Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ. ПРЕДСТАВЛЕНИЯ. РАБОТА С ИНДЕКСАМИ»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Тимаков Егор Павлович
Факультет прикладной информатики
Группа K3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

Введение

Цель работы.

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

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL(согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).
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 для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Выполнение.

Схема базы данных созданная в postgres ERD изображена на рисунке 1.

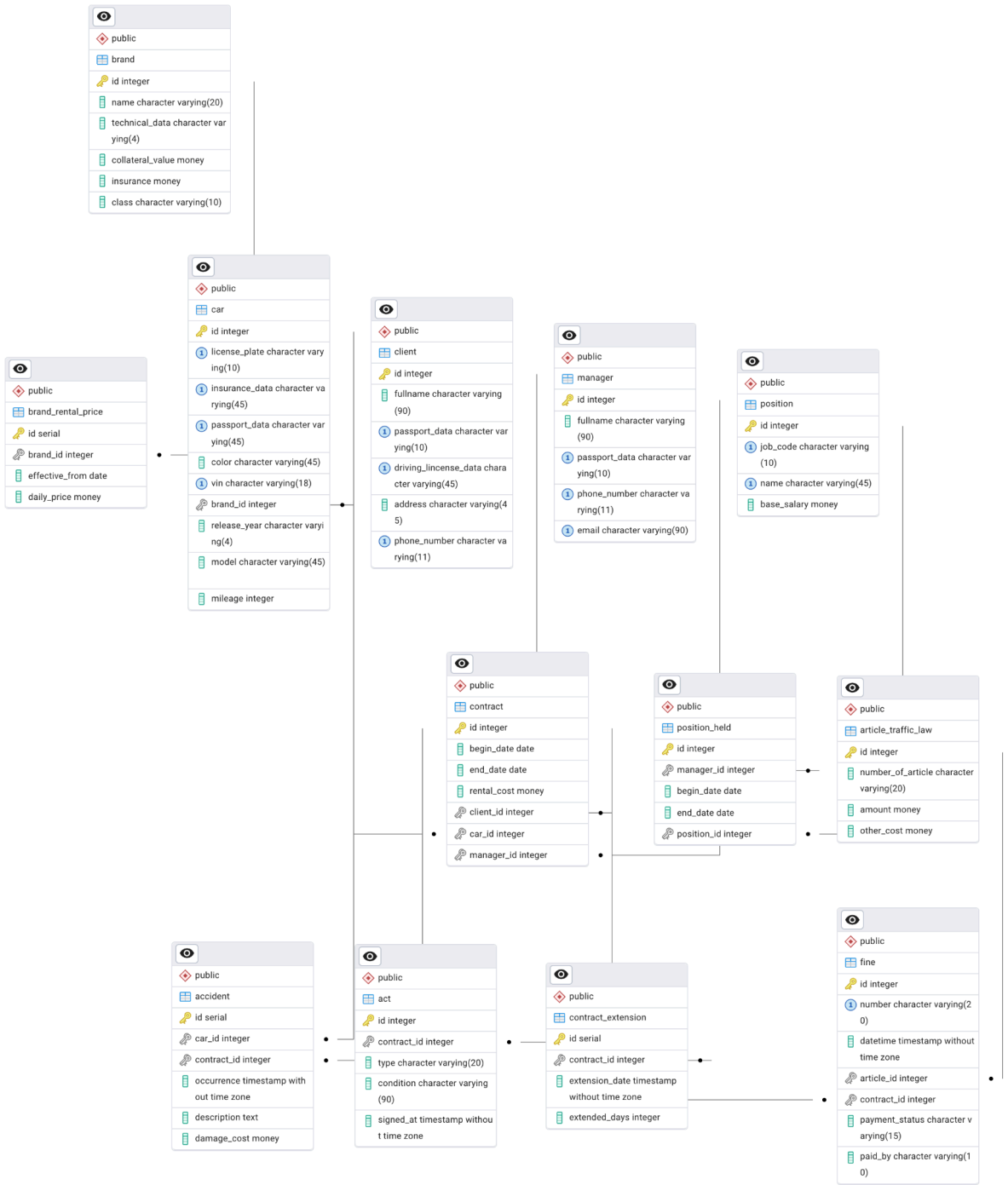


Рисунок 1 - схема базы данных согласно варианту

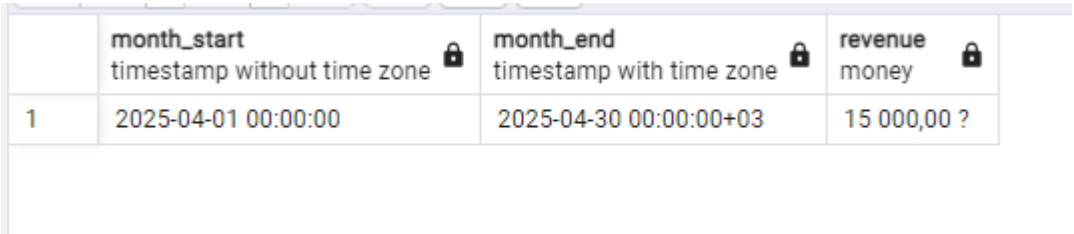
Задание 1:Рассчитать выручку компании за последний календарный месяц.

SQL скрипт выполняющий данный запрос:

```
SELECT
  DATE_TRUNC('month', CURRENT_DATE - INTERVAL '1 month') AS
month_start,
```

```
    DATE_TRUNC('month', CURRENT_DATE) - INTERVAL '1 day' AS month_end,
    SUM(Rental_cost) AS revenue
FROM
    Contract
WHERE
    Begin_Date >= DATE_TRUNC('month', CURRENT_DATE - INTERVAL '1
month') AND
    Begin_Date < DATE_TRUNC('month', CURRENT_DATE);
```

Результат выполнения скрипта можно увидеть на изображении 2.



	month_start timestamp without time zone	month_end timestamp with time zone	revenue money
1	2025-04-01 00:00:00	2025-04-30 00:00:00+03	15 000,00 ?

рисунок 2 - результат выполнения задания 1

Задание 2: Автомобили какой марки чаще всего брались в прокат?
SQL скрипт выполняющий данный запрос:

```
WITH Brand_Rentals AS (
    SELECT
        b.name AS brand_name,
        COUNT(*) AS rental_count
    FROM
        Contract c
    JOIN
        Car car ON c.Car_id = car.ID
    JOIN
        Brand b ON car.Brand_id = b.ID
    GROUP BY
        b.name
),
Max_Count AS (
    SELECT MAX(rental_count) AS max_rentals FROM Brand_Rentals
)
SELECT
    br.brand_name,
    br.rental_count
FROM
    Brand_Rentals br
```

```
JOIN
   Max_Count mc ON br.rental_count = mc.max_rentals;
```

Результат выполнения скрипта можно увидеть на изображении 2.

	brand_name character varying (20)	rental_count bigint
1	TOYOTA	2

Рисунок 3 - результат выполнения задания 2

Задание 3: Определить убытки от простоя автомобилей за вчерашний день.

SQL скрипт выполняющий данный запрос:

```
SELECT
   SUM(brp.Daily_Price) AS downtime_loss
FROM
   Car c
      JOIN
   Brand b ON c.Brand_id = b.ID
      JOIN
   Brand_Rental_Price brp ON b.ID = brp.Brand_ID
WHERE
   brp.Effective_From <= CURRENT_DATE - INTERVAL '1 day'
  AND NOT EXISTS (
   SELECT 1 FROM Contract ct
   WHERE ct.Car_id = c.ID
     AND DATE(CURRENT_DATE - INTERVAL '1 day') BETWEEN ct.Begin_Date
AND ct.End_date);
```

Результат выполнения скрипта можно увидеть на изображении 4.

	downtime_loss money
1	8 500,00 ?

Рисунок 4 - результат выполнения задания 3

Задание 4: Вывести данные автомобиля, имеющего максимальный пробег.

SQL скрипт выполняющий данный запрос:

```
WITH MaxMileage AS (
   SELECT MAX(mileage) AS max_mileage FROM Car
)
SELECT
   car.ID,
```

```
    car.license_plate,
    car.vin,
    car.mileage
FROM
    Car
JOIN
    MaxMileage mm ON car.mileage = mm.max_mileage;
```

Результат выполнения скрипта можно увидеть на изображении 5.

	id [PK] integer	license_plate character varying (10)	vin character varying (18)	mileage integer
1	2	A456AA77	1FAFP34P01W123456	293021

Рисунок 5 - результат выполнения задания 4

Задание 5: Какой автомобиль суммарно находится в прокате дольше всех.

SQL скрипт выполняющий данный запрос:

```
WITH Car_Total_Rent AS (
    SELECT
        car.ID,
        car.License_plate,
        car.VIN,
        SUM(c.End_date::date - c.Begin_Date::date) AS total_days
    FROM
        Car car
    JOIN
        Contract c ON car.ID = c.Car_id
    GROUP BY
        car.ID, car.License_plate, car.VIN
),
MaxRent AS (
    SELECT MAX(total_days) AS max_total FROM Car_Total_Rent
)
SELECT
    ctr.ID,
    ctr.License_plate,
```

```
    ctr.VIN,
    ctr.total_days
FROM
    Car_Total_Rent ctr
JOIN
    MaxRent mr ON ctr.total_days = mr.max_total;
```

Результат выполнения скрипта можно увидеть на изображении 6.

	id [PK] integer	license_plate character varying (10)	vin character varying (18)	total_days bigint
1	3	A789AA77	JTDBR32E520123457	5

Рисунок 6 - результат выполнения задания 5

Задание 6: Определить, каким количеством автомобилей каждой марки и модели владеет компания.

SQL скрипт выполняющий данный запрос:

```
SELECT
    b.name AS brand,
    car.model AS model,
    COUNT(*) AS car_count
FROM
    Car car
        JOIN
    Brand b ON car.Brand_id = b.ID
GROUP BY
    b.name, car.model
ORDER BY
    car_count DESC;
```

Результат выполнения скрипта можно увидеть на изображении 7.

	brand character varying (20)	model character varying (45)	car_count bigint
1	TOYOTA	Corolla	2
2	FORD	Focus	1

Рисунок 7 - результат выполнения задания 6

Задание 7: Определить средний “возраст” автомобилей компании.

SQL скрипт выполняющий данный запрос:

```
SELECT
   ID,
   license_plate,
   ROUND(mileage::INTEGER / (EXTRACT(YEAR FROM
CURRENT_DATE) - release_year::INTEGER + 1), 2) as avg_year
FROM
   car
where release_year is not null and mileage is not null;
```

Результат выполнения скрипта можно увидеть на изображении 8.

	id [PK] integer	license_plate character varying (10)	avg_year numeric
1	1	A123AA77	3333.33
2	2	A456AA77	41860.14
3	3	A789AA77	1500.00

Рисунок 8 - результат выполнения задания 7

Далее по заданию нам необходимо создать представления.

Задание 1: Какой автомобиль не был в прокате?

SQL скрипт выполняющий данный запрос:

```
CREATE VIEW Unrented_Cars AS
SELECT
   c.ID,
   c.License_plate,
   c.VIN
FROM
   Car c
WHERE
   NOT EXISTS (
      SELECT 1 FROM Contract con WHERE con.Car_id = c.ID
   );
SELECT * FROM Unrented_Cars
```

Результат выполнения скрипта можно увидеть на изображении 9.

id integer	license_plate character varying (10)	vin character varying (18)

Рисунок 9 - результат представления для задания 1

Задание 2: Вывести данные клиентов, не вернувших автомобиль вовремя:
SQL скрипт выполняющий данный запрос:

```
CREATE VIEW Late_Clients AS
SELECT
   cl.ID AS client_id,
   cl.Fullname AS client_name,
   car.License_plate,
   con.ID AS contract_id,
   con.End_date,
   COALESCE(SUM(ext.Extended_Days), 0) AS extended_days,
   (con.End_date::date + make_interval(days =>
COALESCE(SUM(ext.Extended_Days), 0)::INTEGER))::date AS
expected_return_date
FROM
   Client cl
JOIN
   Contract con ON cl.ID = con.Client_id
JOIN
   Car car ON con.Car_id = car.ID
LEFT JOIN
   Contract_Extension ext ON con.ID = ext.Contract_ID
WHERE
   NOT EXISTS (
      SELECT 1 FROM Act a
      WHERE a.Contract_ID = con.ID AND a.Type = 'Прием'
   )
GROUP BY
   cl.ID, cl.Fullname, car.License_plate, con.ID, con.End_date
HAVING
   (con.End_date::date + COALESCE(SUM(ext.Extended_Days)::INTEGER,
0)) < CURRENT_DATE;
SELECT * FROM Late_Clients;
```

Результат выполнения скрипта можно увидеть на изображении 10.

	client_id integer	client_name character varying (90)	license_plate character varying (10)	contract_id integer	end_date date	extended_days bigint	expected_return_date date
1	2	Петров Петр	A456AA77	2	2025-05-21	4	2025-05-25

Рисунок 10 - результат представления для задания 2

Далее мы выполним запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

Задание: Увеличить пробег у всех автомобилей, которые дольше всего находились в прокате

SQL скрипт выполняющий данный запрос:

```
UPDATE Car
SET mileage = mileage + 1000
WHERE ID IN (
   SELECT Car_id
   FROM Contract
   GROUP BY Car_id
   ORDER BY SUM(End_date::date - Begin_Date::date) DESC
   LIMIT 1
);
```

Данные до выполнения скрипта можно увидеть на изображении 11.

	id [PK] integer	license_plate character varying (10)	insurance_data character varying (45)	passport_data character varying (45)	color character varying (45)	vin character varying (18)	brand_id integer	release_year character varying (4)	model character varying (45)	mileage integer
1	3	A789AA77	3456789012	3333333333	белый	JTDBR32E520123457	1	2025	Corolla	56500
2	2	A456AA77	2345678901	2222222222	черный	1FAFP34P01W123456	2	2019	Focus	308021
3	1	A123AA77	1234567890	1111111111	черный	JTDBR32E520123456	1	2020	Corolla	35000

Рисунок 11 - данные до обновления

Данные после выполнения скрипта можно увидеть на изображении 12.

	id [PK] integer	license_plate character varying (10)	insurance_data character varying (45)	passport_data character varying (45)	color character varying (45)	vin character varying (18)	brand_id integer	release_year character varying (4)	model character varying (45)	mileage integer
1	3	A789AA77	3456789012	3333333333	белый	JTDBR32E520123457	1	2025	Corolla	57500
2	2	A456AA77	2345678901	2222222222	черный	1FAFP34P01W123456	2	2019	Focus	308021
3	1	A123AA77	1234567890	1111111111	черный	JTDBR32E520123456	1	2020	Corolla	35000

Рисунок 12 - данные после обновления

Задание: Удалить клиентов, у которых не было ни одного контракта.

SQL скрипт выполняющий данный запрос:

```
DELETE FROM Client
WHERE ID NOT IN (
   SELECT DISTINCT Client_id
   FROM Contract
);
```

Данные до выполнения скрипта можно увидеть на изображении 13.

	id [PK] integer	contract_id integer	extension_date timestamp without time zone	extended_days integer
1	1	2	2025-05-22 00:00:00	2
2	2	2	2025-05-22 00:00:00	2

Рисунок 13 - данные до обновления

Данные после выполнения скрипта можно увидеть на изображении 14.

	id [PK] integer	fullname character varying (90)	passport_data character varying (10)	driving_lincense_data character varying (45)	address character varying (45)	phone_number character varying (11)
1	1	Иванов Иван	1234567890	111222333	Москва	89161234567
2	2	Петров Петр	0987654321	444555666	Санкт Петербург	89161234568

Рисунок 14 - данные после обновления

Задание: Вставить в таблицу “Act” акты типа ”Передача” по всем контрактам, у которых нет ни одного акта .

SQL скрипт выполняющий данный запрос:

```
INSERT INTO Act (Contract_ID, Type, Condition, Signed_At)
SELECT
    c.ID,
    'Передача',
    'Без повреждений',
    NOW()
FROM Contract c
WHERE NOT EXISTS (
    SELECT 1 FROM Act a WHERE a.Contract_ID = c.ID
);
SELECT * FROM Act;
```

Данные до выполнения скрипта можно увидеть на изображении 17.

	id [PK] integer	contract_id integer	type character varying (20)	condition character varying (90)	signed_at timestamp without time zone
1	1	1	Прием	Хорошее	2025-05-12 00:00:00

Рисунок 17 - данные до обновления

Данные после выполнения скрипта можно увидеть на изображении 18.

	id [PK] integer	contract_id integer	type character varying (20)	condition character varying (90)	signed_at timestamp without time zone
1	1	1	Прием	Хорошее	2025-05-12 00:00:00
2	2	2	Передача	Без повреждений	2025-05-26 06:08:57.513037
3	3	3	Передача	Без повреждений	2025-05-26 06:08:57.513037

Рисунок 18 - данные после обновления

Далее мы создаем индексы и сравниваем время работы с ними и без них.
SQL скрипт без индекса:
EXPLAIN ANALYZE
SELECT * FROM Contract
WHERE Client_id = 1;
Результат выполнения скрипты можно увидеть на рисунок 19.

	QUERY PLAN text
1	Seq Scan on contract (cost=0.00..27.00 rows=7 width=32) (actual time=0.012..0.013 rows=2 loops...
2	Filter: (client_id = 1)
3	Rows Removed by Filter: 1
4	Planning Time: 0.058 ms
5	Execution Time: 0.028 ms

Рисунок 19 - результат без индекса

SQL скрипт, который создает индекс:
CREATE INDEX idx_contract_client ON Contract(Client_id);

Теперь повторим запрос уже с созданным индексом. Результат выполнения запроса с индексом можно увидеть на рисунке 20.

	QUERY PLAN text
1	Seq Scan on contract (cost=0.00..1.04 rows=1 width=32) (actual time=0.006..0.007 rows=2 loops...
2	Filter: (client_id = 1)
3	Rows Removed by Filter: 1
4	Planning Time: 0.673 ms
5	Execution Time: 0.014 ms

Рисунок 19 - результат без индекса

Теперь удалим индекс:
DROP INDEX idx_contract_client;

Из результатов выполнения скриптов с индексом и без индекса мы видим, что при использовании индекса время исполнения в 2 раза меньше, чем без него.

Вывод:

В результате выполнения работы мы изучили сложные SQL-запрос. Также исполнили запросы на изменения данных с подзапросами. Еще мы поняли, что индексы значительно сокращают время работы скрипта.