

**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 К32392

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:


```

JOIN "LR_2".model m ON cr.car_model = m.car_model
GROUP BY m.name
HAVING COUNT(*) = (
    SELECT MAX(num_rentals)
    FROM (
        SELECT 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
    ) subquery
)
ORDER BY num_rentals DESC;

```

	name text	num_rentals bigint
1	Volkswagen	2

Рис. 3 - SELECT

3. Определить убытки от простоя автомобилей за вчерашний день.

```

SELECT SUM(m.cost_per_day) AS total_loss
FROM "LR_2".car c
JOIN "LR_2".model m ON c.car_model = m.car_model
WHERE c.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'
);

```

	total_loss bigint
1	811000

Рис. 4 - SELECT

4. Вывести данные автомобиля, имеющего максимальный пробег.

```

SELECT *
FROM "LR_2".car

```

WHERE car_mileage = (SELECT MAX(car_mileage) FROM "LR_2".car);

	car_number [PK] integer	vin integer	year_of_issue date	car_model text	car_mileage integer
1	65	7890165	2010-01-01	Polo	250000
2	90	7890165	2010-01-01	Polo	250000

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
HAVING
    SUM(end_of_rental - start_of_rental) = (
        SELECT
            MAX(total_hours)
        FROM
            (SELECT
                SUM(end_of_rental - start_of_rental) AS total_hours
            FROM
                "LR_2".contract
            GROUP BY
                car_number) AS subquery
    );

```

	car_number integer	total_rental_hours bigint
1	123	11

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

- Определить средний “возраст” автомобилей компании.

```
SELECT ROUND(AVG((CURRENT_DATE - year_of_issue) / 365.0), 1) as year
FROM "LR_2".car;
```

	year numeric
1	5.3

Рис. 8 - SELECT

Представления

- 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

- 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
);

```

	full_name [PK] text	post text	contacts text	id_branch integer
1	Александр Александрович Иванов	Директор отдела	+798434519345	1
2	Иван Иванович Иванов	Сотрудник	+7984343243245	2
3	Иван Александрович Петров	Сотрудник	+7984343243286	3
4	Александр Павлович Иванов	Сотрудник	+79843452349	2
5	Новый сотрудник	Должность	-	4

Рис. 11 - INSERT

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
);
```

	car_model [PK] text	name text	cost_per_hour integer	cost_per_day integer	characteristics text
1	XC90	Volvo	1300	7000	235 л.с
2	Polo	Volkswagen	800	4000	110 л.с
3	XC60	Volvo	1500	10000	235 л.с
4	Q5	Audi	2000	15000	235 л.с

Рис. 12 - DELETE

- 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
);
```

	full_name [PK] text	post text	contacts text	id_branch integer
1	Александр Александрович Иванов	Директор отдела	+798434519345	1
2	Иван Иванович Иванов	Сотрудник	+7984343243245	2
3	Иван Александрович Петров	Сотрудник	+7984343243286	3
4	Александр Павлович Иванов	Сотрудник	+79843452349	2
5	Новый сотрудник	Глава отдела	-	4

Рис. 13 - UPDATE

Индексация

Простой индекс:

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

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


Graphical	Analysis	Statistics
#	Node	
1.	→ Seq Scan on car as car Filter: (car_mileage > 200000)	

Рис. 14 - EXPLAIN

При помощи индексации удалось ускорить запрос с 68мс. до 62мс. Хотя порядок выполнения никак не изменился. 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. 15 - 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.