ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий

Дисциплина:
«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ, ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В POSTGRESQL»

Выполнил:
студент группы K32392
Жаров Александр Павлович

_____
(подпись)

Проверил(а):
Говорова Марина Михайловна

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург
2023 г.

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, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и посмотреть 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.

Индивидуальное практическое задание:

База данных “Прокат автомобилей”

Задание 2. Создать запросы:

- Какой автомобиль находился в прокате максимальное количество часов?
- Автомобили какой марки чаще всего брались в прокат?
- Определить убытки от простоя автомобилей за вчерашний день.
- Вывести данные автомобиля, имеющего максимальный пробег.
- Какой автомобиль суммарно находился в прокате дольше всех.
- Определить, каким количеством автомобилей каждой марки и модели владеет компания.
- Определить средний “возраст” автомобилей компании.

Задание 3. Создать представление:

- Какой автомобиль ни разу не был в прокате?
- Вывести данные клиентов, не вернувших автомобиль вовремя.

Схема базы данных:

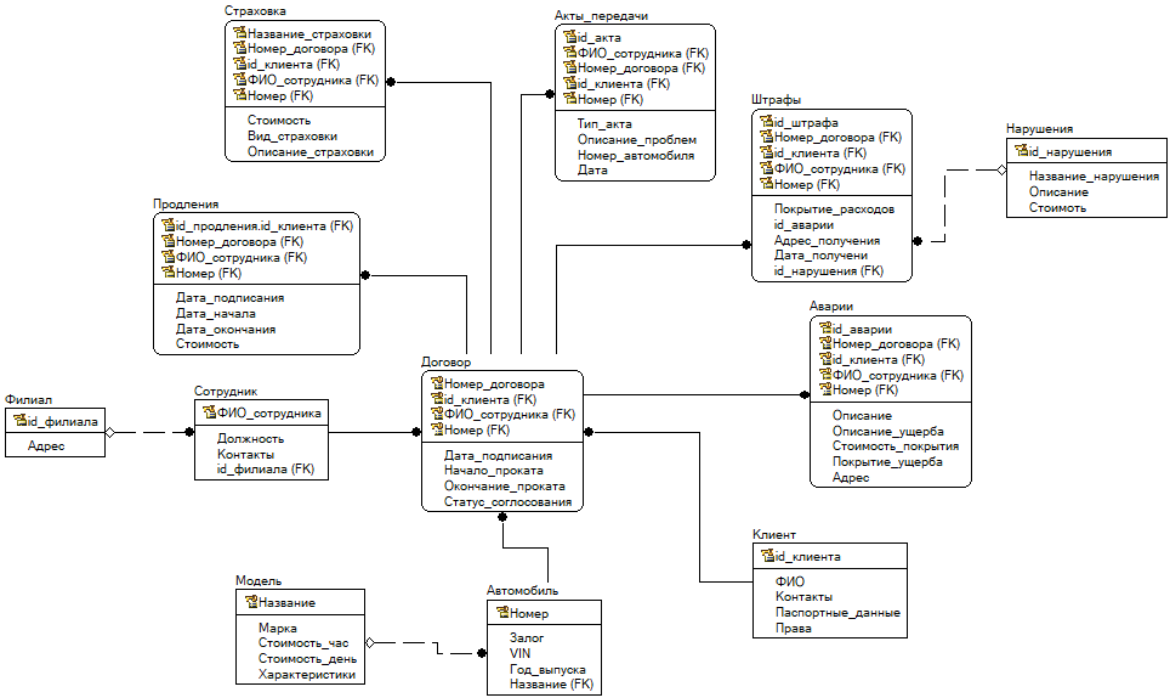

Рис. 1 - Схема базы данных

Выполнение

Запросы на выборку

1. Какой автомобиль находился в прокате максимальное количество часов?

```
SELECT (end_of_rental - start_of_rental) AS hours_diff, co.car_number
FROM "LR_2".contract co
JOIN "LR_2".car ca
ON co.car_number = ca.car_number
ORDER BY co.end_of_rental DESC;
```

	hours_diff integer	car_number integer
1	6	123
2	5	123
3	5	124

Рис. 2 - SELECT

2. Автомобили какой марки чаще всего брались в прокат?

```
SELECT m.name, COUNT(*) AS num_rentals
FROM "LR_2".contract c
JOIN "LR_2".car cr ON c.car_number = cr.car_number
JOIN "LR_2".model m ON cr.car_model = m. car_model
```

```
GROUP BY m.name
ORDER BY num_rentals DESC;
```

	name text	num_rentals bigint
1	Volkswagen	2
2	Volvo	1

Рис. 3 - SELECT

3. Определить убытки от простоя автомобилей за вчерашний день.

```
SELECT car_number
FROM "LR_2".car
WHERE car_number NOT IN (
  SELECT car_number
  FROM "LR_2".acts_of_transfer
  WHERE type_of_act = 'Передача' AND date = current_date - interval '1 day'
);
```

	car_number [PK] integer
1	123
2	124
3	125
4	1
5	2
6	3
7	4
8	5
9	6
10	7

Рис. 4 - SELECT

4. Вывести данные автомобиля, имеющего максимальный пробег.

```
SELECT *
FROM "LR_2".car
ORDER BY car_mileage DESC;
```

	car_number [PK] integer	vin integer	year_of_issue date	car_model text	car_mileage integer
1	90	7890165	2010-01-01	Polo	250000
2	65	7890165	2010-01-01	Polo	250000
3	64	6789054	2021-01-01	Q5	240000
4	89	6789054	2021-01-01	Q5	240000
5	63	5678943	2020-01-01	XC60	230000
6	88	5678943	2020-01-01	XC60	230000
7	87	4567832	2019-01-01	XC90	220000
8	62	4567832	2019-01-01	XC90	220000
9	86	3456720	2018-01-01	Polo	210000
10	61	3456720	2018-01-01	Polo	210000
11	60	2345609	2017-01-01	Q5	200000
12	85	2345609	2017-01-01	Q5	200000
13	84	1234598	2016-01-01	XC60	190000
14	59	12359876	2016-01-01	XC60	190000
15	58	901238765	2015-01-01	XC90	180000
16	83	9012387	2015-01-01	XC90	180000
17	57	890127654	2014-01-01	Polo	170000
18	82	8901276	2014-01-01	Polo	170000
19	56	789016543	2013-01-01	Q5	160000

Рис. 5 - SELECT

5. Какой автомобиль суммарно находился в прокате дольше всех.

```
SELECT
    co.car_number,
    SUM(end_of_rental - start_of_rental) AS total_rental_hours
FROM
    "LR_2".contract co
    JOIN "LR_2".car ca ON co.car_number = ca.car_number
GROUP BY
    co.car_number
ORDER BY
    total_rental_hours DESC;
```

	car_number integer	total_rental_hours bigint
1	123	11
2	124	5

Рис. 6 - SELECT

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

```
SELECT m.car_model, m.name, COUNT(c.car_number) AS number_of_cars
FROM "LR_2".car c
right JOIN "LR_2".model m ON c.car_model = m. car_model
GROUP BY m.car_model, m.name;
```

	car_model [PK] text	name text	number_of_cars bigint
1	Polo	Volkswagen	25
2	XC60	Volvo	22
3	Q5	Audi	22
4	XC90	Volvo	23

Рис. 7 - SELECT

7. Определить средний “возраст” автомобилей компании.

```
SELECT AVG(CURRENT_DATE - year_of_issue)/365 as year
FROM "LR_2".car;
```

	year numeric
1	5.2778141751042287

Рис. 8 - SELECT

Представления

1. Какой автомобиль ни разу не был в прокате?

```
CREATE VIEW "LR_2".car_not_in_rent AS
SELECT car.car_number
FROM "LR_2".car car
LEFT JOIN "LR_2".contract con ON car.car_number = con.car_number
WHERE con.car_number IS NULL;
```

	car_number integer
1	125
2	1
3	2
4	3
5	4
6	5
7	6

Рис. 9 - CREATE VIEW

2. Вывести данные клиентов, не вернувших автомобиль вовремя.

```
CREATE VIEW "LR_2".clients_info AS
SELECT cl.id_client, cl.full_name, cl.contacts
FROM "LR_2".acts_of_transfer act
JOIN "LR_2".contract con ON act.contract_number = con.id_contract
JOIN "LR_2".clients cl ON con.id_client = cl.id_client
WHERE act.date > con.end_of_rental AND act.type_of_act = 'Прием';
```

	id_client integer	full_name text	contacts text
1	1	Сергей Иванович Иванов	+793455345436
2	1	Сергей Иванович Иванов	+793455345436

Рис. 10 - CREATE VIEW

DELETE INSERT UPDATE

1. Добавить нового сотрудника в филиал, если там нет ни одного сотрудника

```
INSERT INTO "LR_2".worker (full_name, post, id_branch, contacts)
SELECT 'Новый сотрудник', 'Должность', b.id_branch, '-'
FROM "LR_2".branch b
WHERE NOT EXISTS (
    SELECT 1
    FROM "LR_2".worker w
    WHERE w.id_branch = b.id_branch
    );
```

2. Удаление данных моделей, которых нет в автопарке

```
DELETE FROM "LR_2".model
WHERE name IN (
    SELECT m.name
```

```
        FROM "LR_2".model m
        LEFT JOIN "LR_2".car c ON m.name = c.car_model
        GROUP BY m.name
        HAVING COUNT(c.car_number) = 0
    );
```

3. Повысить до главы отдела работника, сделавшего 5 контрактов

```
UPDATE "LR_2".worker w
SET post = 'глава отдела'
WHERE EXISTS (
        SELECT 1
        FROM "LR_2".contract c
        WHERE c.workers_full_name = w.full_name
        GROUP BY c.workers_full_name
        HAVING COUNT(*) >= 5
);
```

Индексация

Простой индекс:

```
SELECT *
FROM "LR_2".car
WHERE car_mileage > 200000;

CREATE INDEX mileage
ON "LR_2".car (car_mileage);
```

При помощи индексации удалось ускорить запрос с 68мс. до 62мс. В качестве результата бралось среднее время 10 запросов.

Составной индекс:

```
SELECT m.car_model, m.name, COUNT(c.car_number) AS number_of_cars
FROM "LR_2".car c
right JOIN "LR_2".model m ON c.car_model = m. car_model
GROUP BY m.car_model, m.name;

CREATE INDEX car_index ON "LR_2".model (car_model, name);
```

При помощи индексации удалось ускорить запрос с 105мс. до 75мс. В качестве результата бралось среднее время 10 запросов.

Graphical | Analysis | Statistics

#	Node
1.	→ Aggregate
2.	→ Hash Right Join **Hash Cond**: (c.car_model = m.car_model)
3.	→ Seq Scan on car as c
4.	→ Hash
5.	→ Seq Scan on model as m

Рис. 10 - EXPLAIN

Выводы

В процессе выполнения лабораторной работы я ознакомился с созданием запросов INSERT, UPDATE и DELETE, а также с графическим представлением запросов. Я также изучил, как создавать простые и составные индексы, и заметил, что это позволяет сокращать количество этапов выполнения запросов и снижать время выполнения.