

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«Национальный исследовательский университет ИТМО»  
(Университет ИТМО)**

**Факультет прикладной информатики**

**Образовательная программа Мобильные и сетевые технологии**

**Направление подготовки 09.03.03 Мобильные и сетевые технологии**

**О Т Ч Е Т**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**"ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ. ПРЕДСТАВЛЕНИЯ.  
РАБОТА С ИНДЕКСАМИ"**

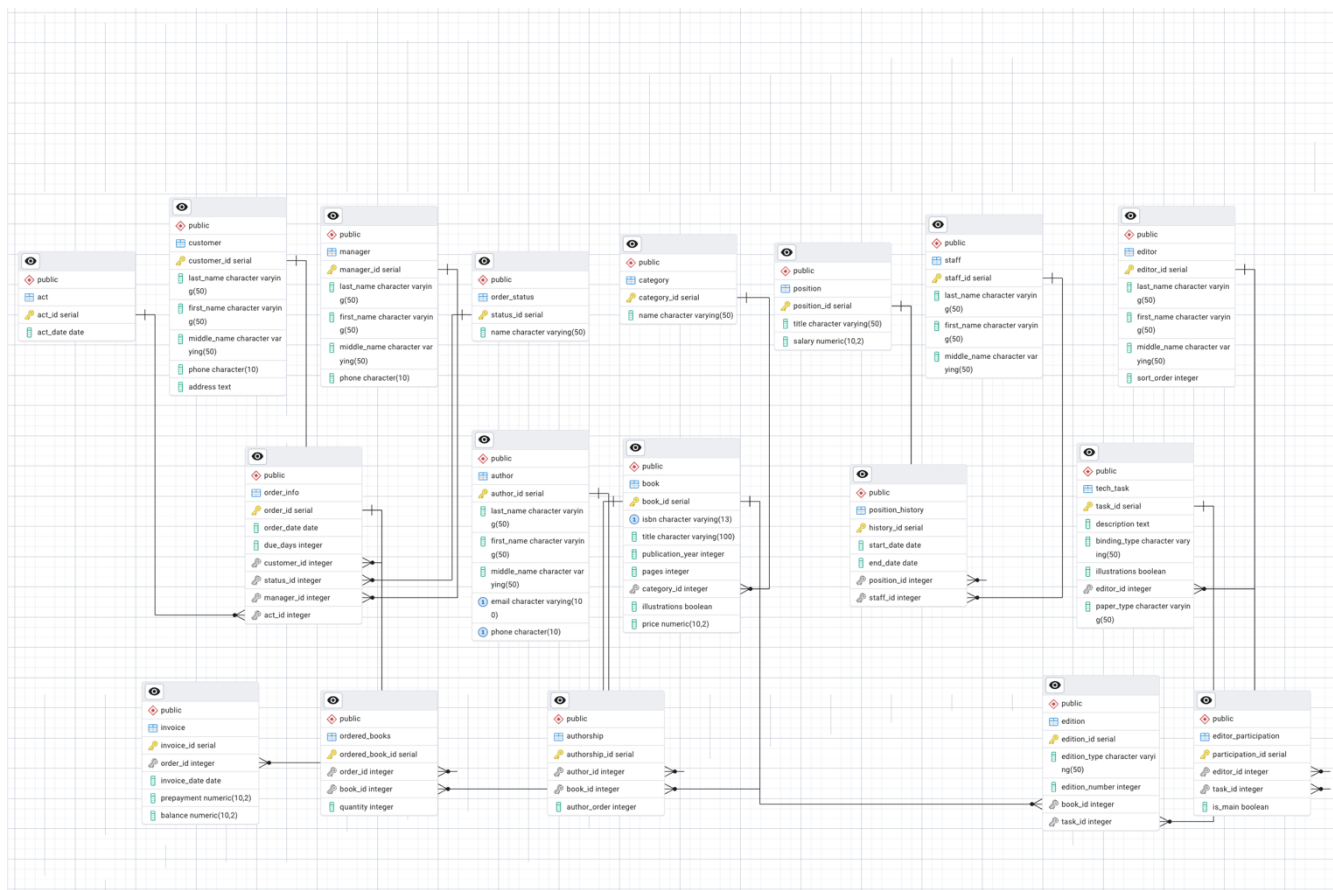
**Обучающийся: Кошкарев Кирилл 3239**

**Преподаватель: Говорова Марина Михайловна**

Санкт-  
Петербург,

**1. Цель работы:** овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

**2. Схема базы данных (ЛР 3).**



**3. Выполнение:**

**3.1 Запросы к базе данных.**

В рамках выполнения лабораторной работы были составлены и выполнены SQL-запросы в соответствии с индивидуальным заданием (часть 2). Каждый запрос формировался исходя из требований задания, отражающих конкретные задачи по выборке данных из базы.

Запрос 1: найти книги, технические задания на которые были оформлены, с указанием даты и редактора

Query

Query History

1

SELECT

b."title",

-- название книги

pr."date",

-- дата тиража (аналог даты ТЗ)

e."last\_name" || ' ' || e."first\_name" || ' ' || e."middle\_name" AS "editor\_name" -- ФИО редактора

5

FROM

"tech\_task" t

6

JOIN

"edition" ed ON ed."tech\_task\_id" = t."tech\_task\_id"

-- связь с изданием

7

JOIN

"book" b ON ed."book\_id" = b."book\_id"

-- книга из издания

8

JOIN

"print\_run" pr ON pr."print\_run\_id" = b."print\_run\_id"

-- дата тиража

9

JOIN

"editor" e ON t."editor\_id" = e."editor\_id";

-- редактор по ТЗ

10

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🗑️

📥

📥

📶

SQL

Showing rows: 1 to 2

Page No: 1

	title text	date date	editor_name text
1	Основы PostgreSQL	2024-02-10	Фёдоров Антон Леонидович
2	Сетевое программирование	2023-11-15	Мельник Елена Владимировна

Запрос 2: найти заказы клиентов с задолженностью или просрочкой и указать менеджеров, оформлявших их

QueryQuery History

```

1 SELECT
2     c."last_name" || ' ' || c."first_name" || ' ' || c."middle_name" AS "client_name", -- клиент
3     o."order_id", -- номер заказа
4     m."last_name" || ' ' || m."first_name" || ' ' || m."middle_name" AS "manager_name" -- менеджер
5 FROM "orders" o
6 JOIN "client" c ON o."client_id" = c."client_id" -- заказ → клиент
7 JOIN "manager" m ON o."manager_id" = m."manager_id" -- заказ → менеджер
8 WHERE o."status_id" IN (
9     SELECT "status_id"
10    FROM "order_status"
11   WHERE "name" ILIKE '%долг%' OR "name" ILIKE '%просроч%' -- статусы с долгами
12 );
13

```

Data OutputMessagesNotifications

Showing rows: 1 to 2

Page No: 1 of 1

	client_name text	order_id [PK] integer	manager_name text
1	Петров Андрей Игоревич	5	Иванов Дмитрий Васильевич
2	Петров Андрей Игоревич	6	Иванов Дмитрий Васильевич

Запрос 3: найти заказы, сумма которых выше средней суммы по всем заказам

QueryQuery History

```

1 SELECT
2     i."order_id",
3     i."prepayment" + i."balance" AS "total_sum"
4 FROM "invoice" i
5 WHERE (i."prepayment" + i."balance") > (
6     SELECT AVG("prepayment" + "balance") FROM "invoice"
7 );
8

```

Data OutputMessagesNotifications

Showing rows: 1 to 1

Page No:

	order_id integer	total_sum numeric
1	1	7000.00

Запрос 4: определить общую сумму заказов, оформленных каждым менеджером.

Query Query History

```
1 SELECT
2     m."last_name" || ' ' || m."first_name" AS "manager",
3     SUM(i."prepayment" + i."balance") AS "total_revenue"
4 FROM "orders" o
5 JOIN "invoice" i ON o."order_id" = i."order_id"
6 JOIN "manager" m ON o."manager_id" = m."manager_id"
7 GROUP BY manager;
8
```

Data Output Messages Notifications



Showing rows: 1 to 2

	manager text	total_revenue numeric
1	Иванов Дмитрий	7000.00
2	Кузнецова Мария	4500.00

## Запрос 5: вывести клиентов и количество заказов, которые они оформили

Query Query History

```
1 SELECT
2     c."last_name" || ' ' || c."first_name" AS "client",
3     COUNT(o."order_id") AS "order_count"
4 FROM "client" c
5 LEFT JOIN "orders" o ON c."client_id" = o."client_id"
6 GROUP BY client;
```

Data Output Messages Notifications

SQL

Showing rows: 1 to

	client text	order_count bigint
1	Иванов Иван	0
2	Сидоров Борис	1
3	Петров Андрей	3
4	Петров Петр	0

### Запрос 6: найти клиентов, которые не сделали ни одного заказа

Query

Query History

1

2

3

4

5

6

7

SELECT

c."last\_name" || ' ' || c."first\_name" AS "client"

FROM "client" c

WHERE c."client\_id" NOT IN (

SELECT DISTINCT "client\_id" FROM "orders"

);

Data Output

Messages

Notifications

≡

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows

	client text	🔒
1	Иванов Иван	
2	Петров Петр	

### Запрос 7: определить клиента, оформившего самый дорогой заказ

Query
Query History

```

1  SELECT
2      c."last_name" || ' ' || c."first_name" AS "client",
3      (i."prepayment" + i."balance") AS "total_sum"
4  FROM "invoice" i
5  JOIN "orders" o ON i."order_id" = o."order_id"
6  JOIN "client" c ON o."client_id" = c."client_id"
7  WHERE (i."prepayment" + i."balance") = (
8      SELECT MAX("prepayment" + "balance") FROM "invoice"
9  );
10

```

Data Output
Messages
Notifications

Showing rows: 1 to 1

	client text	total_sum numeric
1	Петров Андрей	7000.00

## 3.2 Представления

### Представление 1: заказы с долгами и их менеджеры

Query
Query History

```

1  CREATE VIEW overdue_orders AS
2  SELECT
3      c."last_name" || ' ' || c."first_name" AS "client",      -- Клиент
4      o."order_id",      -- Номер заказа
5      m."last_name" || ' ' || m."first_name" AS "manager"    -- Менеджер
6  FROM "orders" o
7  JOIN "client" c ON o."client_id" = c."client_id"
8  JOIN "manager" m ON o."manager_id" = m."manager_id"
9  WHERE o."status_id" IN (
10     SELECT "status_id"
11     FROM "order_status"
12     WHERE "name" ILIKE '%долг%' OR "name" ILIKE '%просроч%' -- Статусы с долгами или просроченные
13 );
14

```



- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1..3 Sequences
- > Tables (18)
- > Trigger Functions
- > Types
- > Views (1)
  - overdue\_orders
    - > Columns
    - > Rules
    - > Triggers
- > Subscriptions

Data Output Messages Notifications

	client text	order_id integer	manager text
1	Петров Андрей	5	Иванов Дмитрий
2	Петров Андрей	6	Иванов Дмитрий

## Представление 2: заказы с количеством книг более 10

Query Query History

```

1 CREATE VIEW large_orders AS
2 SELECT DISTINCT
3     o."order_id", -- Номер заказа
4     c."last_name" || ' ' || c."first_name" AS "client" -- ФИО клиента
5 FROM "ordered_books" ob
6 JOIN "orders" o ON ob."order_id" = o."order_id" -- Соединение с заказами
7 JOIN "client" c ON o."client_id" = c."client_id" -- Соединение с клиентами
8 WHERE ob."quantity" > 10; -- Заказы, где больше 10 книг
9

```

- > Rules
- > Triggers
- > orders
- > position
- > print\_run
- > tech\_task
- > Trigger Functions
- > Types
- > Views (2)
  - large\_orders
  - overdue\_orders
    - > Columns
    - > Rules
    - > Triggers

Data Output Messages Notifications

	order_id integer	client text
1	2	Сидоров Борис
2	1	Петров Андрей

### 3.3 Запросы на модификацию данных

Выполнение запросов на модификацию данных (INSERT, UPDATE, DELETE) с подзапросами

В ходе работы были составлены и успешно выполнены три сложных запроса на модификацию данных:

Задание: добавить заказ от клиента с самым большим ID

Data Output

Messages

Notifications

≡+

▼

▼

Showing rows: 1 to 4

	order_id [PK] integer	order_date date	deadline date	completion_date date	status_id integer	client_id integer	manager_id integer	act_id integer
1	1	2025-04-01	2025-04-11	2025-04-10	1	1	1	1
2	2	2025-04-05	2025-04-12	2025-04-12	2	2	2	1
3	5	2025-06-10	2025-06-15	[null]	12	1	1	[null]
4	6	2025-06-05	2025-06-13	2025-06-17	14	1	1	[null]

Data Output

Messages

Notifications

Showing rows: 1 to 5

	order_id [PK] integer	order_date date	deadline date	completion_date date	status_id integer	client_id integer	manager_id integer	act_id integer
1	1	2025-04-01	2025-04-11	2025-04-10	1	1	1	1
2	2	2025-04-05	2025-04-12	2025-04-12	2	2	2	1
3	5	2025-06-10	2025-06-15	[null]	12	1	1	[null]
4	6	2025-06-05	2025-06-13	2025-06-17	14	1	1	[null]
5	7	2025-06-20	2025-07-04	[null]	1	8	1	[null]

Query	Query History
1	INSERT INTO "orders" (
2	"order_date", "deadline", "completion_date", "status_id", "client_id", "manager_id", "act_id"
3	)
4	SELECT
5	CURRENT_DATE,
6	CURRENT_DATE + INTERVAL '14 days',
7	NULL,
8	1,                    -- статус: "оплачен"
9	client_id,
10	1,
11	NULL
12	FROM "client"
13	ORDER BY client_id DESC
14	LIMIT 1;
15	

## Задание: обновить статус заказов без книг

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

Showing rows: 1 to 5 

✎

	order_id [PK] integer ✎	order_date date ✎	deadline date ✎	completion_date date ✎	status_id integer ✎	client_id integer ✎	manager_id integer ✎	act_id integer ✎
1	1	2025-04-01	2025-04-11	2025-04-10	1	1	1	1
2	2	2025-04-05	2025-04-12	2025-04-12	2	2	2	1
3	5	2025-06-10	2025-06-15	[null]	12	1	1	[null]
4	6	2025-06-05	2025-06-13	2025-06-17	14	1	1	[null]
5	7	2025-06-20	2025-07-04	[null]	1	8	1	[null]

## Query Query History

```

1  UPDATE "orders"
2  SET "status_id" = 3  -- например, "отменен"
3  WHERE "order_id" NOT IN (
4      SELECT DISTINCT "order_id" FROM "ordered_books"
5  );
6

```

Data Output

Messages

Notifications

Showing rows: 1 to 5

Page

	order_id [PK] integer	order_date date	deadline date	completion_date date	status_id integer	client_id integer	manager_id integer	act_id integer
1	1	2025-04-01	2025-04-11	2025-04-10	1	1	1	1
2	2	2025-04-05	2025-04-12	2025-04-12	2	2	2	1
3	5	2025-06-10	2025-06-15	[null]	3	1	1	[null]
4	6	2025-06-05	2025-06-13	2025-06-17	3	1	1	[null]
5	7	2025-06-20	2025-07-04	[null]	3	8	1	[null]

## Задание: обновить статус заказов без книг

Data Output

Messages

Notifications

≡

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	invoice_id [PK] integer	order_id integer	invoice_date date	prepayment numeric (10,2)	balance numeric (10,2)
1	1	5	2025-04-01	5000.00	2000.00
2	2	2	2025-04-05	3000.00	1500.00
3	3	[null]	2025-04-06	2500.00	1000.00

Query

Query History

1

2

3

4

5

`DELETE FROM "invoice"
WHERE "order_id" NOT IN (
SELECT "order_id" FROM "orders"
);`

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	invoice_id [PK] integer	order_id integer	invoice_date date	prepayment numeric (10,2)	balance numeric (10,2)
1	1	5	2025-04-01	5000.00	2000.00
2	2	2	2025-04-05	3000.00	1500.00

### 3.4 Создание индексов

В данном пункте лабораторной работы были проведены следующие действия:

#### 1. Выполнение тестовых запросов без индексов

Были выполнены два выбранных запроса к базе данных, и с помощью команды EXPLAIN ANALYZE получены планы их выполнения. Зафиксировано время выполнения и использованные методы доступа к данным (например, последовательное сканирование).

Выберем 2 запроса:

А: поиск строк в таблице invoice, где значение в поле balance больше 1000. Без индекса на поле balance, PostgreSQL будет выполнять **полное сканирование таблицы** (поиск через все строки), что может занять много времени, если таблица большая.

Запрос В: выбрать фио клиента и количество его заказов, делая left join с таблицей заказов. Этот запрос может быть **тяжёлым** для выполнения, особенно если таблицы `client` и `orders` большие, потому что он включает **объединение и агрегацию данных**.

Query

Query History

1

2

3

4

5

6

EXPLAIN ANALYZE

SELECT c."last\_name" || ' ' || c."first\_name" AS "client", COUNT(o."order\_id") AS "order\_count"

FROM "client" c

LEFT JOIN "orders" o ON c."client\_id" = o."client\_id"

GROUP BY client;

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑

🗄

⬇

📈

SQL

Showing rows: 1 to 11

✎

Page No: 1

QUERY PLAN

text

🔒

1

HashAggregate (cost=60.25..63.25 rows=200 width=40) (actual time=0.070..0.072 rows=4 loops=1)

2

Group Key: ((c.last\_name || ' ':text) || c.first\_name)

3

Batches: 1 Memory Usage: 40kB

4

-> Hash Right Join (cost=19.45..53.45 rows=1360 width=36) (actual time=0.057..0.062 rows=6 loops=...

5

Hash Cond: (o.client\_id = c.client\_id)

6

-> Seq Scan on orders o (cost=0.00..23.60 rows=1360 width=8) (actual time=0.008..0.008 rows=5 l...

7

-> Hash (cost=14.20..14.20 rows=420 width=68) (actual time=0.037..0.037 rows=4 loops=1)

8

Buckets: 1024 Batches: 1 Memory Usage: 9kB

9

-> Seq Scan on client c (cost=0.00..14.20 rows=420 width=68) (actual time=0.012..0.013 rows=...

10

Planning Time: 1.587 ms

11

Execution Time: 0.128 ms

## 2. Создание индексов

Для поля balance

Query Query History

```
1 CREATE INDEX idx_invoice_balance ON "invoice" ("balance");
2 |
```

Для соединений

Query Query History

```
1 CREATE INDEX idx_client_order ON "orders" ("client_id");
2
```

## 3. Выполнение тех же запросов с индексами

Data Output		Messages	Notifications
<div><div><div>≡+</div><div>▼</div><div>▼</div><div></div><div></div><div></div><div></div><div>SQL</div></div><div>Showing rows</div></div>			
	QUERY PLAN		
	text		
1	Seq Scan on invoice (cost=0.00..23.30 rows=1224 width=24) (actual time=0.011..0.145 rows=1224 loops=...		
2	Filter: (balance > '1000'::numeric)		
3	Planning Time: 0.069 ms		
4	Execution Time: 0.178 ms		

QUERY PLAN		Showing rows
	text	
1	HashAggregate (cost=22.19..25.19 rows=200 width=40) (actual time=0.069..0.072 rows=12 loops=...	
2	Group Key: ((c.last_name    ' '::text)    c.first_name)	
3	Batches: 1 Memory Usage: 40kB	
4	-> Hash Left Join (cost=1.11..20.09 rows=420 width=36) (actual time=0.055..0.060 rows=16 loop...	
5	Hash Cond: (c.client_id = o.client_id)	
6	-> Seq Scan on client c (cost=0.00..14.20 rows=420 width=68) (actual time=0.017..0.018 rows=...	
7	-> Hash (cost=1.05..1.05 rows=5 width=8) (actual time=0.025..0.025 rows=5 loops=1)	
8	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
9	-> Seq Scan on orders o (cost=0.00..1.05 rows=5 width=8) (actual time=0.014..0.014 rows=...	
10	Planning Time: 0.382 ms	
11	Execution Time: 0.167 ms	

Запросы были повторно выполнены, планы запросов с помощью EXPLAIN ANALYZE показали уменьшенное плановое время (для второго запроса незначительно, поскольку датасет малый).

Использование индексов особенно эффективно при увеличении объёмов данных или более сложных условиях выборки. Это было проверено по мере увеличения тестового дата сета с шагом 300.

### Удаление индексов

В конце созданные индексы были удалены, чтобы вернуть базу к исходному состоянию.

## 4. Вывод по лабораторной работе:

В процессе выполнения лабораторной работы были получены практические навыки разработки и выполнения сложных SQL-запросов для выборки и модификации данных в реляционной базе PostgreSQL.