1234567890	Краснодар, ул. Восточная, д. 21	100	100
52	Пельменная	ИНН 0123456789	Самара, ул. Ленина, д. 10	100	100
51	Вкусная-закусная	ИНН 0123456789	Санкт-Петербург, улица Марата, д. 10	100	100
50	Бургерная у Пролетариата	ИНН 0123456789	Владикавказ, пр. Октябрьский, д. 17	139	115
49	Французский Бистро	ИНН 9012345678	Чита, ул. Гурманов, д. 35	129	83

Рис. 16. Добавленная запись в таблицу `restaurant`

id_ingredient_in_storage [PK] integer	amount integer	id_restaurant integer	id_ingredient integer
3003	100	53	61
3002	100	52	61
3001	30	51	1
3000	200	50	60
2999	200	50	59

Рис. 17. Добавленная запись в таблицу `ingredient_in_storage`

## 3 вариант использования

Следующий запрос добавит запись в таблицу `ingredient` (Рис. 18), а так же строку в таблицу `ingredient_in_storage` (Рис. 19), таблица `restaurant` останется прежней,

так как ресторан «Закусочная Весла» с регистрацией «ИНН 1234567890» по адресу «Краснодар, ул. Восточная, д. 21» уже присутствует в ней.

```

1 CALL upsert_restaurant_ingredient(
2     'Закусочная Весла',
3     'Краснодар, ул. Восточная, д. 21',
4     'ИНН 1234567890',
5     'Желатин сухой пищевой',
6     111
7 );

```

id_ingredient [PK] integer	name character varying (50)
62	Желатин сухой пищевой
61	Фарш свинина-говядина
60	Сливки
59	Спаржа
58	Хумус

Рис. 18. Добавленная запись в таблицу `ingredient`

id_ingredient_in_storage [PK] integer	amount integer	id_restaurant integer	id_ingredient integer
3004	111	53	62
3003	100	53	61
3002	100	52	61
3001	30	51	1
3000	200	50	60

Рис. 19. Добавленная запись в таблицу `ingredient_in_storage`

## 4 вариант использования

Следующий запрос обновит количество ингредиентов в таблице `ingredient_in_storage` (Рис. 20), для указанных ингредиента и ресторана.

```

1 CALL upsert_restaurant_ingredient(
2     'Закусочная Весла',
3     'Краснодар, ул. Восточная, д. 21',
4     'ИНН 1234567890',
5     'Желатин сухой пищевой',
6     222
7 );

```

id_ingredient_in_storage [PK] integer	amount integer	id_restaurant integer	id_ingredient integer
3004	222	53	62
3003	100	53	61
3002	100	52	61
3001	30	51	1
3000	200	50	60

Рис. 20. Обновленная запись в таблице `ingredient_in_storage`

## 7 Использование транзакций

### 7.1 Формулировка задания

Необходимо задать уровень изоляции транзакций `READ COMMITTED` и выполнить проверку неповторяющегося чтения.

## 7.2 Реализация и результат

№	Транзакция 1	Транзакция 2
$t_1$	Установка уровня изоляции и начало транзакции 1	
	BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;  BEGIN	
$t_2$	Установка уровня изоляции и начало транзакции 2	
		BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;  BEGIN
$t_3$	Чтение данных из таблицы client	
	SELECT * FROM client WHERE name = 'ИРИНА' AND surname = 'АБАЛКИНА';  <div> <div>id_client</div> <div>[PK] integer</div> <div>418</div> </div> <div> <div>surname</div> <div>character varying (30)</div> <div>АБАЛКИНА</div> </div> <div> <div>name</div> <div>character varying (30)</div> <div>ИРИНА</div> </div> <div> <div>patronymic</div> <div>character varying (30)</div> <div>ВИКТОРОВНА</div> </div> <div> <div>phone_number</div> <div>character</div> <div>+000000000000418</div> </div>	
$t_4$	Изменение отчества (patronymic) клиента в таблице client	
		UPDATE client  SET patronymic = 'АРТЕМОВНА' WHERE name = 'ИРИНА' AND surname = 'АБАЛКИНА';  UPDATE 1

№	Первая транзакция	Вторая транзакция														
t <sub>5</sub>	Сохранение результата															
		COMMIT;  COMMIT														
t <sub>6</sub>	Проверка изменения данных															
		SELECT * FROM client  WHERE name = 'ИРИНА'  AND surname = 'АБАЛКИНА'; <table><tr><td>id_client</td><td>surname</td><td>name</td><td>patronymic</td><td>phone_number</td></tr><tr><td>[PK] integer</td><td>character varying (30)</td><td>character varying (30)</td><td>character varying (30)</td><td>character</td></tr><tr><td>418</td><td>АБАЛКИНА</td><td>ИРИНА</td><td>АРТЕМОВНА</td><td>+000000000000418</td></tr></table>	id_client	surname	name	patronymic	phone_number	[PK] integer	character varying (30)	character varying (30)	character varying (30)	character	418	АБАЛКИНА	ИРИНА	АРТЕМОВНА
id_client	surname	name	patronymic	phone_number												
[PK] integer	character varying (30)	character varying (30)	character varying (30)	character												
418	АБАЛКИНА	ИРИНА	АРТЕМОВНА	+000000000000418												
t <sub>7</sub>	Чтение данных из таблицы client															
	SELECT * FROM client  WHERE name = 'ИРИНА'  AND surname = 'АБАЛКИНА'; <table><tr><td>id_client</td><td>surname</td><td>name</td><td>patronymic</td><td>phone_number</td></tr><tr><td>[PK] integer</td><td>character varying (30)</td><td>character varying (30)</td><td>character varying (30)</td><td>character</td></tr><tr><td>418</td><td>АБАЛКИНА</td><td>ИРИНА</td><td>АРТЕМОВНА</td><td>+000000000000418</td></tr></table>	id_client	surname	name	patronymic	phone_number	[PK] integer	character varying (30)	character varying (30)	character varying (30)	character	418	АБАЛКИНА	ИРИНА	АРТЕМОВНА	+000000000000418
id_client	surname	name	patronymic	phone_number												
[PK] integer	character varying (30)	character varying (30)	character varying (30)	character												
418	АБАЛКИНА	ИРИНА	АРТЕМОВНА	+000000000000418												

В момент времени  $t_3$  в первой транзакции при чтении строки в таблице `client` пользователь видит строку с `id_client = 418` где отчество `patronymic = «ВИКТОРОВНА»`. В момент времени  $t_7$  в первой транзакции при выполнении того же запроса на чтение пользователь ожидает увидеть ту же самую строку с тем же отчеством, однако видит `patronymic = «АРТЕМОВНА»`. Это связано с тем, что во второй транзакции в момент  $t_4$  данная строка была изменена и на моменте  $t_5$  был сделан `COMMIT`. Таким образом, при уровне изоляции `READ COMMITTED` неповторяющееся чтение выполняется.

# Заключение

В ходе выполнения поставленных задач было сделано следующее:

- Было создано представление (view), которое рассчитывает количество блюд, в которых используется каждый ингредиент, а также общее число заказов этих блюд. Составлен запрос, который использует представление. Показано, что в представлении невозможно добавить или удалить данные.
- Реализованы триггеры, которые автоматически обновляют количество клиентов для каждого ресторана. Создана новая таблица, которая отражает изменения данных при добавлении или удалении клиентов.
- Созданы два пользователя с различными уровнями доступа. Первый пользователь имеет права только на просмотр созданного представления, в то время как второй может не только просматривать, но и вносить изменения в исходные таблицы, что показало возможность детального управления правами доступа. Была сделана таблица для демонстрации и сравнения прав доступа двух пользователей для различных операций.
- Написана функция, которая принимает на вход фамилию, имя и отчество, и форматирует их в заданном виде. В случае отсутствия отчества функция корректно обрабатывает данные, что упрощает работу с ФИО в различных форматах. Также продемонстрирован вариант неверного вызова функции.
- Разработана процедура, проводящая проверку наличия данных в нескольких связанных таблицах. При отсутствии данных, соответствующие таблицы дополняются новыми строками.
- Реализовано управление транзакциями с уровнем изоляции `READ COMMITTED`, проведена проверка неповторяющегося чтения. Изучение этого уровня изоляции подтвердило что защита от неповторяющегося чтения не выполняется при параллельных транзакциях.

Итого было сделано 5 лабораторных работ, в ходе которых было создано 1 представление, 2 триггера, 2 пользователя с различными привелегиями, 1 функция, 1 процедура и было проведено параллельное выполнение 2 транзакций при режиме изоляции `READ COMMITTED`. Таким образом, лабораторные работы способствовали развитию навыков работы с SQL и базами данных, а также позволили глубже понять механизмы обеспечения целостности данных и управления доступом.



## Список литературы

- [1] PostgreSQL: Documentation : сайт. – URL: <https://www.postgresql.org/docs/> (дата обращения: 17.05.2024)