



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА □ Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)

Кафедра цифровой трансформации (ЦТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Разработка баз данных»

Практическое занятие №8

Студенты группы

ИКБО-65-23 Олефиров Г.Г.

(подпись)

Ассистент

Морозов Д.В.

(подпись)

Отчет представлен

«___»_____ 2025 г.

Москва 2025 г.

ПРАКТИЧЕСКАЯ РАБОТА №8. СЛОЖНЫЕ ТИПЫ ДАННЫХ И МАСШТАБИРОВАНИЕ POSTGRESQL

Постановка задачи:

Целью данной практической работы является освоение методов работы со сложными и неструктурированными типами данных (JSONB, BYTEA) в PostgreSQL, а также получение практических навыков реализации горизонтального масштабирования базы данных с помощью декларативного секционирования.

Задание №1: хранение паролей (BYTEA)

1. Создать справочную таблицу ролей и таблицу системных пользователей (или модифицировать существующую).
2. В таблице пользователей предусмотреть поля для хранения «сырого» пароля (только для учебной демонстрации) и его хеша (тип BYTEA).
3. Сохранить хеши паролей пользователей, используя функцию хеширования digest.
4. Выполнить запрос, проверяющий соответствие введенного пароля сохраненному хешу. Одна проверка должна быть успешной, другая - нет (допустимо сделать это в одном запросе).**1. Аудит и логирование.**

Запись факта изменения данных.

Пример: при любом изменении столбца salary в таблице employees создавать запись в salary_log, сохраняя OLD.salary, NEW.salary и CURRENT_USER.

Задание №2: секционирование таблицы логов

1. Создать новую секционированную таблицу для хранения логов событий.
2. Использовать стратегию секционирования по диапазонам (RANGE) по полю даты.
3. Создать две секции для периодов:
 - 2-е полугодие 2024 (с 2024-07-01 по 2025-01-01).
 - 1-е полугодие 2025 (с 2025-01-01 по 2025-07-01).
4. Создать секцию по умолчанию, куда будут попадать все данные, выходящие за рамки указанных диапазонов (будущее время).

Задание №3: генерация данных

Написать скрипт на PL/pgSQL для генерации 20 000 записей в диапазоне дат, покрывающем созданные секции. Сгенерированные данные должны содержать JSON-строку с различным набором полей для разных типов событий (например, чтобы логи «продажи» отличались по структуре от логов «входа»).

Вывести сгенерированные данные – показать запросом SELECT, что данные успешно созданы и физически распределились по разным таблицам секциям.

Задание №4: анализ и поиск по JSONB

Написать три запроса:

1. Фильтрация по значению – найти записи, где числовое поле внутри JSON удовлетворяет математическому условию (например, > 800), используя операторы извлечения $->$ и $->>$.
2. Поиск по вхождению – найти записи, содержащие точный фрагмент JSON-структуры (например, `{"quantity": 5}`), используя оператор `@>`.
3. Агрегация – посчитать сумму или среднее значение числового поля из JSON-документа.

Задание №5: модификация данных JSONB

Продемонстрировать изменение данных внутри JSON-поля:

1. Массовое добавление нового поля во все записи определенного типа.
2. Точечное изменение значения по ключу в конкретной записи.

1. Работа с бинарными данными(BYTEA) и хеширование

1.1 Создание таблиц

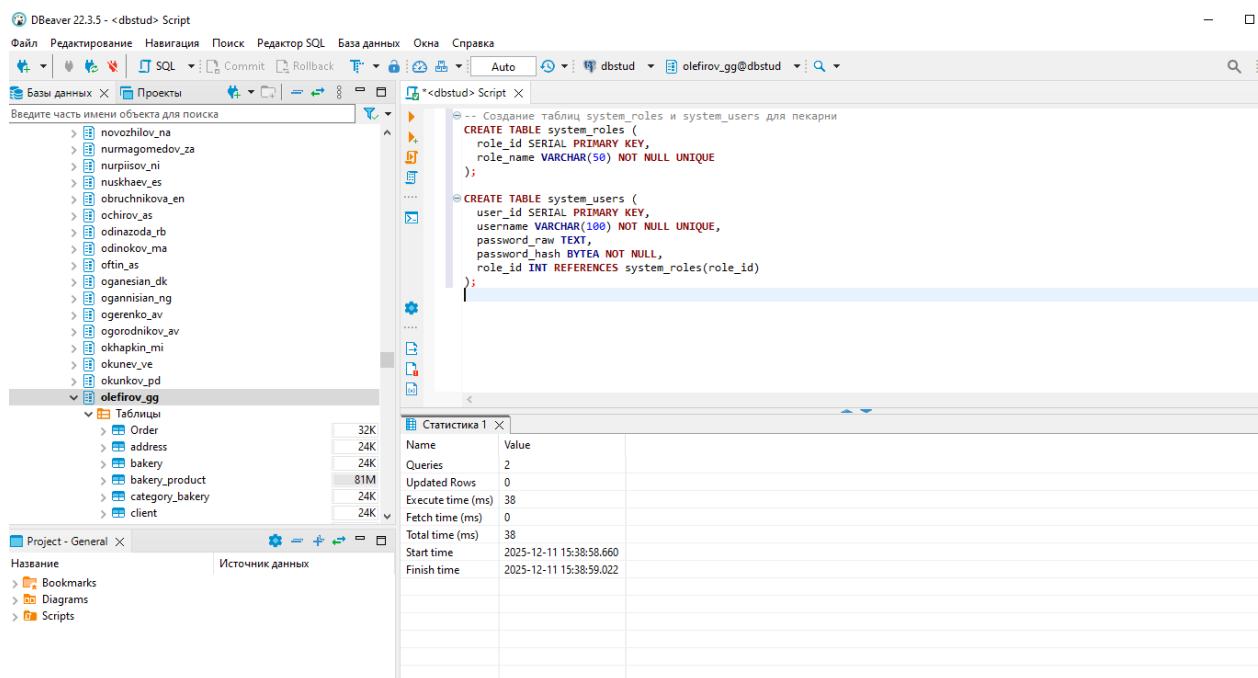


Рисунок 1 – Создание таблиц

DBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Помощь Редактор SQL База данных Окна Справка

Базы данных Проекты

Ведите часть имени объекта для поиска

```
-- Заполнение ролей для пекарни
INSERT INTO system_roles (role_name) VALUES
('Администратор'),
('Пекарь'),
('Кассир'),
('Курьер'),
('Менеджер');

-- Проверка результата
SELECT * FROM system_roles ORDER BY role_id;
```

system_roles 1 Статистика 1

role_id	role_name
1	Администратор
2	Пекарь
3	Кассир
4	Курьер
5	Менеджер

Значение

Select a cell to view/edit value
Press F7 to hide this panel

Обновить Save Cancel Экспорт данных ... 200 5 : 5 строк получено - 18ms, 2025-12-11 в 15:39:40

Инт.ставка 11:1:225 Sel: 0 | 0

Рисунок 2 – Заполнение ролей

DBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Помощь Редактор SQL База данных Окна Справка

Базы данных Проекты

Ведите часть имени объекта для поиска

```
-- Вставка пользователей пекарни с хешами
INSERT INTO system_users (username, password_raw, password_hash, role_id) VALUES
('bakery_admin', 'Bakery2025!', digest('Bakery2025!', 'sha256'), 1),
('baker_ivan', 'Pechka123', digest('Pechka123', 'sha256'), 2),
('cashier_marina', 'Kassa2025', digest('Kassa2025', 'sha256'), 3);

-- Показать вставленные данные
SELECT username, password_raw, encode(password_hash, 'hex') AS password_hash_hex
FROM system_users ORDER BY user_id;
```

system_users 1 Статистика 1

username	password_raw	password_hash_hex
bakery_admin	Bakery2025!	2b25fd5a9c137303b3e104d26a082e7fe3945cd67f89cf2f6f9297c885975b
baker_ivan	Pechka123	eb21c297e8ef04867b8fffea577c6e092ef2d97e580f76c228c97276fd20d0
cashier_marina	Kassa2025	08a022dc5c9b2d6a4e0d1d6ee6700af0049154d6aca7e62cac62b145f5fb98d

Значение

Select a cell to view/edit value
Press F7 to hide this panel

Обновить Save Cancel Экспорт данных ... 200 3 : 3 строк получено

Инт.ставка 10:1:485 Sel: 0 | 0

Рисунок 3 – Хеширование паролей и вставка пользователей

1.2 Проверка паролей

DBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

Базы данных Проекты

Ведите часть имени объекта для поиска

```
-- Проверка паролей пользователей пекарни
SELECT
    username,
    password_raw,
    encode(password_hash, 'hex') AS stored_hash,
    encode(digest(password_raw, 'sha256'), 'hex') AS input_hash,
    (digest(password_raw, 'sha256') + password_hash) AS is_valid
FROM system_users
ORDER BY user_id;
```

system_users 1

	abc username	abc password_raw	abc stored_hash	abc input.hash	is_	Значение
1	baker_admin	Bakery2025!	2b25fd5a9c137303b3e104d26a0828e;			
2	baker_ivan	Pechka123	eb21c297e8ef904a867b8fe577ce0e0;			
3	cashier_maria	Kassa2025	0ba022dc5c9b2d64a40d1de66700af1f6201778828dc3a48df859072380a94			

Таблица

Этапы

Обновить Save Cancel Экспорт данных ... 200 3 строки получено

Рисунок 4 – Проверка соответствия паролей

2.Масштабирование базы данных(секционирование)

2.1 Создание таблицы логов

DBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

Базы данных Проекты

Ведите часть имени объекта для поиска

```
-- Создание секционированной таблицы логов заказов пекарни
CREATE table if not EXISTS bakery_logs (
    log_id BIGSERIAL,
    created_at TIMESTAMPTZ NOT NULL,
    employee_id BIGINT,
    order_id BIGINT,
    log_data JSONB
) PARTITION BY RANGE (created_at);
```

Выход

relation "bakery_logs" already exists, skipping

Статистика 1

Name	Value
Updated Rows	0
Query	-- Создание секционированной таблицы логов заказов пекарни CREATE table if not EXISTS bakery_logs (log_id BIGSERIAL, created_at TIMESTAMPTZ NOT NULL, employee_id BIGINT, order_id BIGINT, log_data JSONB) PARTITION BY RANGE (created_at)
Start time	Thu Dec 11 15:42:55 MSK 2025
Finish time	Thu Dec 11 15:42:55 MSK 2025

MSK | ru | Запись | Инт. вставка | 2 : 27 : 86 | Sel: 0 | 0

Рисунок 5 – Создание таблицы логов

2.2 Создание секций(партиций)

DBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

Базы данных Проекты

Ведите часть имени объекта для поиска

CREATE TABLE bakery_logs_2024_h2 PARTITION OF bakery_logs FOR VALUES FROM ('2024-07-01') TO ('2025-01-01');

CREATE TABLE bakery_logs_2025_h1 PARTITION OF bakery_logs FOR VALUES FROM ('2025-01-01') TO ('2025-07-01');

CREATE TABLE bakery_logs_default PARTITION OF bakery_logs DEFAULT;

-- Показать все партиции

SELECT schemaname, tablename, tableowner FROM pg_tables WHERE tablename LIKE 'bakery_logs%' ORDER BY tablename;

pg_tables 1 Статистика 1

название	схема	имя таблицы	владелец
1	olefirov_gg	bakery_logs	olefirov_gg
2	olefirov_gg	bakery_logs_2024_h2	olefirov_gg
3	olefirov_gg	bakery_logs_2025_h1	olefirov_gg
4	olefirov_gg	bakery_logs_default	olefirov_gg

Значение

Select a cell to view/edit value
Press F7 to hide this panel

Обновить Save Cancel Экспорт данных ... 200 4

4 строки получено - 50ms, 2025-12-11 в 15:43:39

MSK ru Запись Инт.ставка 15 : 1 : 479 Sel: 0 | 0

Рисунок 6 – Создание секций

DBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

Базы данных Проекты

Ведите часть имени объекта для поиска

-- Создание индексов для логов пекарни

CREATE INDEX idx_bakery_logs_date ON bakery_logs(created_at);
CREATE INDEX idx_bakery_logs_employee ON bakery_logs(employee_id);
CREATE INDEX idx_bakery_logs_order ON bakery_logs(order_id);

-- Показать индексы

SELECT indexname, indexdef FROM pg_indexes WHERE tablename LIKE 'bakery_logs' OR tablename LIKE 'bakery_logs_%' ORDER BY indexname;

pg_indexes 1 Статистика 1

название	индекс
1	bakery_logs_2024_h2_created_at_idx
2	bakery_logs_2024_h2_employee_id_idx
3	bakery_logs_2024_h2_order_id_idx
4	bakery_logs_2025_h1_created_at_idx
5	bakery_logs_2025_h1_employee_id_idx
6	bakery_logs_2025_h1_order_id_idx
7	bakery_logs_default_created_at_idx
8	bakery_logs_default_employee_id_idx
9	bakery_logs_default_order_id_idx
10	idx_bakery_logs_date
11	idx_bakery_logs_employee
12	idx_bakery_logs_order

Значение

I to view
to hide t

Обновить Save Cancel Экспорт данных ... 200 12

12 строк получено

MSK ru Запись Инт.ставка 11 : 1 : 394 Sel: 0 | 0

Рисунок 7 – Создание индексов

3. Генерация данных и тип JSONB

3.1 Генерация тестовых данных

```

DECLARE
    random_date TIMESTAMP;
    random_random INT;
    event_type TEXT;
    json_payload JSONB;
    random_order INT;
    random_product INT;
BEGIN
    FOR i IN 1..20000 LOOP
        -- Генерация даты с середине 2024 по середину 2025
        random_date := '2024-07-01'::TIMESTAMP + random() * ('2025-07-01'::TIMESTAMP - '2024-07-01'::TIMESTAMP);
        random_employee := floor(random() * 5 + 1)::INT; -- ID сотрудников 1-5
        random_order := floor(random() * 5 + 1)::INT; -- ID заказов 1-5
        random_product := floor(random() * 100 + 1);

        IF (i % 3 = 0) THEN
            event_type := 'печь_продукт';
            json_payload := jsonb_build_object(
                'event', event_type,
                'product_id', floor(random() * 8 + 1)::INT, -- ID продуктов 1-8
                'quantity', floor(random() * 100 + 1),
                'batch_price', (random() * 5000 + 1000)::NUMERIC(10,2)
            );
        ELSIF (i % 3 = 1) THEN
            event_type := 'принять_заказ';
            json_payload := jsonb_build_object(
                'event', event_type,
                'order_id', random_order,
                'client_id', floor(random() * 5 + 1)::INT,
                'total_amount', (random() * 5000 + 500)::NUMERIC(10,2)
            );
        ELSE
            event_type := 'доставка';
            json_payload := jsonb_build_object(
                'event', event_type,
                'order_id', random_order,
                'employee_id', random_employee,
                'delivery_time', floor(random() * 120 + 15) -- минуты 15-135
            );
        END IF;

        INSERT INTO bakery_logs (created_at, employee_id, order_id, log_data)
        VALUES (random_date, random_employee, random_order, json_payload);
    END LOOP;
    -- Проверка общего количества
    SELECT COUNT(*) as total_bakery_logs FROM bakery_logs;
END $$;

```

Результат 1 | Статистика 1

	123 total_bakery_logs
1	20 005

Рисунок 8 – Скрипт генерации данных

3.2 Проверка распределения данных

```

-- Проверка распределения логов пекарни по секциям
SELECT
    '2024_h2' as partition_name,
    COUNT(*) as records_count
FROM ONLY bakery_logs_2024_h2
UNION ALL
SELECT
    '2025_h1',
    COUNT(*)
FROM ONLY bakery_logs_2025_h1
UNION ALL
SELECT
    'default',
    COUNT(*)
FROM ONLY bakery_logs_default
ORDER BY partition_name;

```

Результат 1 | Статистика 1

	partition_name	records_count
1	2024_h2	10 059
2	2025_h1	9 945
3	default	1

Рисунок 9 – Результат распределения данных по секциям

4. Анализ данных JSONB

4.1 Поиск по ключу и фильтрация с условием

The screenshot shows the DBeaver interface with a database connection named 'olefirov_gg@dbstud'. In the top navigation bar, the 'SQL' tab is selected. A script editor window contains the following SQL query:

```
-- Продажа/пекарня с высокой стоимостью > 2000 руб
SELECT
    log_id,
    created_at,
    employee_id,
    log_data->>'event' AS event_type,
    (log_data->>'batch_price')::NUMERIC AS batch_price
FROM bakery_logs
WHERE (log_data->>'batch_price')::NUMERIC > 2000
    OR (log_data->>'total_amount')::NUMERIC > 2000
LIMIT 5;
```

The results of this query are displayed in a table titled 'bakery_logs 1'. The table has columns: log_id, created_at, employee_id, event_type, and batch_price. The data is as follows:

	log_id	created_at	employee_id	event_type	batch_price
1	8	2024-11-11 12:07:40.804 +0300	3	печь_продукт	4 610,63
2	9	2024-11-27 12:22:09.522 +0300	5	принять_заказ	[NULL]
3	11	2024-07-14 13:38:53,754 +0300	3	печь_продукт	5 621,55
4	14	2024-10-22 16:10:48.605 +0300	4	печь_продукт	3 384,4
5	15	2024-11-19 06:45:30.160 +0300	3	принять_заказ	[NULL]

Рисунок 10 – Результат выборки суммы > 2000

4.2 Фильтрация по вхождению (JSONB Containment)

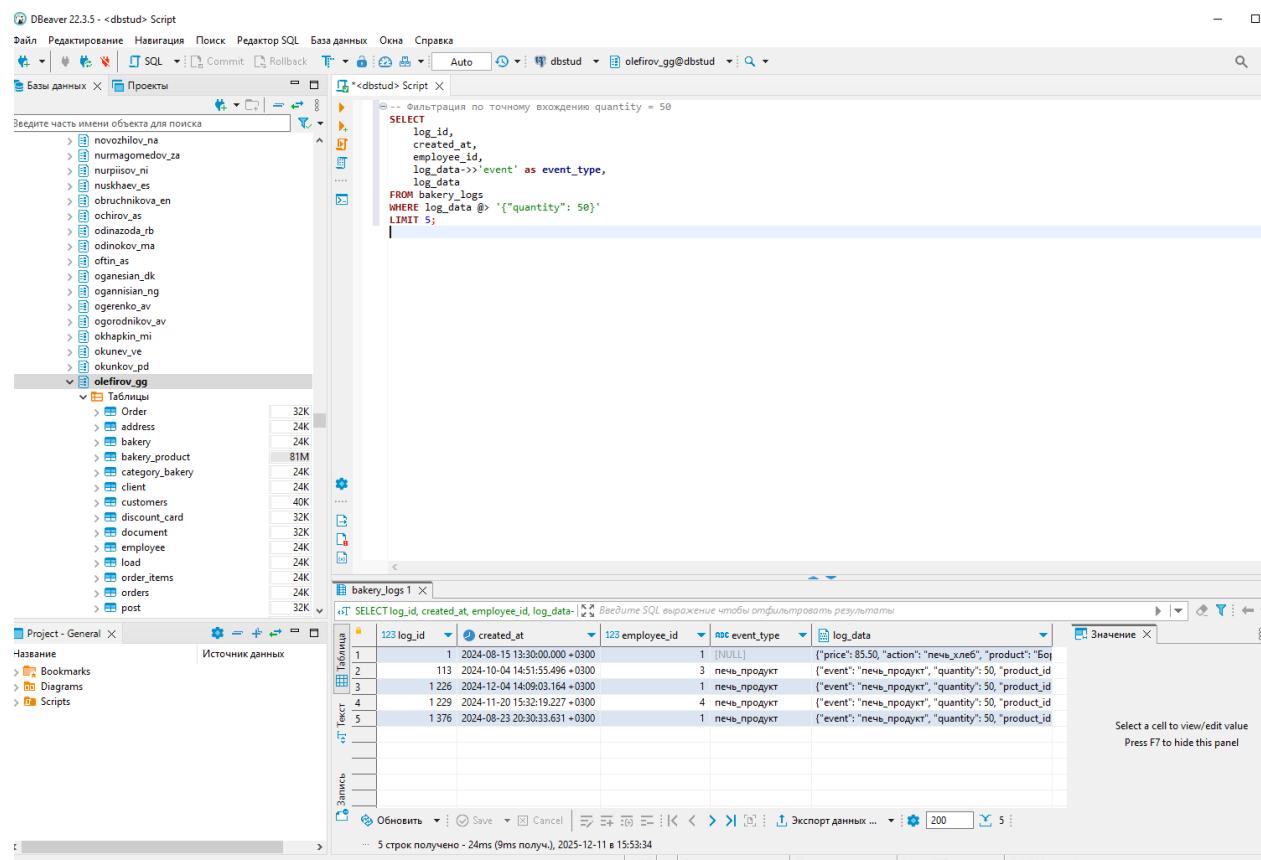


Рисунок 11 – Результаты поиска всех записей равный quantity = 50

4.3 Агрегация данных

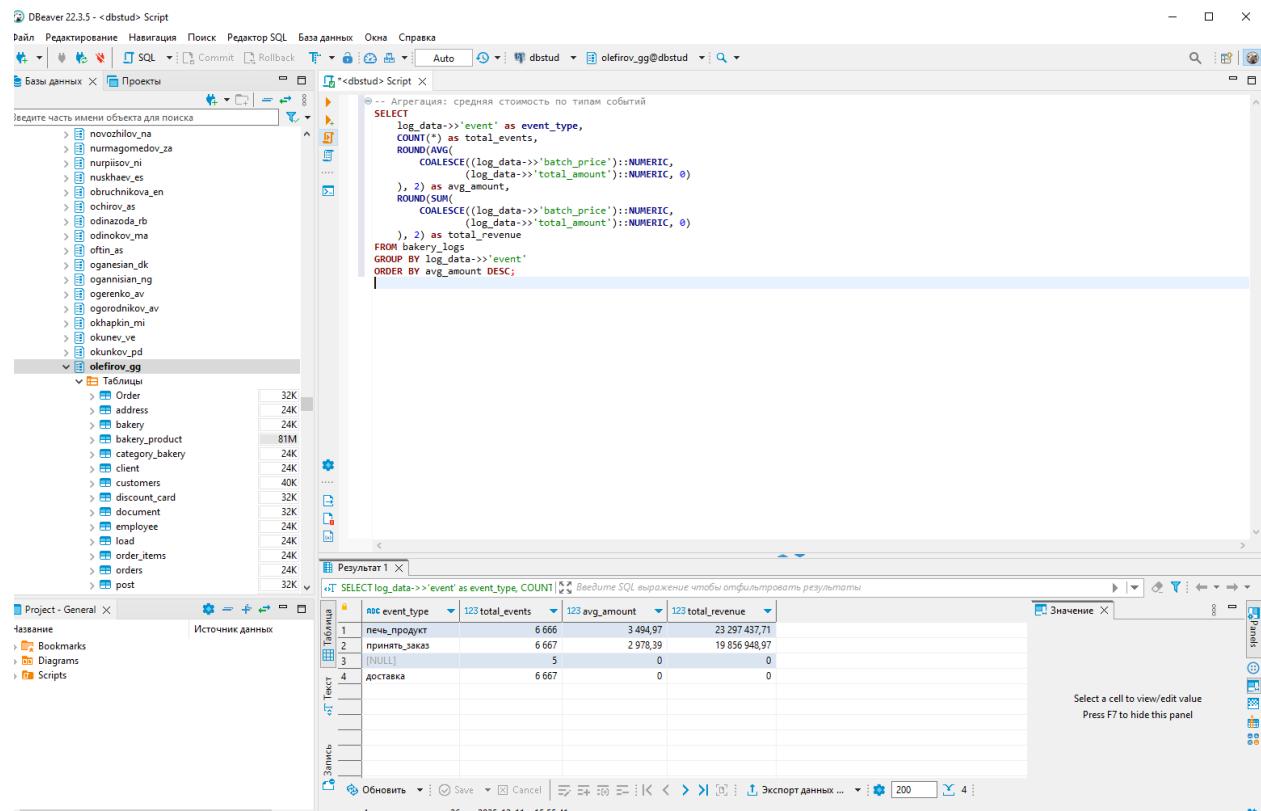


Рисунок 12 – Средняя стоимость операций пекарни

5. Модификация данных JSONB

5.1 Массовое обновление

The screenshot shows the DBeaver interface with a script editor and a results grid.

Script Editor:

```
-- Массовое обновление - добавление поля bakery_id
UPDATE bakery_logs
SET log_data +> log_data || jsonb_build_object('bakery_id', 1, 'source', 'bakery_system')
WHERE log_data ? 'event';

-- Проверка результата (первые 3 записи)
SELECT
    log_id,
    created_at,
    log_data->>'event' as event,
    log_data->>'bakery_id' as bakery_id,
    log_data->>'source' as source
FROM bakery_logs
ORDER BY log_id
LIMIT 3;
```

Results Grid:

log_id	created_at	event	bakery_id	source
1	2024-08-15 13:30:00.000 +0300	[NULL]	[NULL]	[NULL]
2	2024-09-20 17:20:00.000 +0300	[NULL]	[NULL]	[NULL]
3	2025-03-10 12:15:00.000 +0300	[NULL]	[NULL]	[NULL]

Рисунок 13 – Результат добавления поля bakery_id

5.2 Точечное изменение (функция jsonb_set)

DBBeaver 22.3.5 - <dbstud> Script

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

Базы данных Проекты

Задайте часть имени объекта для поиска

```

-- Точечное изменение статуса для первой записи пекарки
UPDATE bakery_logs
SET log_data = jsonb_set(
    log_data,
    '{status}',
    '"выполнен"'::jsonb
)
WHERE log_id = (SELECT MIN(log_id) FROM bakery_logs WHERE log_data ? 'order_id');

-- Проверка результата
SELECT
    log_id,
    created_at,
    log_data->>'event' AS event_type,
    log_data->>'order_id' AS order_id,
    log_data->>'status' AS status,
    log_data
FROM bakery_logs
WHERE log_data ? 'status'
ORDER BY log_id
LIMIT 1;

```

Таблицы

- Order
- address
- bakery
- bakery_product
- category_bakery
- client
- customers
- discount_card
- document
- employee
- load
- order_items
- orders
- post

Статистика 1

	log_id	created_at	event_type	order_id	status	log_data
1	123	2025-03-08 07:49:00.114 +0300	принять_заказ	3	выполнен	{"event": "принять_заказ", "source": "bakery_sy"}

Значение

elect a cell to view/edit value
Press F7 to hide this panel

Рисунок 14 – Результат изменения статуса