



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 №7
по дисциплине «Разработка баз данных»

Студент группы *ИКБО-50-23. Павлов Н.С..*

(подпись)

Преподаватель *Мажей Я. В.*

(подпись)

Москва 2025 г.

СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ	3
2 ВЫПОЛНЕНИЕ РАБОТЫ	5
2.1 ИСХОДНЫЕ ТАБЛИЦЫ	5
2.2 ПОДГОТОВКА БД	7
2.3 ЗАДАНИЕ 1. АНАЛИЗ И ОПТИМИЗАЦИЯ.....	8
2.3.1 Сценарий 1. B-Tree	8
2.3.2 Сценарий 2. Индекс по выражению	10
2.3.3 Сценарий 3. Частичный индекс	12
2.4 ЗАДАНИЕ 2. ДЕМОНСТРАЦИЯ АТОМАРНОСТИ. УСПЕШНЫЙ СОММІТ	14
2.5 ЗАДАНИЕ 3. ДЕМОНСТРАЦИЯ АТОМАРНОСТИ. АВАРИЙНЫЙ ROLLBACK	16
2.6 ЗАДАНИЕ 4. МОДЕЛИРОВАНИЕ АНОМАЛИИ «НЕПОВТОРЯЕМОЕ ЧТЕНИЕ»	18
2.7 ЗАДАНИЕ 5. УСТРАНЕНИЕ АНОМАЛИИ «НЕПОВТОРЯЕМОЕ ЧТЕНИЕ»	20

1. ПОСТАНОВКА ЗАДАЧИ

Цель: формирование у студентов практических навыков анализа и оптимизации производительности SQL-запросов, а также освоение механизмов управления транзакциями для обеспечения целостности данных (согласно принципам ACID) в СУБД PostgreSQL.

Задачи:

Задание №1: анализ и оптимизация (3 сценария)

Определить три различных «медленных» запроса к вашей БД, которые можно оптимизировать с помощью разных типов индексов (например, стандартный B-Tree, индекс по выражению, частичный/отфильтрованный индекс).

Если в вашей базе недостаточно данных, выполните для одной из своих таблиц действия по автоматической генерации содержимого, описанные в разделе «Подготовка базы данных».

Для каждого из 3-х сценариев:

1. Выполнить анализ запроса «КАК ЕСТЬ» (без индекса) с помощью EXPLAIN ANALYZE.
2. Привести план выполнения «ДО», письменно проанализировать его и выявить причину низкой производительности (например, Seq Scan).
3. Создать необходимый INDEX для оптимизации.
4. Повторно выполнить EXPLAIN ANALYZE.
5. Привести план выполнения «ПОСЛЕ», демонстрирующий использование индекса (например, Index Scan).
6. Обязательно сформировать сравнительную таблицу (см. Таблица 1), демонстрирующую разницу в производительности (план, Execution Time) «ДО» и «ПОСЛЕ».

Задание №2: демонстрация атомарной транзакции (COMMIT).

По примеру раздела 2.2 реализуйте в своей базе одну бизнес-операцию (минимум две связанные операции изменения данных) внутри транзакции BEGIN...COMMIT.

Задокументировать все шаги и результаты, сделать выводы.

Задание №3: демонстрация отката транзакции (ROLLBACK).

Адаптировать приведённый в разделе 2.3 SQL-скрипт, моделирующий сбой операции, под свою предметную область, повторив описанные действия.

Задокументировать все шаги и результаты, сделать выводы.

Задание №4: моделирование аномалии «Неповторяющее чтение».

Используя два редактора SQL, смоделировать проблему «неповторяющее чтение» на уровне изоляции по умолчанию (READ COMMITTED) по приведённому в разделе 2.4 образцу.

Задокументировать все шаги и результаты, сделать выводы.

Задание №5: устранение аномалии «Неповторяющее чтение».

Повторить моделирование из Задания №4, но с использованием уровня изоляции REPEATABLE READ. В качестве образца использовать раздел 2.5.

Задокументировать все шаги и результаты, сделать выводы.

2 ВЫПОЛНЕНИЕ РАБОТЫ

2.1 ИСХОДНЫЕ ТАБЛИЦЫ

application					
	123 ↗ application_id	123 ↘ client_id	⌚ application_date	🔤 A-Z problem_description	
1	1	1	2024-01-15	Прокол задней камеры, требуется замена	
2	2	2	2024-01-16	Износ тормозных колодок, скрип при торможении	
3	3	3	2024-01-17	Погнутый обод переднего колеса	
4	4	4	2024-01-18	Проблемы с переключением передач	
5	5	5	2024-01-19	Требуется полная регулировка трансмиссии	
6	6	6	2024-01-20	Замена цепи и звезд	
7	7	7	2024-01-21	Обслуживание вилки и амортизаторов	
8	8	8	2024-01-22	Установка нового оборудования	
9	9	9	2024-01-23	Диагностика электронной системы переключения	
10	10	10	2024-01-24	Комплексное обслуживание после сезона	
11	11	1	2024-02-01	Тестовый заказ	

Рисунок 1 – Таблица application

user_order					
	123 ↗ user_order_id ↑	123 ↘ application_id	123 ↘ status_id	⌚ start_date	⌚ end_date
1	1	1	5	2024-01-15	2024-01-16
2	2	2	5	2024-01-16	2024-01-18
3	3	3	3	2024-02-17	2024-02-20
4	4	4	2	2024-07-18	2024-07-19
5	5	5	1	2024-07-19	2024-07-22
6	6	6	6	2024-11-20	2024-11-21
7	7	7	4	2025-01-21	2025-01-23
8	8	8	5	2025-01-22	2025-01-23
9	9	9	2	2025-01-23	2025-01-25
10	10	10	1	2025-01-24	2025-01-26
11	11	11	1	2024-02-01	2024-02-05

Рисунок 2 – Таблица user_order

transaction			
	123 ↗ transaction_id	123 ↘ user_order_id	⌚ payment_date
1	1	1	2024-01-16
2	2	6	2024-01-21
3	3	8	2024-01-23

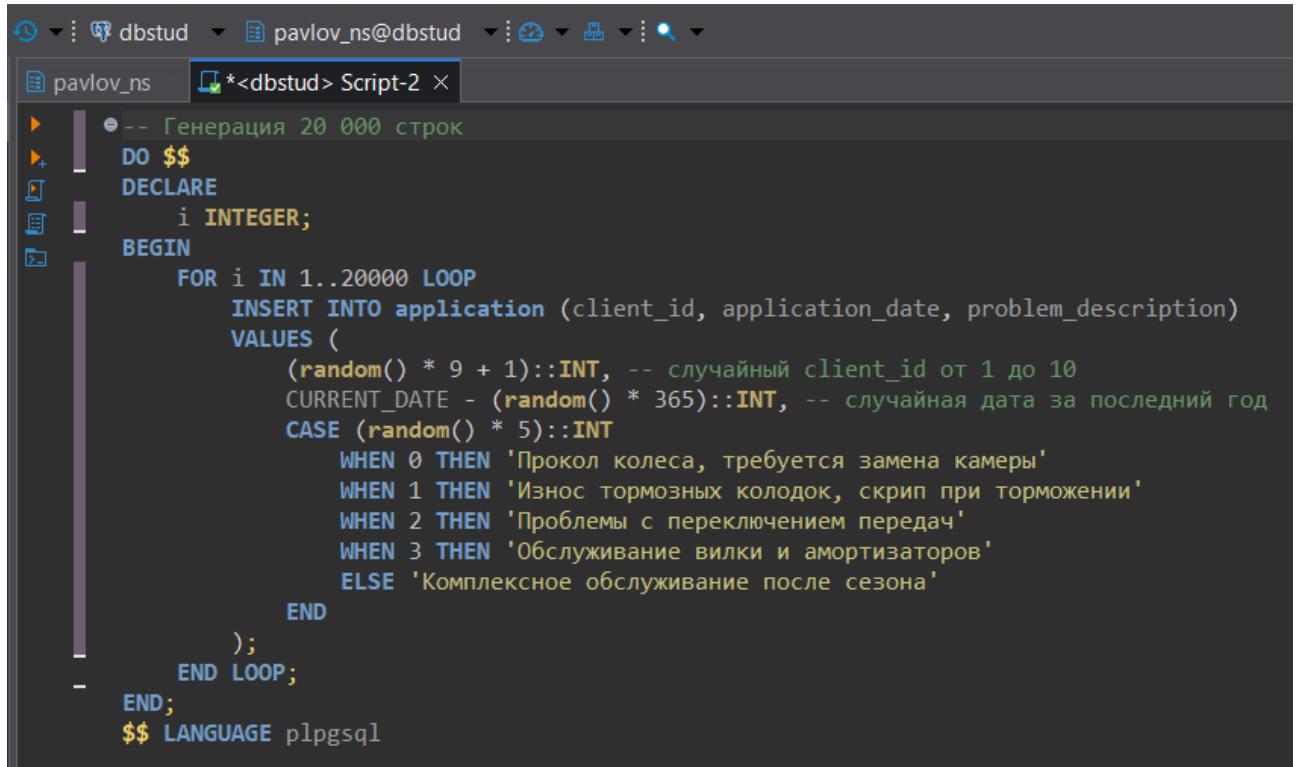
Рисунок 3 – Таблица transaction

work | Введите SQL выражение чтобы отфильтровать результаты

	work_id	title	description_of_the_execution	lead_time	price
Таблица					
1	3	Правка обода	Выравнивание обода на станке, регулировка спиц	90	1 500
2	4	Регулировка переключения	Настройка переднего и заднего переключателей	60	1 000
3	5	Полная регулировка трансмиссии	Чистка, смазка, регулировка всей трансмиссии	120	2 000
4	6	Замена цепи и звезд	Замена цепи и кассеты, регулировка	75	1 200
5	7	Обслуживание вилки	Разборка, чистка, замена масла, сборка	180	3 000
6	8	Установка оборудования	Установка и настройка нового оборудования	60	1 000
7	9	Диагностика электроники	Проверка электронной системы, перепрошивка	90	1 500
8	10	Комплексное обслуживание	Полная диагностика и обслуживание всех систем	240	4 000
9	1	Замена камеры	Демонтаж покрышки, замена камеры, монтаж покрышки	30	585
10	2	Замена тормозных колодок	Снятие старых колодок, установка новых, регулировка	45	860

Рисунок 4 – Таблица work

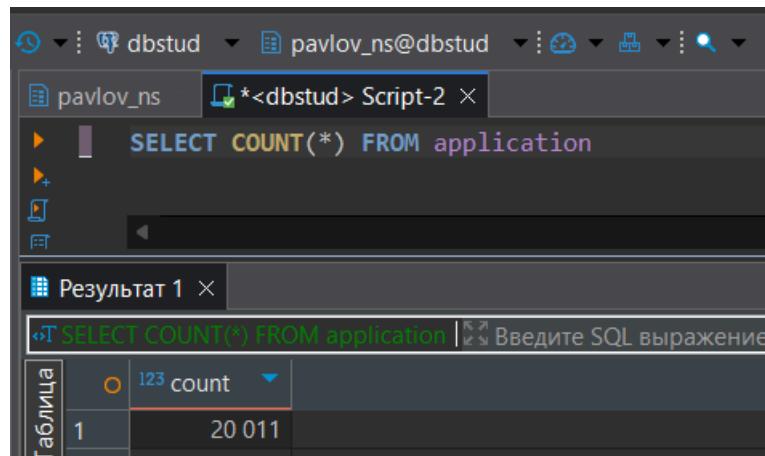
2.2 ПОДГОТОВКА БД



```
-- Генерация 20 000 строк
DO $$

DECLARE
    i INTEGER;
BEGIN
    FOR i IN 1..20000 LOOP
        INSERT INTO application (client_id, application_date, problem_description)
        VALUES (
            (random() * 9 + 1)::INT, -- случайный client_id от 1 до 10
            CURRENT_DATE - (random() * 365)::INT, -- случайная дата за последний год
            CASE (random() * 5)::INT
                WHEN 0 THEN 'Прокол колеса, требуется замена камеры'
                WHEN 1 THEN 'Износ тормозных колодок, скрип при торможении'
                WHEN 2 THEN 'Проблемы с переключением передач'
                WHEN 3 THEN 'Обслуживание вилки и амортизаторов'
                ELSE 'Комплексное обслуживание после сезона'
            END
        );
    END LOOP;
END;
$$ LANGUAGE plpgsql
```

Рисунок 5 – Генерация 20000 строк в таблицу application для анализа



```
SELECT COUNT(*) FROM application
```

Таблица	count
1	20 011

Рисунок 6 – Проверка добавления строк

2.3 ЗАДАНИЕ 1. АНАЛИЗ И ОПТИМИЗАЦИЯ

2.3.1 Сценарий 1. B-Tree

The screenshot shows the pgAdmin interface with the following details:

- Connection: dbstud - pavlov_ns@dbstud
- Tab: Analyze
- Query: `EXPLAIN ANALYZE SELECT * FROM application WHERE application_date = '2024-01-15'`
- Results:
 - QUERY PLAN:
 - Seq Scan on application (cost=0.00..537.14 rows=49 width=83) (actual time=0.093..4.676 rows=1 loops=1)
 - Filter: (application_date = '2024-01-15'::date)
 - Rows Removed by Filter: 20010
 - Planning Time: 0.071 ms
 - Execution Time: 4.701 ms

Рисунок 7 – Анализ запроса без индекса

The screenshot shows the pgAdmin interface with the following details:

- Connection: dbstud - pavlov_ns@dbstud
- Tab: Index
- Query: `CREATE INDEX idx_application_date ON application(application_date)`
- Results:
 - Статистика 1:

Name	Value
Updated Rows	0
Execute time	0.037s
Start time	Fri Dec 05 09:25:53 MSK 2025
Finish time	Fri Dec 05 09:25:53 MSK 2025
Query	CREATE INDEX idx_application_date ON application(application_date)

Рисунок 8 – Создание индекса дат для таблицы

```

EXPLAIN ANALYZE
SELECT * FROM application
WHERE application_date = '2024-01-15'

 QUERY PLAN
 1  Bitmap Heap Scan on application (cost=4.67..134.03 rows=49 width=83) (actual time=0.031..0.032 rows=1 loops=1)
    Recheck Cond: (application_date = '2024-01-15'::date)
    Heap Blocks: exact=1
   -> Bitmap Index Scan on idx_application_date (cost=0.00..4.65 rows=49 width=0) (actual time=0.019..0.019 rows=1 loops=1)
        Index Cond: (application_date = '2024-01-15'::date)
 Planning Time: 0.102 ms
 Execution Time: 0.059 ms

```

Рисунок 9 – Анализ запроса с созданным индексом

Таблица 1 – Сравнительная таблица времени выполнения

Метрика	До оптимизации	После оптимизации	Вывод
План (Оператор)	Seq Scan	Bitmap Index Scan	Выбор отличается
Execution Time	4.701 ms	0.059 ms	Запрос ускорился примерно в 80 раз

2.3.2 Сценарий 2. Индекс по выражению

The screenshot shows the pgAdmin interface with the following details:

- Connection: dbstud - pavlov_ns@dbstud
- Query tab: *<dbstud> Analyze (highlighted)
- Text tab: *application
- Query content:

```
EXPLAIN ANALYZE
SELECT * FROM application
WHERE LOWER(problem_description) = 'комплексное обслуживание после сезона'
```
- Result tab: Результат 1 (highlighted)
- Result content:

Показатель	Значение
QUERY PLAN	A-Z
Seq Scan on application	(cost=0.00..587.16 rows=100 width=83) (actual time=0.057..51.733 rows=5935 loops=1)
Filter:	(lower(problem_description) = 'комплексное обслуживание после сезона')::text
Rows Removed by Filter:	14076
Planning Time:	0.085 ms
Execution Time:	52.098 ms

Рисунок 10 – Анализ запроса без индекса

The screenshot shows the pgAdmin interface with the following details:

- Connection: dbstud - pavlov_ns@dbstud
- Query tab: *<dbstud> Index (highlighted)
- Text tab: *application
- Query content:

```
CREATE INDEX idx_application_problem_lower ON application(LOWER(problem_description))
```
- Result tab: Статистика 1 (highlighted)
- Result content:

Name	Value
Updated Rows	0
Execute time	0.088s
Start time	Fri Dec 05 09:30:43 MSK 2025
Finish time	Fri Dec 05 09:30:43 MSK 2025
Query	CREATE INDEX idx_application_problem_lower ON application(LOWER(problem_description))

Рисунок 11 – Создание индекса по выражению LOWER()

The screenshot shows the pgAdmin interface with the following details:

- Connection: dbstud - pavlov_ns@dbstud
- Query tab: *<dbstud> Analyze (highlighted)
- Text tab: *application
- Query content:

```
EXPLAIN ANALYZE
SELECT * FROM application
WHERE LOWER(problem_description) = 'комплексное обслуживание после сезона'
```
- Result tab: Результат 1 (highlighted)
- Result content:

Показатель	Значение
QUERY PLAN	A-Z
Bitmap Heap Scan on application	(cost=5.06..209.33 rows=100 width=83) (actual time=0.212..1.535 rows=5935 loops=1)
Recheck Cond:	(lower(problem_description) = 'комплексное обслуживание после сезона')::text
Heap Blocks:	exact=287
->	Bitmap Index Scan on idx_application_problem_lower (cost=0.00..5.04 rows=100 width=0) (actual time=0.169..0.169)
Index Cond:	(lower(problem_description) = 'комплексное обслуживание после сезона')::text
Planning Time:	0.116 ms
Execution Time:	1.819 ms

Рисунок 12 – Анализ запроса с использованием индекса

Таблица 2 – Сравнительная таблица времени выполнения

Метрика	До оптимизации	После оптимизации	Вывод
План (Оператор)	Seq Scan	Bitmap Index Scan	Выбор отличается
Execution Time	52.098 ms	1.819 ms	Запрос ускорился примерно в 28 раз

2.3.3 Сценарий 3. Частичный индекс

The screenshot shows the pgAdmin interface with the following details:

- Session: dbstud - pavlov_ns@dbstud
- Query tab: *<dbstud> Analyze (selected)
- Query content:

```
EXPLAIN ANALYZE
SELECT * FROM application
WHERE client_id = 1
```
- Results tab: Результат 1 (Result 1) (selected)
- Result content:

```
EXPLAIN ANALYZE SELECT * FROM application | Введите SQL выражение чтобы отфильтровать результаты
```

Таблица	QUERY PLAN
1	Seq Scan on application (cost=0.00..537.14 rows=1154 width=83) (actual time=0.078..6.644 rows=1154 loops=1)
2	Filter: (client_id = 1)
3	Rows Removed by Filter: 18857
4	Planning Time: 0.160 ms
5	Execution Time: 6.797 ms

Рисунок 13 – Анализ запроса без индекса

The screenshot shows the pgAdmin interface with the following details:

- Session: dbstud - pavlov_ns@dbstud
- Query tab: *<dbstud> Index (selected)
- Query content:

```
CREATE INDEX idx_application_client_1 ON application(application_date)
WHERE client_id = 1
```
- Results tab: Статистика 1 (Statistics 1) (selected)
- Result content:

Name	Value
Updated Rows	0
Execute time	0.020s
Start time	Fri Dec 05 09:41:07 MSK 2025
Finish time	Fri Dec 05 09:41:08 MSK 2025
Query	CREATE INDEX idx_application_client_1 ON application(application_date) WHERE client_id = 1

Рисунок 14 – Добавление частичного индекса

The screenshot shows the pgAdmin interface with the following details:

- Connections:** dbstud (selected), pavlov_ns
- Current Tab:** Analyze (highlighted)
- Query:**

```

EXPLAIN ANALYZE
SELECT * FROM application
WHERE client_id = 1
    
```
- Result Panel:** Результат 1 (Result 1)
 - Text tab (selected): EXPLAIN ANALYZE SELECT * FROM application
 - Table tab: A-Z QUERY PLAN

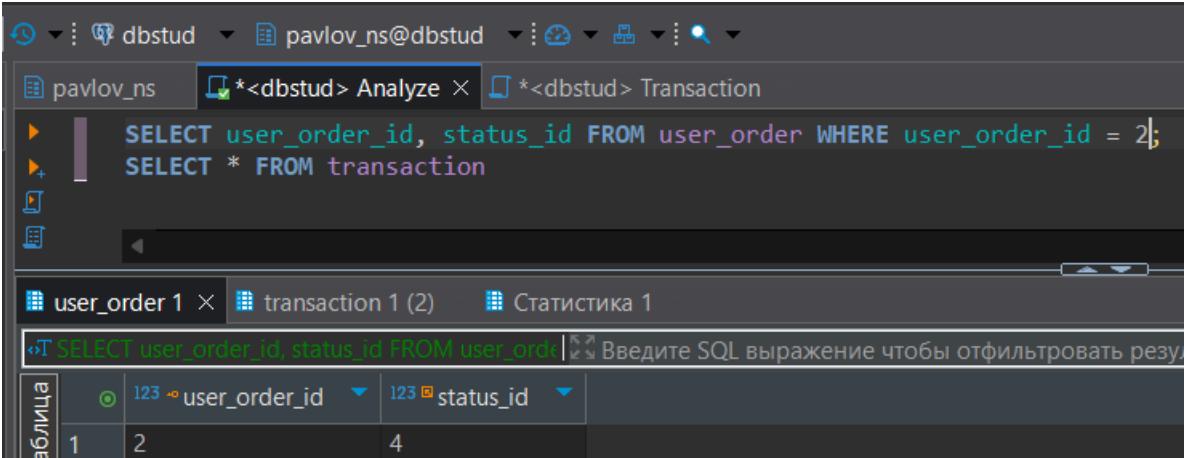
Номер	Описание
1	Bitmap Heap Scan on application (cost=22.34..323.76 rows=1154 width=83) (actual time=0.115..0.682 rows=1154)
2	Recheck Cond: (client_id = 1)
3	Heap Blocks: exact=286
4	-> Bitmap Index Scan on idx_application_client_1 (cost=0.00..22.05 rows=1154 width=0) (actual time=0.073..0.073 rows=1154)
5	Planning Time: 0.099 ms
6	Execution Time: 0.755 ms

Рисунок 15 – Анализ запроса с использованием частичного индекса

Таблица 3 – Сравнительная таблица времени выполнения

Метрика	До оптимизации	После оптимизации	Вывод
План (Оператор)	Seq Scan	Bitmap Index Scan	Выбор отличается
Execution Time	6.797 ms	0.755 ms	Запрос ускорился примерно в 9 раз

2.4 ЗАДАНИЕ 2. ДЕМОНСТРАЦИЯ АТОМАРНОСТИ. УСПЕШНЫЙ COMMIT



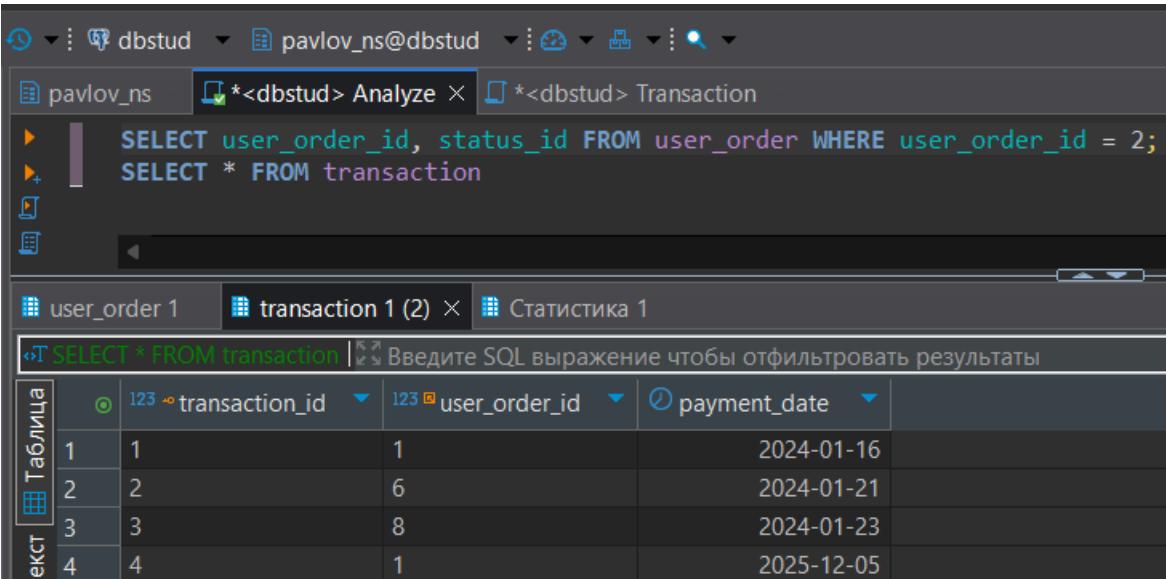
The screenshot shows a database interface with the following details:

- Connection: dbstud / pavlov_ns@dbstud
- Current Schema: pavlov_ns
- Open Queries:
 - SELECT user_order_id, status_id FROM user_order WHERE user_order_id = 2;
 - SELECT * FROM transaction
- Result Set for user_order:

user_order_id	status_id
1	2
- Result Set for transaction:

transaction_id	user_order_id	payment_date
1	1	2024-01-16
2	6	2024-01-21
3	8	2024-01-23
4	1	2025-12-05

Рисунок 16 – Проверка исходного статуса заказа



The screenshot shows a database interface with the following details:

- Connection: dbstud / pavlov_ns@dbstud
- Current Schema: pavlov_ns
- Open Queries:
 - SELECT user_order_id, status_id FROM user_order WHERE user_order_id = 2;
 - SELECT * FROM transaction
- Result Set for user_order:

user_order_id	status_id
1	2
- Result Set for transaction:

transaction_id	user_order_id	payment_date
1	1	2024-01-16
2	6	2024-01-21
3	8	2024-01-23
4	1	2025-12-05

Рисунок 17 – Проверка исходного состояния таблицы transaction

The screenshot shows the dbstud database interface. In the top navigation bar, it says 'dbstud' and 'pavlov_ns@dbstud'. Below the navigation bar, there are two tabs: 'pavlov_ns' and 'Transaction'. The 'Transaction' tab is active, displaying the following SQL code:

```

BEGIN;
UPDATE user_order
SET status_id = (SELECT status_id FROM status WHERE title = 'Готово')
WHERE user_order_id = 2;

INSERT INTO transaction (user_order_id, payment_date)
VALUES (2, CURRENT_DATE);
COMMIT;

```

Below the code, there is a section titled 'Статистика 1' (Statistics 1) containing the following data:

Name	Value
Queries	4
Updated Rows	2
Execute time	0.110s
Fetch time	0.000s
Total time	0.110s
Start time	2025-12-05 10:03:48.409
Finish time	2025-12-05 10:03:48.580

Рисунок 18 – Выполнение атомарной операции обновления данных

The screenshot shows the dbstud database interface. In the top navigation bar, it says 'dbstud' and 'pavlov_ns@dbstud'. Below the navigation bar, there are two tabs: 'pavlov_ns' and 'Transaction'. The 'Transaction' tab is active, displaying the following SQL code:

```

SELECT user_order_id, status_id FROM user_order WHERE user_order_id = 2;
SELECT * FROM transaction

```

Below the code, there are two table viewers. The first viewer is for the 'user_order' table, showing the following data:

Таблица	123 ↗ user_order_id	123 ↗ status_id
1	2	5

The second viewer is for the 'transaction' table, showing the following data:

Таблица	123 ↗ transaction_id	123 ↗ user_order_id	payment_date
1	1	1	2024-01-16
2	2	6	2024-01-21
3	3	8	2024-01-23
4	4	1	2025-12-05
5	5	2	2025-12-05

Рисунок 19 – Проверка обновления статуса заказа

The screenshot shows the dbstud database interface. In the top navigation bar, it says 'dbstud' and 'pavlov_ns@dbstud'. Below the navigation bar, there are two tabs: 'pavlov_ns' and 'Transaction'. The 'Transaction' tab is active, displaying the following SQL code:

```

SELECT user_order_id, status_id FROM user_order WHERE user_order_id = 2;
SELECT * FROM transaction

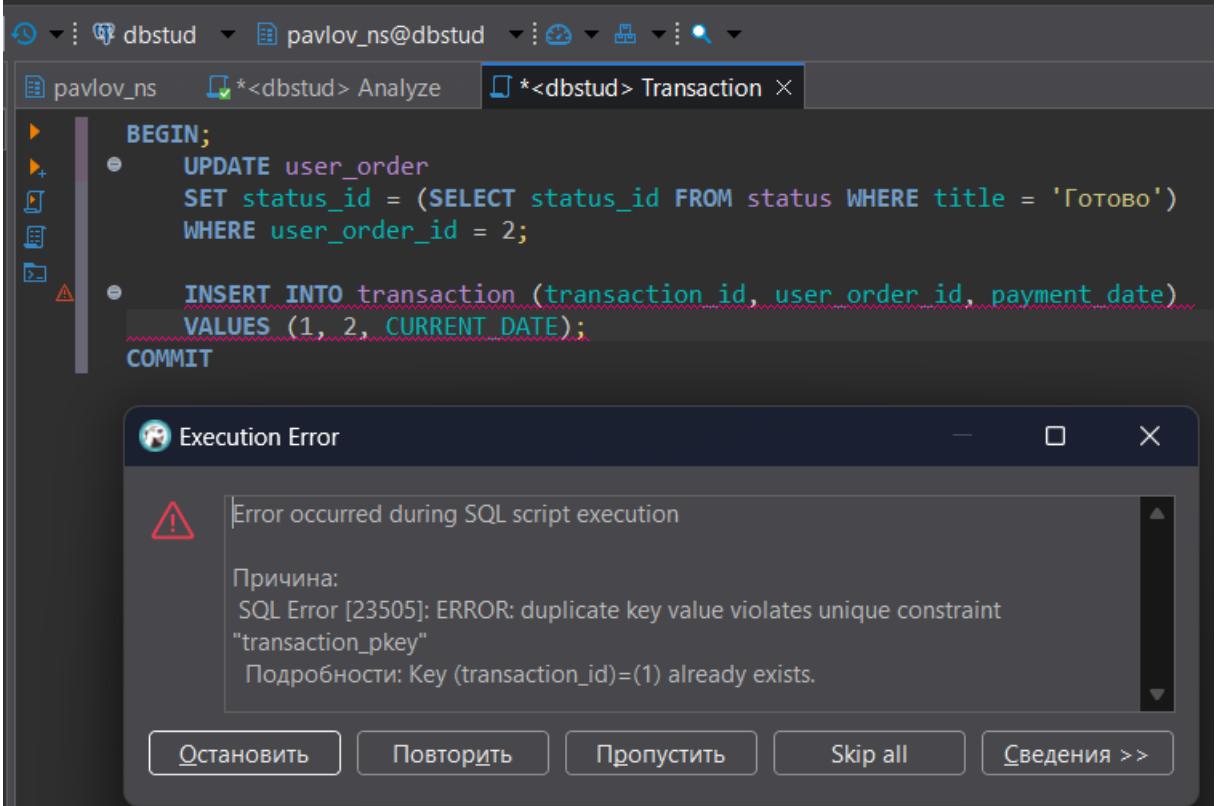
```

Below the code, there is a table viewer for the 'transaction' table, showing the following data:

Таблица	123 ↗ transaction_id	123 ↗ user_order_id	payment_date
1	1	1	2024-01-16
2	2	6	2024-01-21
3	3	8	2024-01-23
4	4	1	2025-12-05
5	5	2	2025-12-05

Рисунок 20 – Проверка обновления таблицы transaction

2.5 ЗАДАНИЕ 3. ДЕМОНСТРАЦИЯ АТОМАРНОСТИ. АВАРИЙНЫЙ ROLLBACK



The screenshot shows a database client interface with a transaction editor and an execution error dialog.

In the transaction editor:

```
BEGIN;
UPDATE user_order
SET status_id = (SELECT status_id FROM status WHERE title = 'Готово')
WHERE user_order_id = 2;

INSERT INTO transaction (transaction_id, user_order_id, payment_date)
VALUES (1, 2, CURRENT_DATE);

COMMIT;
```

An error icon is visible in the editor's sidebar.

The execution error dialog is titled "Execution Error" and contains the following information:

Error occurred during SQL script execution

Причина:
SQL Error [23505]: ERROR: duplicate key value violates unique constraint
"transaction_pkey"
Подробности: Key (transaction_id)=(1) already exists.

Buttons at the bottom of the dialog: Остановить, Повторить, Пропустить, Skip all, Сведения >>

Рисунок 21 – Выполнение запроса с ошибкой

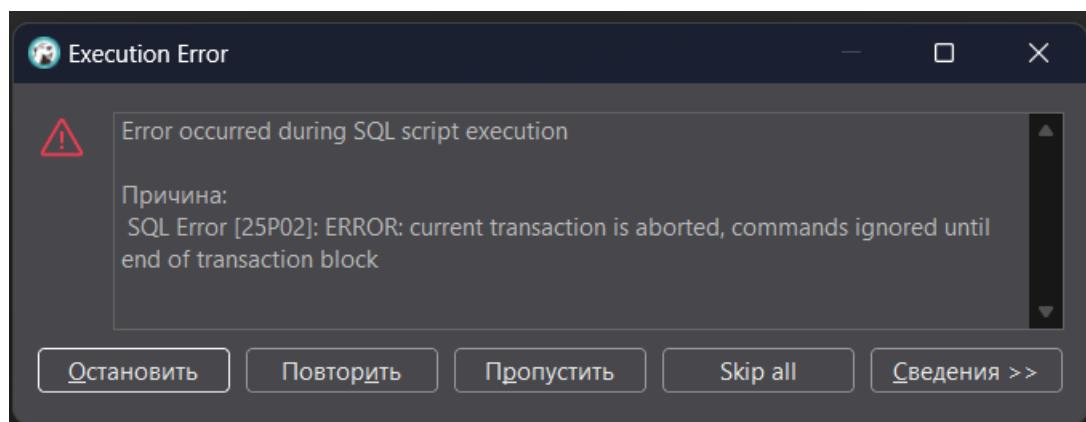


Рисунок 22 – Ошибка при попытке выполнить любой запрос

The screenshot shows a MySQL Workbench interface. In the top tab bar, the connection is set to 'dbstud' and the schema is 'pavlov_ns'. There are two tabs: 'Analyze' and 'Transaction'. The 'Transaction' tab is active, displaying the following SQL code:

```
SET status_id = (SELECT status_id FROM status WHERE title = 'Готово')  
WHERE user_order_id = 2;  
INSERT INTO transaction (transaction_id, user_order_id, payment_date)  
VALUES (1, 2, CURRENT_DATE);  
COMMIT;  
ROLLBACK
```

Below the code, there is a 'Статистика 1' (Statistics 1) panel showing the following results:

Name	Value
Updated Rows	0
Execute time	0.016s
Start time	Fri Dec 05 10:20:45 MSK 2025
Finish time	Fri Dec 05 10:20:45 MSK 2025
Query	ROLLBACK

Рисунок 23 – Выполнение ROLLBACK

The screenshot shows the same MySQL Workbench interface. The 'Transaction' tab is active, displaying the following SQL code:

```
SELECT user_order_id, status_id FROM user_order WHERE user_order_id = 2;  
SELECT * FROM transaction
```

Below the code, there are three tabs: 'user_order 1', 'transaction 1 (2)', and 'Статистика 1'. The 'transaction 1 (2)' tab is active, showing a table with the following data:

Таблица	transaction_id	user_order_id	payment_date
Текст	1	1	2024-01-16
	2	6	2024-01-21
	3	8	2024-01-23
	4	1	2025-12-05
	5	2	2025-12-05

Рисунок 24 – Проверка состояния после ROLLBACK

2.6 ЗАДАНИЕ 4. МОДЕЛИРОВАНИЕ АНОМАЛИИ «НЕПОВТОРЯЕМОЕ ЧТЕНИЕ»

The screenshot shows a pgAdmin interface with a transaction named 'pavlov_ns'. The transaction code is as follows:

```
BEGIN;  
SELECT price FROM work WHERE work_id = 10;  
SELECT pg_sleep(10);  
SELECT price FROM work WHERE work_id = 10;  
COMMIT;
```

Below the transaction, there is a table named 'work' with one row:

аблица	123 price
1	4 000

Рисунок 25 – Вывод при первом чтении

The screenshot shows a pgAdmin interface with a transaction named 'pavlov_ns'. The transaction code is:

```
UPDATE work SET price = price * 2 WHERE work_id = 10
```

Below the transaction, there is a table titled 'Статистика 1' (Statistics 1) showing the results of the update:

Name	Value
Updated Rows	1
Execute time	0.010s
Start time	Fri Dec 05 10:44:13 MSK 2025
Finish time	Fri Dec 05 10:44:13 MSK 2025
Query	UPDATE work SET price = price * 2 WHERE work_id = 10

Рисунок 26 – Обновление данных во время ожидания

The screenshot shows a pgAdmin 4 interface. At the top, there's a toolbar with various icons. Below it is a tab bar with three tabs: 'pavlov_ns' (selected), 'Analyze', and 'Transaction'. The main area contains a vertical list of SQL statements:

```
BEGIN;  
SELECT price FROM work WHERE work_id = 10;  
SELECT pg_sleep(10);  
SELECT price FROM work WHERE work_id = 10;  
COMMIT;
```

Below this is a horizontal tab bar with four tabs: 'work 1', 'Результат 1 (2)', 'work 1 (3)' (selected), and 'Статистика 1'. Under 'work 1 (3)', there's a search bar containing the text 'SELECT price FROM work WHERE work_id = 10' and a note 'Введите SQL выражение'. A results table is displayed:

Таблица	price
1	8 000

Рисунок 27 – Вывод при втором чтении

2.7 ЗАДАНИЕ 5. УСТРАНЕНИЕ АНОМАЛИИ «НЕПОВТОРЯЕМОЕ ЧТЕНИЕ»

The screenshot shows a pgAdmin interface with a transaction script and its execution results.

Transaction Script:

```
BEGIN;  
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
SELECT price FROM work WHERE work_id = 10;  
SELECT pg_sleep(10);  
SELECT price FROM work WHERE work_id = 10;  
COMMIT;
```

Execution Results:

абзаца	123 price
1	4 000

Рисунок 28 – Вывод при первом чтении

The screenshot shows a pgAdmin interface with an UPDATE query and its execution statistics.

Query:

```
UPDATE work SET price = price * 2 WHERE work_id = 10
```

Statistics:

Name	Value
Updated Rows	1
Execute time	0.017s
Start time	Fri Dec 05 10:47:33 MSK 2025
Finish time	Fri Dec 05 10:47:33 MSK 2025
Query	UPDATE work SET price = price * 2 WHERE work_id = 10

Рисунок 29 – Обновление данных во время ожидания

The screenshot shows the pgAdmin interface with a transaction script and its execution results.

Transaction Script:

```
BEGIN;  
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
SELECT price FROM work WHERE work_id = 10;  
SELECT pg_sleep(10);  
SELECT price FROM work WHERE work_id = 10;  
COMMIT;
```

Execution Results:

Таблица	price
1	4 000

Рисунок 30 – Вывод при втором чтении