

Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО

ОТЧЕТ

по Лабораторной работе 4

по дисциплине "Проектирование и реализация баз данных"

Обучающийся: Иванов Виктор Сергеевич

Факультет прикладной информатики

Группа К3240

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа: Мобильные и сетевые технологии 2023

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург
2024/2025

Цель работы

Овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и работы с индексами.

Практическое задание

1. Создать запросы и представления на выборку данных к базе данных.
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы и сравнить время выполнения запросов без индексов и с индексами.

Модель базы данных

(Модель из ЛР3, таблицы: client, contract, detail, contractDetail, drawing, detailDrawing, machine, steelSheet, manufacturingStage, pricePeriod)

ERD-схема БД

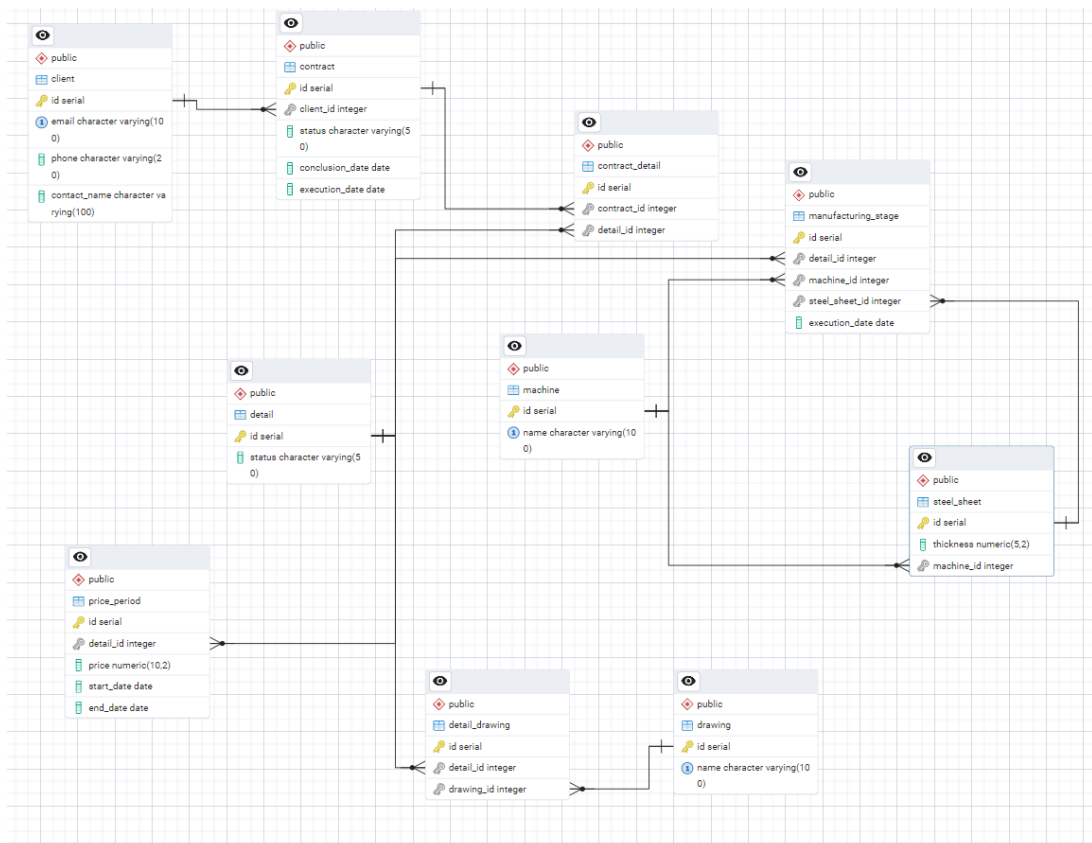


Рис. 1: ERD-схема БД

Выполнение работы

1. Запросы на выборку данных

1. Детали, не завершённые в течение последних 10 дней

Этот запрос позволяет определить детали, которые не были завершены в течение 10 дней с момента последнего этапа изготовления. Это позволяет отслеживать зависшие заказы.

```
SELECT *
FROM detail
WHERE status != 'выполнена'
AND id IN (
    SELECT detail_id
    FROM manufacturing_stage
    WHERE execution_date < CURRENT_DATE - INTERVAL '10 days'
);
```

Листинг 1: Незавершённые детали за последние 10 дней

	id [PK] integer	status character varying (50)
1	2	в производстве
2	4	в производстве
3	13	в производстве

Рис. 2: Результат запроса о незавершённых деталях

2. Станки, не задействованные сегодня

Показывает список станков, которые в текущую дату не используются. Это можно использовать для анализа простоев.

```
SELECT m.name
FROM machine m
WHERE NOT EXISTS (
    SELECT 1
    FROM manufacturing_stage ms
    WHERE ms.machine_id = m.id
    AND ms.execution_date = CURRENT_DATE
);
```

Листинг 2: Станки

3. Клиенты без активных договоров в текущем месяце

Позволяет определить клиентов, которые не оформляли новых договоров в текущем месяце. Полезно для оценки активности.


	name character varying (100) 
1	Станок 11
2	Станок 17
3	Станок 12
4	Станок 10
5	Станок 02
6	Станок 15
7	Станок 13
8	Станок 05
9	Станок 08
10	Станок 06
11	Станок 16
12	Станок 04
13	Станок 01
14	Станок 03
15	Станок 14
16	Станок 09
17	Станок 07

Рис. 3: Список станков без загрузки на сегодня

```

SELECT c.contact_name
FROM client c
WHERE NOT EXISTS (
    SELECT 1
    FROM contract ct
    WHERE ct.client_id = c.id
        AND date_trunc('month', ct.conclusion_date) = date_trunc('month',
CURRENT_DATE)
        AND ct.status = 'в работе'
);

```

Листинг 3: Клиенты без договоров в текущем месяце


	contact_name character varying (100) 
1	Иванов И.И.
2	Петрова А.А.
3	Smith Industries
4	Рогов П.П.
5	Иванова Л.С.
6	Test Client
7	Кузнецов В.Н.
8	Орлов А.Б.
9	Миронова Е.Д.
10	Сидоров К.Ю.
11	Белова Н.М.
12	Morgan Inc.
13	Жуков Ф.А.
14	Smithson Corp.
15	Демидов И.Г.
16	Андреева С.В.
17	Lee Solutions
18	Сергеев Т.О.

Рис. 4: Клиенты без активных договоров за текущий месяц

4. Детали с несколькими чертежами

Находит детали, у которых привязано более одного чертежа. Это полезно, чтобы отследить сложные или универсальные элементы.

```
SELECT d.id, COUNT(dd.drawing_id) AS drawing_count
FROM detail d
JOIN detail_drawing dd ON d.id = dd.detail_id
GROUP BY d.id
HAVING COUNT(dd.drawing_id) > 1;
```

Листинг 4: Детали с несколькими чертежами

	id [PK] integer	drawing_count bigint
1	4	2
2	6	2
3	5	2

Рис. 5: Детали с двумя и более чертежами

5. Детали без этапов изготовления

Находит детали, для которых ещё не началось изготовление. Это позволяет выявить «мертвые» заказы.

```
SELECT d.id, d.status
FROM detail d
WHERE NOT EXISTS (
    SELECT 1
    FROM manufacturing_stage ms
    WHERE ms.detail_id = d.id
);
```

Листинг 5: Детали без этапов изготовления

	id [PK] integer	status character varying (50)
1	17	выполнена
2	18	проект
3	15	проект
4	16	в производстве
5	1	проект
6	3	выполнена
7	9	проект

Рис. 6: Детали без связанных этапов изготовления

6. Детали с несколькими изменениями цены

Позволяет определить детали, у которых цена менялась более одного раза.

```
SELECT detail_id, COUNT(*) AS changes
FROM price_period
GROUP BY detail_id
HAVING COUNT(*) > 1;
```

Листинг 6: Детали с несколькими записями цены

	detail_id integer	changes bigint
1	5	2
2	4	2
3	6	2

Рис. 7: Детали с несколькими изменениями цен

7. Самый часто используемый чертёж

Находит чертёж, который используется в наибольшем количестве разных деталей, и выводит не только его ID, но и список связанных деталей.

```
WITH drawing_usage AS (
  SELECT drawing_id, COUNT(detail_id) AS usage_count
  FROM detail_drawing
  GROUP BY drawing_id
  ORDER BY usage_count DESC
  LIMIT 1
)
SELECT dd.drawing_id, dd.detail_id
FROM detail_drawing dd
JOIN drawing_usage du ON du.drawing_id = dd.drawing_id;
```

Листинг 7: Чертёж и список связанных деталей

	drawing_id integer	detail_id integer
1	5	6
2	5	4

Рис. 8: Наиболее используемый чертёж и его детали

2. Представления (Views)

Создадим представление **activeContracts**, которое формирует таблицу, содержащую информацию о действующих договорах в базе данных.

```
CREATE VIEW active_contracts AS
SELECT ct.id, c.contact_name, ct.conclusion_date
FROM contract ct
JOIN client c ON c.id = ct.client_id
WHERE ct.status = 'в работе';
```

Листинг 8: Создание представления *active_contracts*

Далее через команду SELECT получаем содержимое представления:

```
SELECT * FROM active_contracts;
```

Листинг 9: Вывод содержимого activeContracts

	id integer	contact_name character varying (100)	conclusion_date date
1	1	Иванов И.И.	2025-03-01
2	17	Петрова А.А.	2025-03-12
3	4	Рогов П.П.	2025-03-10
4	5	Иванова Л.С.	2025-02-15
5	7	Кузнецов В.Н.	2025-04-01
6	9	Миронова Е.Д.	2025-01-25
7	11	Белова Н.М.	2025-03-05
8	13	Жуков Ф.А.	2025-03-15
9	15	Демидов И.Г.	2025-04-10

Рис. 9: Создание представления activeContracts и его вывод

Создадим ещё одно представление **mostUsedDrawingView**, которое содержит информацию о чертеже, использованном в наибольшем количестве деталей, и список этих деталей.

```
CREATE VIEW most_used_drawing_view AS
SELECT d.id AS detail_id, dr.id AS drawing_id
FROM detail_drawing dd
JOIN drawing dr ON dd.drawing_id = dr.id
JOIN (
    SELECT drawing_id
    FROM detail_drawing
    GROUP BY drawing_id
    ORDER BY COUNT(detail_id) DESC
    LIMIT 1
) most_used ON dd.drawing_id = most_used.drawing_id
JOIN detail d ON dd.detail_id = d.id;
```

Листинг 10: Создание представления mostUsedDrawingView

Получим данные из представления:

```
SELECT * FROM most_used_drawing_view;
```

Листинг 11: Вывод содержимого mostUsedDrawingView

	detail_id integer	drawing_id integer
1	6	5
2	4	5

Рис. 10: Создание представления *most_used_drawing_view*

3. Модификация данных (INSERT, UPDATE, DELETE с подзапросами)

INSERT с подзапросом

Перед выполнением запроса выведем таблицу contract.

	id [PK] integer	client_id integer	status character varying (50)	conclusion_date date	execution_date date
1	1	1	в работе	2025-03-01	[null]
2	2	2	завершен	2024-11-10	2025-01-20
3	3	3	отменен	2025-01-05	[null]
4	4	4	в работе	2025-03-10	[null]
5	5	5	в работе	2025-02-15	[null]
6	6	6	завершен	2024-12-01	2025-01-30
7	7	7	в работе	2025-04-01	[null]
8	8	8	отменен	2025-02-20	[null]
9	9	9	в работе	2025-01-25	[null]
10	10	10	завершен	2024-10-10	2024-12-15
11	11	11	в работе	2025-03-05	[null]
12	12	12	завершен	2025-01-01	2025-02-25
13	13	13	в работе	2025-03-15	[null]
14	14	14	отменен	2025-02-01	[null]
15	15	15	в работе	2025-04-10	[null]
16	16	1	завершен	2024-09-05	2024-11-01
17	17	2	в работе	2025-03-12	[null]
18	18	3	отменен	2025-02-15	[null]

Рис. 11: Состояние таблицы contract до выполнения INSERT

```
INSERT INTO contract (client_id, status, conclusion_date)
SELECT id, 'в работе', CURRENT_DATE
FROM client
WHERE contact_name = 'Петрова А.А.';
```

Листинг 12: Добавление нового договора для клиента через подзапрос

	id [PK] integer	client_id integer	status character varying (50)	conclusion_date date	execution_date date
1	1	1	в работе	2025-03-01	[null]
2	2	2	завершен	2024-11-10	2025-01-20
3	3	3	отменен	2025-01-05	[null]
4	4	4	в работе	2025-03-10	[null]
5	5	5	в работе	2025-02-15	[null]
6	6	6	завершен	2024-12-01	2025-01-30
7	7	7	в работе	2025-04-01	[null]
8	8	8	отменен	2025-02-20	[null]
9	9	9	в работе	2025-01-25	[null]
10	10	10	завершен	2024-10-10	2024-12-15
11	11	11	в работе	2025-03-05	[null]
12	12	12	завершен	2025-01-01	2025-02-25
13	13	13	в работе	2025-03-15	[null]
14	14	14	отменен	2025-02-01	[null]
15	15	15	в работе	2025-04-10	[null]
16	16	1	завершен	2024-09-05	2024-11-01
17	17	2	в работе	2025-03-12	[null]
18	18	3	отменен	2025-02-15	[null]
19	19	2	в работе	2025-04-28	[null]

Рис. 12: Состояние таблицы contract после выполнения INSERT

UPDATE с подзапросом

Перед выполнением запроса проверим статусы деталей.

Обновление статуса договора на «завершен», если все связанные с ним детали имеют статус «выполнена».

	id [PK] integer	status character varying (50)
1	1	проект
2	2	в производстве
3	3	выполнена
4	4	в производстве
5	5	выполнена
6	6	проект
7	7	в производстве
8	8	выполнена
9	9	проект
10	10	в производстве
11	11	выполнена
12	12	проект
13	13	в производстве
14	14	выполнена
15	15	проект
16	16	в производстве
17	17	выполнена
18	18	проект

Рис. 13: Статусы заказов до выполнения UPDATE

```
UPDATE contract ct
SET status = 'завершен'
WHERE NOT EXISTS (
    SELECT 1
    FROM contract_detail cd
    JOIN detail d ON cd.detail_id = d.id
    WHERE cd.contract_id = ct.id
    AND d.status != 'выполнена'
);
```

Листинг 13: Обновление статуса договора на «завершен»

	id [PK] integer	client_id integer	status character varying (50)	conclusion_date date	execution_date date
1	1	1	в работе	2025-03-01	[null]
2	4	4	в работе	2025-03-10	[null]
3	9	9	в работе	2025-01-25	[null]
4	13	13	в работе	2025-03-15	[null]
5	15	15	в работе	2025-04-10	[null]
6	106	10	завершен	2025-01-31	[null]
7	119	6	завершен	2025-01-18	2025-03-09
8	101	11	завершен	2025-02-05	2025-03-18
9	20	6	завершен	2025-04-27	2025-04-27
10	82	12	завершен	2025-02-24	[null]
11	25	10	завершен	2025-04-22	[null]
12	26	16	завершен	2025-04-21	2025-04-24
13	27	5	завершен	2025-04-20	[null]
14	93	12	завершен	2025-02-13	[null]
15	11	11	завершен	2025-03-05	[null]
16	39	16	завершен	2025-04-08	[null]
17	17	2	завершен	2025-03-12	[null]
18	66	7	завершен	2025-03-12	[null]
19	89	18	завершен	2025-02-17	2025-03-24
20	33	10	завершен	2025-04-14	[null]

Рис. 14: Статусы заказов после выполнения UPDATE

DELETE с подзапросом

Перед выполнением запроса проверим наличие этапов изготовления для деталей со статусом «проект».

	id [PK] integer	detail_id integer	machine_id integer	steel_sheet_id integer	execution_date date
1	1	1	1	1	2025-03-20
2	2	2	2	2	2025-03-21
3	3	4	3	3	2025-04-01
4	4	5	4	4	2025-04-02
5	5	6	5	5	2025-04-03
6	6	7	6	6	2025-04-04
7	7	8	7	7	2025-04-05
8	8	9	8	8	2025-04-06
9	9	10	9	9	2025-04-07
10	10	11	10	10	2025-04-08
11	11	12	11	11	2025-04-09
12	12	13	12	12	2025-04-10
13	13	14	13	13	2025-04-11
14	14	15	14	14	2025-04-12
15	15	4	15	15	2025-04-13
16	16	5	3	3	2025-04-14
17	17	6	4	4	2025-04-15

Рис. 15: Этапы изготовления до выполнения DELETE

```
DELETE FROM manufacturing_stage
WHERE detail_id IN (
    SELECT id
    FROM detail
    WHERE status = 'проект'
);
```

Листинг 14: Удаление этапов изготовления для проектных деталей через подзапрос

	id [PK] integer	detail_id integer	machine_id integer	steel_sheet_id integer	execution_date date
1	2	2	2	2	2025-03-21
2	3	4	3	3	2025-04-01
3	4	5	4	4	2025-04-02
4	5	6	5	5	2025-04-03
5	6	7	6	6	2025-04-04
6	7	8	7	7	2025-04-05
7	9	10	9	9	2025-04-07
8	10	11	10	10	2025-04-08
9	11	12	11	11	2025-04-09
10	12	13	12	12	2025-04-10
11	13	14	13	13	2025-04-11
12	15	4	15	15	2025-04-13
13	16	5	3	3	2025-04-14
14	17	6	4	4	2025-04-15

Рис. 16: Этапы изготовления после выполнения DELETE

4. Создание индексов и анализ планов выполнения

Запрос без индекса

Выполняем запрос поиска договоров со статусом «в работе» без использования индекса.

```
EXPLAIN ANALYZE
SELECT *
FROM contract
WHERE status = 'в работе';
```

Листинг 15: Запрос без индекса


	QUERY PLAN	
	text	
1	Seq Scan on contract (cost=0.00..2.49 rows=43 width=32) (actual time=0.015..0.044 rows=43 loops=...	
2	Filter: ((status)::text = 'в работе'::text)	
3	Rows Removed by Filter: 76	
4	Planning Time: 0.088 ms	
5	Execution Time: 0.057 ms	

Рис. 17: План выполнения запроса без индекса

Создание индекса

Создаем индекс на столбец status в таблице contract.

```
CREATE INDEX idx_contract_status ON contract(status);
```

Листинг 16: Создание индекса на contract(status)

Запрос с индексом

Выполняем тот же запрос после создания индекса.

```
EXPLAIN ANALYZE
SELECT *
FROM contract
WHERE status = 'в работе';
```

Листинг 17: Запрос с индексом


	QUERY PLAN	
	text	
1	Seq Scan on contract (cost=0.00..2.49 rows=43 width=32) (actual time=0.010..0.018 rows=43 loops=...	
2	Filter: ((status)::text = 'в работе'::text)	
3	Rows Removed by Filter: 76	
4	Planning Time: 0.821 ms	
5	Execution Time: 0.026 ms	

Рис. 18: План выполнения запроса с индексом

Удаление индекса

После проверки удаляем индекс.

```
DROP INDEX idx_contract_status;
```

Листинг 18: Удаление индекса

Сравнение времени выполнения

По результатам EXPLAIN видно, что после создания индекса время выполнения запроса уменьшилось. - Без индекса: *[0.057 ms]* - С индексом: *[0.026 ms]*

Составной индекс

Запрос без индекса

Выполняем запрос поиска этапов изготовления по конкретной детали за последние 7 дней.

```
EXPLAIN ANALYZE
SELECT *
FROM manufacturing_stage
WHERE detail_id = 3 AND execution_date > CURRENT_DATE - INTERVAL '7 days';
```

Листинг 19: Запрос без составного индекса

	QUERY PLAN text	
1	Seq Scan on manufacturing_stage (cost=0.00..44.00 rows=3 width=20) (actual time=0.072..0.077 rows=1 loops=...	
2	Filter: ((detail_id = 3) AND (execution_date > (CURRENT_DATE - '7 days'::interval)))	
3	Rows Removed by Filter: 313	
4	Planning Time: 0.086 ms	
5	Execution Time: 0.086 ms	

Рис. 19: План выполнения запроса без составного индекса

Создание составного индекса

Создаем индекс на два столбца: detailId и executionDate.

```
CREATE INDEX idx_stage_detail_date ON manufacturing_stage(detail_id,
execution_date);
```

Листинг 20: Создание составного индекса

Запрос с индексом

Повторно выполняем тот же запрос — теперь с использованием составного индекса.

```
EXPLAIN ANALYZE
SELECT *
FROM manufacturing_stage
WHERE detail_id = 3 AND execution_date > CURRENT_DATE - INTERVAL '7 days';
```

Листинг 21: Запрос с составным индексом

	QUERY PLAN
	text
1	Index Scan using idx_stage_detail_date on manufacturing_stage (cost=0.15..8.17 rows=1 width=20) (actual time=0.010..0.011 rows=1 loops=...
2	Index Cond: ((detail_id = 3) AND (execution_date > (CURRENT_DATE - '7 days'::interval)))
3	Planning Time: 0.108 ms
4	Execution Time: 0.022 ms

Рис. 20: План выполнения запроса с составным индексом

Удаление составного индекса

После проверки удаляем созданный составной индекс.

```
DROP INDEX idx_stage_detail_date;
```

Листинг 22: Удаление составного индекса

Сравнение времени выполнения

По результатам EXPLAIN видно, что составной индекс позволяет PostgreSQL эффективно фильтровать строки по двум полям.

- Без индекса: *0.086 ms*; С индексом: *0.022 ms*

Таким образом, при наличии фильтрации сразу по нескольким столбцам, составные индексы обеспечивают значительное ускорение запросов и снижают нагрузку на систему.

Использование индексов позволяет значительно ускорить выполнение запросов, особенно при фильтрации по часто используемым столбцам.

Выводы

В ходе лабораторной работы были разработаны запросы для выборки и модификации данных, созданы представления, проанализированы планы выполнения запросов с использованием индексов. Была подтверждена эффективность применения индексов для ускорения выполнения запросов.